Centre for Central Banking Studies

Applied Bayesian econometrics for central bankers

Andrew Blake and Haroon Mumtaz

BANK OF ENGLAND

Applied Bayesian Econometrics for Central Bankers

Andrew Blake

Haroon Mumtaz

Center For Central Banking Studies, Bank of England
E-mail address: Andrew.Blake@bankofengland.co.uk

Queen Mary, University of London
E-mail address: h.mumtaz@qmul.ac.uk

Abstract. The aim of this handbook is to introduce key topics in Bayesian econometrics from an applied perspective.

The handbook assumes that readers have a fair grasp of basic classical econometrics (e.g. maximum likelihood estimation). It is recommended that readers familiarise themselves with the Matlab© programming language to derive the maximum benefit from this handbook. A basic guide to Matlab© is provided in the appendix to this handbook.

The first chapter of the handbook introduces basic concepts of Bayesian analysis. In particular, the chapter focuses on the technique of Gibbs sampling and applies it to a linear regression model. The chapter shows how to code this algorithm via several practical examples. The second chapter introduces Bayesian vector autoregressions (VARs) and discusses how Gibbs sampling can be used for these models. The third chapter shows how Gibbs sampling can be applied to popular econometric models such as time-varying VARS and dynamic factor models. The fourth chapter applies Gibbs sampling to Markov Switching models. The next chapter introduces the Metropolis Hastings algorithm which is applied to DSGE model estimation in Chapter 6. The final chapter considers advanced models such as dynamic factor models with time-varying parameters. We intend to introduce new topics in revised versions of this handbook on a regular basis.

The handbook comes with a set of Matlab© codes that can be used to replicate the examples in each chapter. The code (provided in code.zip) is organised by chapter. For example, the folder 'Chapter1' contains all the examples referred to in the first chapter of this handbook. The views expressed in this handbook are those of the authors, and not necessarily those of the Bank of England. The reference material and computer codes are provided without any guarantee of accuracy. The authors would appreciate feedback on possible coding errors and/or typos.

Contents

Part 1

A Practical Introduction to Gibbs Sampling

CHAPTER 1

Gibbs Sampling for Linear Regression Models

1. Introduction

This chapter provides an introduction to the technique of estimating linear regression models using Gibbs sampling. While the linear regression model is a particularly simple case, the application of Gibbs sampling in this scenario follows the same principles as the implementation in a more complicated models (considered in later chapters) and thus serves as a useful starting point. We draw heavily on the seminal treatment of this topic in Kim and Nelson (1999). A more formal (but equally accessible) reference is Koop (2003).

The reader should aim to become familiar with the following with the help of this chapter

- The prior distribution, the posterior distribution and Bayes Theorem.
- Bayesian treatment of the linear regression model.
- Why Gibbs sampling provides a convenient estimation method.
- Coding the Gibbs sampling algorithm for a linear regression in Matlab

2. A Bayesian approach to estimating a linear regression model

Consider the task of estimating the following regression model

$$\begin{aligned} Y_t &= BX_t + v_t \\ v_t &\sim N\left(0, \sigma^2\right) \end{aligned} \tag{2.1}$$

where Y_t is a $T \times 1$ matrix of the dependent variable, Y_t is a $T \times K$ matrix of the independent variables and deterministic terms. We are concerned with estimating the $K \times 1$ vector of coefficients B and the variance of the error term σ^2.

A classical econometrician proceeds by obtaining data on Y_t and X_t and writes down the likelihood function of the model

$$F\left(Y_t | B, \sigma^2\right) = \left(2\pi\sigma^2\right)^{-T/2} \exp\left(-\frac{\left(Y_t - BX_t\right)' \left(Y_t - BX_t\right)}{2\sigma^2}\right) \tag{2.2}$$

and obtains estimates $\hat{B}$ and $\hat{\sigma}^2$ by maximising the likelihood. In this simple case these deliver the familiar OLS estimator for the coefficients $\hat{B}_{OLS} = \left(X_t'X_t\right)^{-1}\left(X_t'Y_t\right)$ and the (biased) maximum likelihood estimator for the error variance $\hat{\sigma}^2 = \frac{v_t'v_t}{T}$. For our purpose, the main noteworthy feature of the classical approach is the fact that the estimates of the parameters of the model are solely based on information contained in data.

Bayesian analysis departs from this approach by allowing the researcher to incorporate her prior beliefs about the parameters B and σ^2 into the estimation process. To be exact, the Bayesian econometrician when faced with the task of estimating equation 2.1 would proceed in the following steps.

Step 1. The researcher forms a prior belief about the parameters to be estimated. This prior belief usually represents information that the researcher has about B and σ^2 which is not derived using the data Y_t and X_t. These prior beliefs may have been formed through past experience or by examining studies (estimating similar models) using other datasets. (We will discuss the merits of this approach for specific examples in the chapters below). The key point is that these beliefs are expressed in the form of a probability distribution. For example, the prior on the coefficients B is expressed as

$$P(B) \sim N\left(B_0, \Sigma_0\right) \tag{2.3}$$

where the mean B_0 represents the actual beliefs about the elements of B.

Example 1. *In the case of two explanatory variables, the vector* $B_0 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$ *represents the belief that the first coefficient equals* 1 *and the second equals* -1. *The variance of the prior distribution* Σ_0 *controls how strong this prior belief is. A large number for* Σ_0 *would imply that the researcher is unsure about the numbers she has chosen for* B_0 *and wants to place only a small weight on them. In contrast, a very small number for* Σ_0 *implies that the researcher is very sure about the belief expressed in* B_0.*In the case of two explanatory variables* Σ_0 *may equal* $\Sigma_0 = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix}$ *representing a 'loose prior' or an uncertain prior belief.*

Step 2. The researcher collects data on Y_t and X_t and write down the likelihood function of the model

$$F\left(Y_t|B,\sigma^2\right) = \left(2\pi\sigma^2\right)^{-T/2} \exp\left(-\frac{\left(Y_t - BX_t\right)'\left(Y_t - BX_t\right)}{2\sigma^2}\right)$$

This step is identical to the approach of the classical econometrican and represents the information about the model parameters contained in the data.

Step 3. The researcher updates her prior belief on the model parameters (formed in step 1) based on the information contained in the data (using the likelihood function in step 2). In other words, the researcher combines the prior distribution $P\left(B,\sigma^2\right)$ and the likelihood function $F\left(Y_t|B,\sigma^2\right)$ to obtain the *posterior* distribution $H\left(B,\sigma^2|Y_t\right)$.

More formally, the Bayesian econometrician is interested in the posterior distribution $H\left(B,\sigma^2|Y_t\right)$ which is defined by the *Bayes Law*

$$H\left(B,\sigma^2|Y_t\right) = \frac{F\left(Y_t|B,\sigma^2\right) \times P\left(B,\sigma^2\right)}{F(Y)} \tag{2.4}$$

Equation 2.4 simply states that the posterior distribution is a product of the likelihood $F\left(Y_t|B,\sigma^2\right)$ and the prior $P\left(B,\sigma^2\right)$ divided by the density of the data $F(Y)$ (also referred to as the marginal likelihood or the marginal data density). Note that $F(Y)$ is a scalar and will not have any operational significance as far as estimation is concerned (although it is crucial for model comparison, a topic we return to). Therefore the Bayes Law can be written as

$$H\left(B,\sigma^2|Y_t\right) \propto F\left(Y_t|B,\sigma^2\right) \times P\left(B,\sigma^2\right) \tag{2.5}$$

Equation 2.5 states that the posterior distribution is proportional to the likelihood times the prior. In practice, we will consider equation 2.5 when considering the estimation of the linear regression model.

As an aside note that the Bayes law in equation 2.4 can be easily derived by considering the joint density of the data Y_t and parameters B,σ^2, $G\left(Y_t,B,\sigma^2\right)$ and observing that ir can be factored in two ways

$$G\left(Y_t,B,\sigma^2\right) = F(Y_t) \times H(B,\sigma^2|Y_t) = F(Y_t|B,\sigma^2) \times P(B,\sigma^2)$$

That is the joint density $G\left(Y_t,B,\sigma^2\right)$ is the product of the marginal density of Y_t and the conditional density of the parameters $H(B,\sigma^2|Y_t)$. Or equivalently the joint density is the product of the conditional density of the data and the marginal density of the parameters. Rearranging the terms after the first equality leads to equation 2.4.

These steps in Bayesian analysis have a number of noteworthy features. First, the Bayesian econometrician is interested in the posterior *distribution* and not the mode of the likelihood function. Second, this approach combines prior information with the information in the data. In contrast, the classical econometrician focusses on information contained in the data about the parameters as summarised by the likelihood function.

To motivate the use of Gibbs sampling for estimating $H(B,\sigma^2|Y_t)$ we will consider the derivation of the posterior distribution in three circumstances. First we consider estimating the posterior distribution of B under the assumption that σ^2 is known. Next we consider estimating the posterior distribution of σ^2 under the assumption that B is known and finally we consider the general case when both sets of parameters are unknown.

2.1. Case 1: The posterior distribution of B assuming σ^2 is known.

Consider the scenario where the econometrician wants to estimate β in equation 2.1 but knows the value of σ^2 already. As discussed above, the posterior distribution is derived using three steps.

Setting the prior. In the first step the researcher sets the prior distribution for β. A normally distributed prior $P\left(B\right) \sim N\left(B_0,\Sigma_0\right)$ for the coefficients is a *conjugate* prior. That is, when this prior is combined with the likelihood function this results in a posterior with the same distribution as the prior. Since the form of the posterior is known when using conjugate priors these are especially convenient from a practical point of view. The prior distribution is given by the following equation

$$\begin{aligned} & (2\pi)^{-K/2}\left|\Sigma_0\right|^{-\frac{1}{2}} \exp\left[-0.5\left(B-B_0\right)'\Sigma_0^{-1}\left(B-B_0\right)\right] \\ \propto \ & \exp\left[-0.5\left(B-B_0\right)'\Sigma_0^{-1}\left(B-B_0\right)\right] \end{aligned} \tag{2.6}$$

The equation in 2.6 simply defines a normal distribution with mean B_0 and variance Σ_0. Note that for practical purposes we only need to consider terms in the exponent (second line of equation 2.6) as the first two terms in 2.6 $\left((2\pi)^{-K/2}\left|\Sigma_0\right|^{-\frac{1}{2}}\right)$ are constants.

Setting up the likelihood function. In the second step, the researcher collects the data and forms the likelihood function:

$$\begin{aligned} F\left(Y_t|B,\sigma^2\right) &= \left(2\pi\sigma^2\right)^{-T/2} \exp\left(-\frac{\left(Y_t - BX_t\right)'\left(Y_t - BX_t\right)}{2\sigma^2}\right) \\ &\propto \exp\left(-\frac{\left(Y_t - BX_t\right)'\left(Y_t - BX_t\right)}{2\sigma^2}\right) \end{aligned} \tag{2.7}$$

As σ^2 is assumed to be known in this example, we can drop the first term in equation 2.7 $\left(2\pi\sigma^2\right)^{-T/2}$

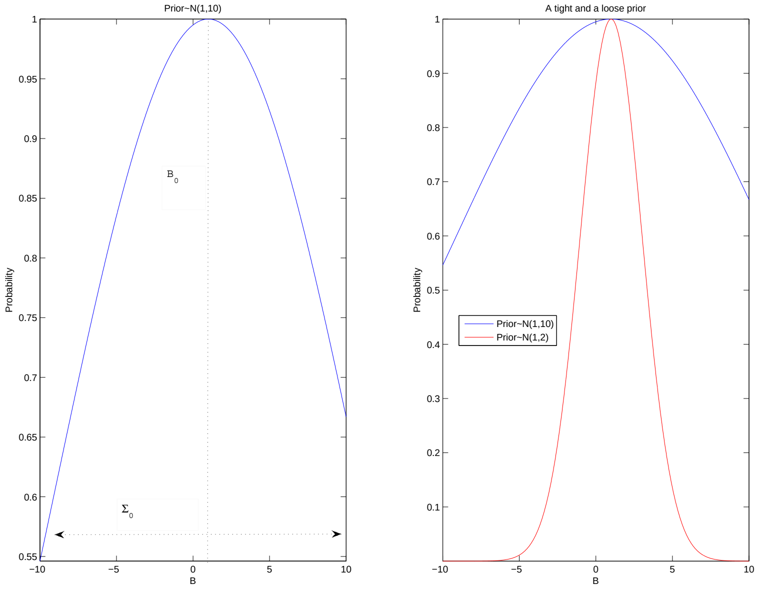


Figure 1. Loose and tight prior for the coefficients. An example

Calculating the posterior distribution. Recall from equation 2.5 that *the posterior distribution is proportional to the likelihood times the prior.* Therefore to find the posterior distribution for B (conditional on knowing σ^2) the researcher multiplies equation 2.6 and 2.7 to obtain

$$H\left(B|\sigma^2, Y_t\right) \propto \exp\left[-0.5\left(B - B_0\right)' \Sigma_0^{-1}\left(B - B_0\right)\right] \times \exp\left(-\frac{\left(Y_t - BX_t\right)'\left(Y_t - BX_t\right)}{2\sigma^2}\right) \tag{2.8}$$

Equation 2.8 is simply a product of two normal distributions and the result is also a normal distribution. Hence the posterior distribution of B conditional on σ^2 is given by:

$$H\left(B|\sigma^2, Y_t\right) \tilde{}N\left(M^*, V^*\right) \tag{2.9}$$

As shown in Hamilton (1994) pp 354 and Koop (2003) pp 61 the mean and the variance of this normal distribution are given by the following expressions

$$\begin{aligned} M^* &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^2}X_t'Y_t\right) \qquad (2.10) \\ V^* &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t'X_t\right)^{-1} \end{aligned}$$

Consider the expression for the mean of the conditional posterior distribution $M^* = \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^2}X_t'Y_t\right)$. Note that the final term $X_t'Y_t$ can be re-written as $X_t'X_tB_{ols}$ where $B_{ols} = \left(X_t'X_t\right)^{-1}X_t'Y_t$. That is

$$M^* = \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^2}X_t'X_tB_{ols}\right) \tag{2.11}$$

The second term of the expression in equation 2.11 shows that the mean of the conditional posterior distribution is weighted average of the prior mean B_0 and the maximum likelihood estimator B_{ols} with the weights given by the reciprocal of the variances of the two (in particular Σ_0^{-1} and $\frac{1}{\sigma^2}X_t'X_t$). A large number for Σ_0 would imply a very small weight on the prior and hence M^* would be dominated by the OLS estimate. A very small number for Σ_0, on the other hand, would imply that the conditional posterior mean is dominated by the prior. Note also that if the prior is removed from the expressions in equation 2.10 (i.e. if one removes B_0 and Σ_0^{-1} from the expressions) , one is left with the maximum likelihood estimates.

Example 2. *Figure 1 shows a simple example about a prior distribution for a regression model with* 1 *coefficient* B. *The X-axis of these figures show a range of values of* B. *The Y-axis plots the value of the normal prior distribution associated with these values of* B. *The left panel shows a a prior distribution with a mean of* 1 *and a variance of* 10. *As expected, the prior distribution is centered at* 1 *and the width of the distribution reflects the variance. The right panel compares this prior distribution with a tighter prior centered around the same mean. In particular, the new prior distribution (shown as the red line) has a variance of* 2 *and is much more tightly concentrated around the mean.*

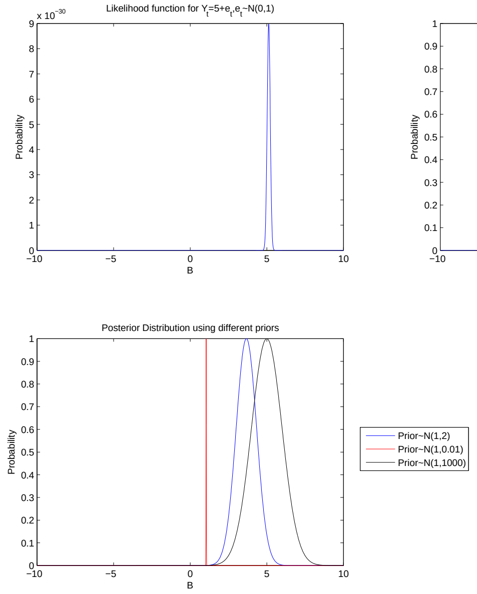


FIGURE 2. The posterior distribution for the model $Y_t = 5X_t + e_t, e_t\tilde{}N(0,1)$ using different priors

The tope left panel of figure 2 plots the likelihood function for the simple regression model $Y_t = BX_t + e_t, e_t\tilde{}N(0,1), B = 5$. *As expected the likelihood function has its peak at* $B = 5$. *The top right panel shows the posterior distribution which combines the prior distribution in figure 1 (*$N(1,2)$ *shown as the red line) with the likelihood function. Note that as the posterior combines the prior information (with a mean of* 1*) and the likelihood function, the posterior distribution is not exactly centered around* 5*, but around a value slightly less than* 5*, reflecting the influence of the prior. Note that if the prior is tightened significantly and its variance reduced to* 0.01*, this has the affect of shifting the posterior distribution with the mass concentrated around* 1 *(red line in the bottom left panel). In contrast, a loose prior with a prior variance of* 1000 *is concentrated around* 5.

2.2. Case 2: The posterior distribution of σ^2 assuming B is known. In the second example we consider the estimation of σ^2 in equation 2.1 assuming that the value of B is known. The derivation of the (conditional) posterior distribution of σ^2 proceeds in exactly the same three steps

Setting the prior. The normal distribution allows for negative numbers and is therefore not appropriate as a prior distribution for σ^2. A conjugate prior for σ^2 is the inverse Gamma distribution or *equivalently* a conjugate prior for $1/\sigma^2$ is the Gamma distribution.

DEFINITION 1. ***(Gamma Distribution)****: Suppose we have* T *iid numbers from the normal distribution* v_t

$$v_t\tilde{}N\left(0,\frac{1}{\theta}\right)$$

If we calculate the sum of squares of $W = \sum_{t=1}^{T} v_t^2$*, then* W *is distributed as a Gamma distribution with* T *degrees of freedom and a scale parameter* θ

$$W\tilde{}\Gamma\left(\frac{T}{2},\frac{\theta}{2}\right) \tag{2.12}$$

The probability density function for the Gamma distribution has a simple form and is given by

$$g(W) \propto W^{\frac{T}{2}-1}\exp\left(\frac{-W\theta}{2}\right) \tag{2.13}$$

where the mean of the distribution is defined as $E(W) = \frac{T}{\theta}$.

Setting the prior (continued). We set a Gamma prior for $1/\sigma^2$. That is $p\left(1/\sigma^2\right) \sim \Gamma\left(\frac{T_0}{2},\frac{\theta_0}{2}\right)$ where T_0 denotes the prior degrees of freedom and θ_0 denotes the prior scale parameter. As discussed below, the choice of T_0 and θ_0 affects the mean and the variance of the prior. The prior density, therefore, has the following form (see equation 2.13.)

$$\frac{1}{\sigma^2}^{\frac{T_0}{2}-1}\exp\left(\frac{-\theta_0}{2\sigma^2}\right) \tag{2.14}$$

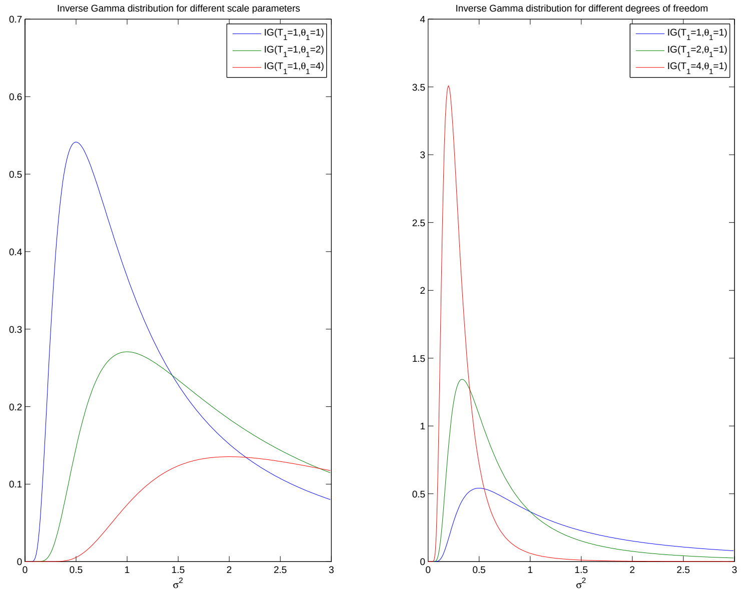


FIGURE 3. The inverse Gamma distribution for different degrees of freedom and scale parameters.

Setting up the likelihood function. In the second step, the researcher collects the data and forms the likelihood function:

$$\begin{aligned} F\left(Y_t|B,\sigma^2\right) &= \left(2\pi\sigma^2\right)^{-T/2}\exp\left(-\frac{\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)}{2\sigma^2}\right) \\ &\propto \left(\sigma^2\right)^{-T/2}\exp\left(-\frac{\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)}{2\sigma^2}\right) \end{aligned} \tag{2.15}$$

As σ^2 is assumed to be unknown in this example, we cannot drop the entire first term in equation 2.15.

Calculating the posterior distribution. To calculate the posterior distribution of $1/\sigma^2$ (conditional on B) we multiply the prior distribution in equation 2.14 and the likelihood function 2.15 to obtain

$$H\left(\frac{1}{\sigma^2}|B,Y_t\right)\propto {\frac{1}{\sigma^2}}^{\frac{T_0}{2}-1}\exp\left(\frac{-\theta_0}{2\sigma^2}\right)\times\sigma^{2-\frac{T}{2}}\exp\left(-\frac{1}{2\sigma^2}\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)\right)$$

$\rightarrow$

$${\frac{1}{\sigma^2}}^{\frac{T_0}{2}-1-\frac{T}{2}}\exp\left(-\frac{1}{2\sigma^2}\left[\theta_0+\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)\right]\right)$$

$\rightarrow$

$${\frac{1}{\sigma^2}}^{\frac{T_1}{2}-1}\exp\left(-\frac{\theta_1}{2\sigma^2}\right) \tag{2.16}$$

The resulting conditional posterior distribution for $1/\sigma^2$ in equation 2.16 can immediately be recognised as a Gamma distribution with degrees of freedom $T_1=\frac{T_0+T}{2}$ and $\theta_1=\frac{\theta_0+(Y_t-BX_t)'(Y_t-BX_t)}{2}$. Note that the conditional posterior distribution for σ^2 is inverse Gamma with degrees of freedom T_1 and scale parameter θ_1.

Consider the mean of the conditional posterior distribution (given by $\frac{T_1}{\theta_1}$)

$$\frac{T_0+T}{\theta_0+\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)} \tag{2.17}$$

It is interesting to note that without the prior parameters T_0 and θ_0, equation 2.17 simply defines the reciprocal of the maximum likelihood estimator of σ^2.

EXAMPLE 4. *The left panel of figure 3 plots the inverse Gamma distribution with the degrees of freedom held fixed at $T_1=1$, but for scale parameter $\theta_1=1,2,4$. Note that as the scale parameter increases, the distribution becomes skewed to the right and the mean increases. This suggests that an inverse Gamma prior with a larger scale parameter incorporates a prior belief of a larger value for σ^2. The right panel of the figure plots the inverse Gamma distribution for $\theta_1=1$, but for degrees of freedom $T_1=1,2,4$. As the degrees of freedom increase, the inverse Gamma distribution is more tightly centered around the mean. This suggests that a higher value for the degrees of freedom implies a tighter set of prior beliefs.*

2.3. Case 3: The posterior distribution of σ^2 and B. We now turn to the empirically relevant case when both the coefficient vector B and the variance $1/\sigma^2$ (in equation 2.1) is unknown. We proceed in exactly the same three steps

Setting the prior. We set the joint prior density for

$$p\left(B,\frac{1}{\sigma^2}\right)=P\left(\frac{1}{\sigma^2}\right)\times P\left(B|\frac{1}{\sigma^2}\right) \tag{2.18}$$

where $P\left(B|\frac{1}{\sigma^2}\right)\tilde{}N(B_0,\sigma^2\Sigma_0)$ and $P\left(\frac{1}{\sigma^2}\right)\tilde{}\Gamma\left(\frac{T_0}{2},\frac{\theta_0}{2}\right)$. That is: $P\left(\frac{1}{\sigma^2}\right)=\frac{1}{\sigma^2}^{\frac{T_0}{2}-1}\exp\left(\frac{-\theta_0}{2\sigma^2}\right)$ as in section 2.2 and $P\left(B|\frac{1}{\sigma^2}\right)=(2\pi)^{-K/2}\left|\sigma^2\Sigma_0\right|^{-\frac{1}{2}}\exp\left[-0.5\left(B-B_0\right)'\left(\sigma^2\Sigma_0\right)^{-1}\left(B-B_0\right)\right]$. Note that the prior for B is set conditional on σ^2. This prior is referred to as the natural conjugate prior for the linear regression model. A natural conjugate prior is a conjugate prior which has the same functional form as the likelihood.

Setting up the likelihood function. As above, the likelihood function is given by

$$F\left(Y_t|B,\sigma^2\right)=\left(2\pi\sigma^2\right)^{-T/2}\exp\left(-\frac{\left(Y_t-BX_t\right)'\left(Y_t-BX_t\right)}{2\sigma^2}\right) \tag{2.19}$$

Calculating the posterior distribution. The *joint* posterior distribution of B and the variance $1/\sigma^2$ is obtained by combining 2.18 and 2.19

$$H\left(\frac{1}{\sigma^2},B|Y_t\right)\propto p\left(B,\frac{1}{\sigma^2}\right)\times F\left(Y_t|B,\sigma^2\right) \tag{2.20}$$

Note that equation 2.20 is a *joint* posterior distribution involving $\frac{1}{\sigma^2}$ and B. Its form is more complicated than the conditional distributions for B and $\frac{1}{\sigma^2}$ shown in sections 2.1 and 2.2. To proceed further in terms of inference, the researcher has to 'isolate' the component of the posterior relevant to B or $\frac{1}{\sigma^2}$. For example, to conduct inference about B, the researcher has to derive the *marginal* posterior distribution for B. Similarly, inference on $\frac{1}{\sigma^2}$ is based on the marginal posterior distribution for $\frac{1}{\sigma^2}$. The marginal posterior for B is defined as

$$H\left(B|Y_t\right)=\int_0^\infty H\left(\frac{1}{\sigma^2},B|Y_t\right)d\frac{1}{\sigma^2} \tag{2.21}$$

while the marginal posterior for $\frac{1}{\sigma^2}$ is given by

$$H\left(\frac{1}{\sigma^2}|Y_t\right)=\int_0^\infty H\left(\frac{1}{\sigma^2},B|Y_t\right)dB \tag{2.22}$$

In the case of this simple linear regression model under *the natural conjugate prior*, analytical results for these integrals are available. As shown in Hamilton (1994) pp 357, the marginal posterior distribution for B is a multivariate T distribution, while the marginal posterior for $\frac{1}{\sigma^2}$ is a Gamma distribution. An intuitive description of these analytical results can also be found in Koop (2003) Chapter 2.

However, for the linear regression model with other prior distributions (for example where the prior for the coefficients is set independently from the prior for the variance) analytical derivation of the joint posterior and then the marginal posterior distribution is not possible. Similarly, in more complex models with a larger set of unknown parameters (i.e. models that may be more useful for inference and forecasting) these analytical results may be difficult to obtain. This may happen if the form of the joint posterior is unknown or is too complex for analytical integration.

Readers should pause at this point and reflect on two key messages from the three cases considered above:

EXAMPLE 3• As shown by Case 1 and Case 2, *conditional* posterior distributions are relatively easy to derive and work with.

- In contrast, as shown by Case 3, derivation of the marginal posterior distribution (from a joint posterior distribution) requires analytical integration which may prove difficult in complex models.

This need for analytical integration to calculate the marginal posterior distribution was the main stumbling block of Bayesian analysis making it difficult for applied researchers.

3. Gibbs Sampling for the linear regression model

It was the development of simulation method such as Gibbs sampling which greatly simplified the integration step discussed above and made it possible to easily extend Bayesian analysis to a variety of econometric models.

DEFINITION 2. *Gibbs sampling is a numerical method that uses draws from **conditional distributions** to approximate joint and marginal distributions.*

As discussed in case 3 above, researchers are interested in marginal posterior distributions which may be difficult to derive analytically. In contrast, the conditional posterior distribution of each set of parameters is readily available. According to definition 2, one can approximate the marginal posterior distribution by sampling from the conditional distributions.

We describe this algorithm in detail below, first in a general setting and then applied specifically to the linear regression model. *Most importantly, we then describe how to code the algorithm for linear regression models. Note that all the files referred to below are saved in the sub-folder called chapter 1 in the main folder called code.*

3.1. Gibbs Sampling a general description. Suppose we have a joint distribution of k variables

$$f(x_1, x_2...x_k) \tag{3.1}$$

This may, for example, be a joint posterior distribution.

and we are interested in obtaining the marginal distributions

$$f(x_i), i = 1...k \tag{3.2}$$

The standard way to do this is to integrate the joint distribution in 3.1. However, as discussed above, this integration may be difficult or infeasible in some cases. It may be that the exact form of 3.1 is unknown or is to complicated for direct analytical integration.

Assume that the form of the conditional distributions $f(x_i|x_j), i \neq j$ is known. A Gibbs sampling algorithm with the following steps can be used to approximate the marginal distributions.

(1) Set starting values for $x_1....x_k$

$$x_1^0, ...x_K^0$$

where the superscript 0 denotes the starting values.

(2) Sample x_1^1 from the distribution of x_1 conditional on current values of $x_2....x_k$

$$f\left(x_1^1|x_2^0, ..x_k^0\right)$$

(3) Sample x_2^1 from the distribution of x_2 conditional on current values of $x_1, x_3...x_k$

$$f\left(x_2^1|x_1^1, x_3^0..x_k^0\right)$$

.

.

.

k. Sample x_k^1 from the distribution of x_k conditional on current values of $x_1, x_2...x_{k-1}$

$$f\left(x_k^1|x_1^1, x_2^1..x_{k-1}^1\right)$$

to complete 1 *iteration of the Gibbs sampling algorithm.*

As the number of Gibbs iterations increases to infinity, the samples or draws from the conditional distributions converge to the joint and marginal distributions of x_i at an exponential rate (for a proof of convergence see Casella and George (1992)). Therefore after a large enough number of iterations, the marginal distributions can be approximated by the empirical distribution of x_i.

In other words, one repeats the Gibbs iterations M times (ie a number of iterations large enough for convergence) and saves the last H draws of x_i (for eg $H = 1000$). This implies that the researcher is left with H values for $x_1....x_k$. The histogram for $x_1....x_k$ (or any other estimate of the empirical density) is an approximation for the marginal density of $x_1....x_k$.

Thus an estimate of the mean of the marginal posterior distribution for x_i is simply the sample mean of the H retained draws

$$\frac{1}{H}\sum_{b=1}^{H} x_i^b$$

where the superscript b indexes the (retained) Gibbs iterations. Similarly, the estimate of the variance of the marginal posterior distribution is given by

How many Gibbs iterations are required for convergence? We will deal with this question in detail in section section 3.7 below.

One crucial thing to note is that the implementation of the Gibbs sampling algorithm requires the researcher to know the form of the conditional distributions $f(x_i|x_j)$. In addition, it must be possible to take random draws from these conditional distributions.

3.2. Gibbs Sampling for a linear regression. We now proceed to our first practical example involving a linear regression model. We first describe the application of the Gibbs sampling algorithm to the regression. This is followed immediately by a line by line description of Matlab code needed to implement the algorithm.

Consider the estimation of the following AR(2) model via Gibbs sampling

$$Y_t = \alpha + B_1 Y_{t-1} + B_2 Y_{t-2} + v_t, v_t\tilde{} N(0, \sigma^2) \tag{3.3}$$

where Y_t is annual CPI inflation for the US over the period 1948Q1 to 2010Q3. Let $X_t = \{1, Y_{t-1,1} Y_{t-2}\}$ denote the RHS variables in equation 3.3 and $B = \{\alpha, B_1, B_2\}$ the coefficient vector. Our aim is to approximate the marginal posterior distribution of α, B_1, B_2 and σ^2. As discussed above it is difficult to derive these marginal distributions analytically. Note, however, that we readily derived the posterior distribution of $B = \{\alpha, B_1, B_2\}$ conditional on σ^2

(see section 2.1) and the posterior distribution of σ^2 conditional on $B = \{\alpha, B_1, B_2\}$ (see section 2.2) Estimation of this model proceeds in the following steps

Step 1 Set priors and starting values. We set a normal prior for the coefficients B.

$$p(B)\tilde{}N\left(\underbrace{\begin{pmatrix}\alpha^0\\ B_1^0\\ B_2^0\end{pmatrix}}_{B_0},\underbrace{\begin{pmatrix}\Sigma_\alpha & 0 & 0\\ 0 & \Sigma_{B1} & 0\\ 0 & 0 & \Sigma_{B2}\end{pmatrix}}_{\Sigma_0}\right) \tag{3.4}$$

In other words, we specify the prior means for each coefficient in B (denoted as B_0 in 3.4) and the prior variance Σ_0. For this example (with three coefficients) B_0 is a 3×1 vector, while Σ_0 is 3×3 matrix with each diagonal element specifying the prior variance of the corresponding element of B_0.

We set an inverse Gamma prior for σ^2 and set the prior degrees of freedom T_0 and the prior scale matrix θ_0 (see equation 3.5). We will therefore work with the inverse Gamma distribution in the Gibbs sampler below. Note that this is equivalent to working with Gamma distribution and $1/\sigma^2$.

$$p\left(\sigma^2\right)\tilde{}\Gamma^{-1}\left(\frac{T_0}{2},\frac{\theta_0}{2}\right) \tag{3.5}$$

To initialise the Gibbs sampler we need a starting value for either σ^2 or B. In this example we will assume that the starting value for $\sigma^2 = \sigma^2_{OLS}$ where σ^2_{OLS} is the OLS estimate of σ^2. In linear models (such as linear regressions and Vector Autoregressions) the choice of starting values has, in our experience, little impact on the final results given that the number of Gibbs iterations is large enough.

Step 2 Given a value for σ^2 we sample from the conditional posterior distribution of B. As discussed in section 2.1, this is a normal distribution with a known mean and variance given

$$H\left(B|\sigma^2, Y_t\right)\tilde{}N\left(M^*, V^*\right) \tag{3.6}$$

where

$$\begin{aligned}\underset{(3\times1)}{M^*} &= \left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0+\frac{1}{\sigma^2}X_t'Y_t\right) \\ \underset{(3\times3)}{V^*} &= \left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}\end{aligned} \tag{3.7}$$

Note that we have all the ingredients to calculate M^* and V^* which in this example are 3×1 and 3×3 matrices respectively. We now need a sample from the normal distribution with mean M^* and variance V^*. For this we can use the following algorithm.

Algorithm 1. *To sample a $k \times 1$vector denoted by z from the $N(m, v)$ distribution, first generate $k \times 1$ numbers from the standard normal distribution (call these z^0. Note that all computer packages will provide a routine to do this). These standard normal numbers can then be transformed such that the mean is equal to m and variance equals v using the following transformation*

$$z = m + z^0 \times v^{1/2}$$

Thus one adds the mean and multiplies z^0 by the square root of the variance.

Step 2 (continued) The procedure in algorithm 1 suggests that once we have calculated M^* and V^*, the draw for B is obtained as

$$\underset{(3\times1)}{B^1} = \underset{(3\times1)}{M^*} + \left[\underset{(1\times3)}{\bar{B}} \times \underset{(3\times3)}{(V^*)^{1/2}}\right]' \tag{3.8}$$

where $\bar{B}$ is a 1×3 vector from the standard normal distribution. Note that the superscript 1 in B^1 denotes the first Gibbs iteration.

Step 3 Given the draw B^1, we draw σ^2 form its conditional posterior distribution. As shown in section 2.2 the conditional posterior distribution for σ^2 is inverse Gamma

$$H\left(\sigma^2|B, Y_t\right)\tilde{}\Gamma^{-1}\left(\frac{T_1}{2},\frac{\theta_1}{2}\right) \tag{3.9}$$

where

$$\begin{aligned}T_1 &= T_0 + T \\ \theta_1 &= \theta_0 + \left(Y_t - B^1X_t\right)'\left(Y_t - B^1X_t\right)\end{aligned} \tag{3.10}$$

A crucial thing to note about the posterior scale parameter of this distribution θ_1 is the fact that the second term $(\left(Y_t - B^1X_t\right)'\left(Y_t - B^1X_t\right))$ is calculated using the previous draw of the coefficient vector (in this case B^1). To draw from the inverse Gamma distribution in equation 3.9 we first calculate the parameters in equation 3.10 and then use the following algorithm to draw $\left(\sigma^2\right)^1$ from the inverse Gamma distribution (note that $\left(\sigma^2\right)^i$ denotes the *ith* Gibbs draw) .

Algorithm 2. *To sample a scalar z from the Inverse Gamma distribution with degrees of freedom $\frac{T}{2}$ and scale parameter $\frac{D}{2}$ i.e. $\Gamma^{-1}(\frac{T}{2}, \frac{D}{2})$: Generate T numbers form the standard normal distribution $z^0\tilde{}N(0,1)$. Then*

$$z = \frac{D}{z^{0\prime}z^0}$$

is a draw from the $\Gamma^{-1}(\frac{T}{2}, \frac{D}{2})$ distribution.

Step 4 Repeat steps 2 and 3 M times to obtain $B^1...B^M$ and $\left(\sigma^2\right)^1 .... \left(\sigma^2\right)^M$. The last H values of B and σ^2 from these iterations is used to form the empirical distribution of these parameters. Note that this empirical distribution is an approximation to the marginal posterior distribution. Note also that the first $M - H$ iterations which are discarded are referred to as burn-in iterations. These are the number of iterations required for the Gibbs sampler to converge.

Its worth noting that it makes no difference which order steps 2 and 3 are repeated. For example one could start the Gibbs sampler by drawing σ^2 conditional on starting values for B (rather than the other way around as we have done here)

3.2.1. *Inference using output from the Gibbs sampler.* The Gibbs sampler applied to the linear regression model produces a sequence of draws from the approximate marginal posterior distribution of B and σ^2. The mean of these draws is an approximation to the posterior mean and provides a point estimate of of B and σ^2. The percentiles calculated from these draws can be used to produce posterior density intervals. For example, the 5^{th} and the 95^{th} percentiles approximate the 10% highest posterior density intervals (HPDI) or 10% credible sets which can be used for simple hypothesis testing. For example, if the highest posterior density interval for B does not contain zero, this is evidence that the hypothesis that $B = 0$ can be rejected.

More formal methods for model comparison involve the marginal likelihood $F(Y)$ mentioned in section 2. The marginal likelihood is defined as

$$F(Y) = \int F\left(Y|B,\sigma^2\right)p\left(B,\sigma^2\right)d\Xi$$

where $\Xi = B, \sigma^2$. In other words, the marginal likelihood represents the posterior distribution with the parameters integrated out. Consider two models M_1 and M_2. Model M_1 is preferred if $F_{M_1}(Y) > F_{M_2}(Y)$ or the Bayes factor $\frac{F_{M_1}(Y)}{F_{M_2}(Y)}$ is larger than 1. In comparison to HPDIs, inference based on marginal likelihoods or Bayes factors is more complicated from a computational and statistical point of view. First, while an analytical expression for $F(Y)$ is available for the linear regression model under the natural conjugate prior, numerical methods are generally required to calculate the integral in the expression for $F(Y)$ above. In the appendix to this chapter, we provide an example of how Gibbs sampling can be used to compute the marginal likelihood for the linear regression model. Second, model comparison using marginal likelihoods requires the researchers to use proper priors (i.e. prior distributions that integrate to 1). In addition, using non-informative priors may lead to problems when interpreting Bayes Factors. An excellent description of these issues can be found in Koop (2003) pp 38.

3.3. Gibbs Sampling for a linear regression in Matlab (example1.m). We now go through the Matlab code needed for implementing the algorithm described in the section above. Note that the aim is to estimate the following AR(2) model via Gibbs sampling.

$$Y_t = \alpha + B_1 Y_{t-1} + B_2 Y_{t-2} + v_t, v_t\tilde{}N(0,\sigma^2) \tag{3.11}$$

where Y_t is annual CPI inflation for the US over the period 1948Q1 to 2010Q3 and $B = \{\alpha, B_1, B_2\}$. The code presented below is marked with comments for convenience. The same code without comments accompanies this monograph and is arranged by chapter. The code for the example we consider in this section is called example1.m and is saved in the folder

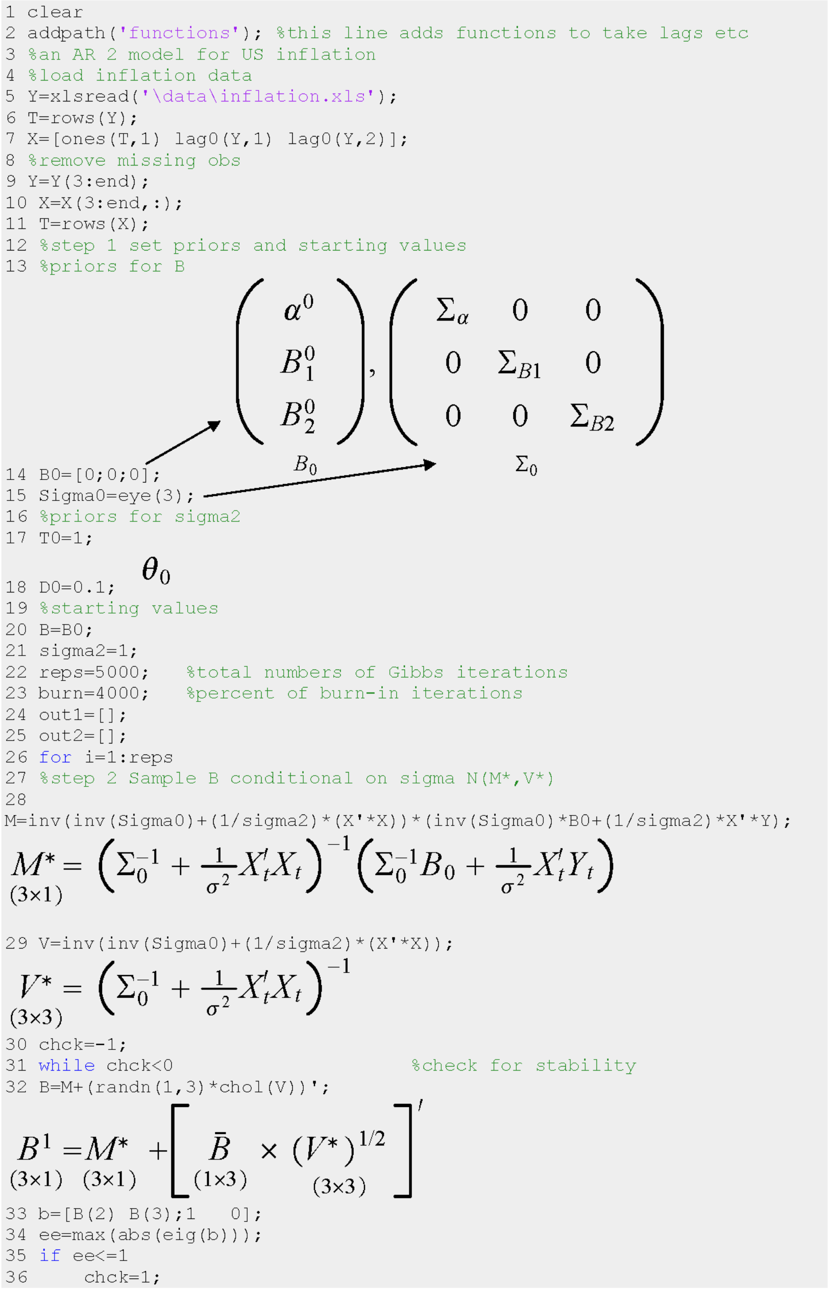
Consider the code for this example presented in 4 and 5. Line 2 of the code adds functions that are needed as utilities–eg for taking lags or differences. We will not discuss these further. On line 5, we load data for US inflation from an excel file. Line 7 creates the regressors, a constant and two lags of inflation (using the function lag0 in the folder functions). Line 11 specifies the total number of time series observations after removing the missing values generated after taking lags. Line 14 sets prior mean for the regression coefficients.

$$\begin{pmatrix} \alpha^0 \\ B_1^0 \\ B_2^0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

The prior mean for each coefficient is set to zero in this example. The prior variance is set to an identity matrix on line 15 in this example.

$$\begin{pmatrix} \Sigma_\alpha & 0 & 0 \\ 0 & \Sigma_{B1} & 0 \\ 0 & 0 & \Sigma_{B2} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Line 17 sets the prior degrees of freedom for the inverse Gamma distribution while line sets θ_0 the prior scale parameter. Line 20 sets the starting value of B, while line 21 sets the starting value for σ^2. Line 22 specifies the total



```
1 clear
2 addpath('functions'); %this line adds functions to take lags etc
3 %an AR 2 model for US inflation
4 %load inflation data
5 Y=xlsread('\data\inflation.xls');
6 T=rows(Y);
7 X=[ones(T,1) lag0(Y,1) lag0(Y,2)];
8 %remove missing obs
9 Y=Y(3:end);
10 X=X(3:end,:);
11 T=rows(X);
12 %step 1 set priors and starting values
13 %priors for B
```

$$\underbrace{\begin{pmatrix}\alpha^0\\ B_1^0\\ B_2^0\end{pmatrix}}_{B_0},\underbrace{\begin{pmatrix}\Sigma_\alpha & 0 & 0\\ 0 & \Sigma_{B1} & 0\\ 0 & 0 & \Sigma_{B2}\end{pmatrix}}_{\Sigma_0}$$

```
14 B0=[0;0;0];
15 Sigma0=eye(3);
16 %priors for sigma2
17 T0=1;
```

θ_0

```
18 D0=0.1;
19 %starting values
20 B=B0;
21 sigma2=1;
22 reps=5000;   %total numbers of Gibbs iterations
23 burn=4000;   %percent of burn-in iterations
24 out1=[];
25 out2=[];
26 for i=1:reps
27 %step 2 Sample B conditional on sigma N(M*,V*)
28
M=inv(inv(Sigma0)+(1/sigma2)*(X'*X))*(inv(Sigma0)*B0+(1/sigma2)*X'*Y);
```

$$\underset{(3\times1)}{M^*}=\left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0+\frac{1}{\sigma^2}X_t'Y_t\right)$$

```
29 V=inv(inv(Sigma0)+(1/sigma2)*(X'*X));
```

$$\underset{(3\times3)}{V^*}=\left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}$$

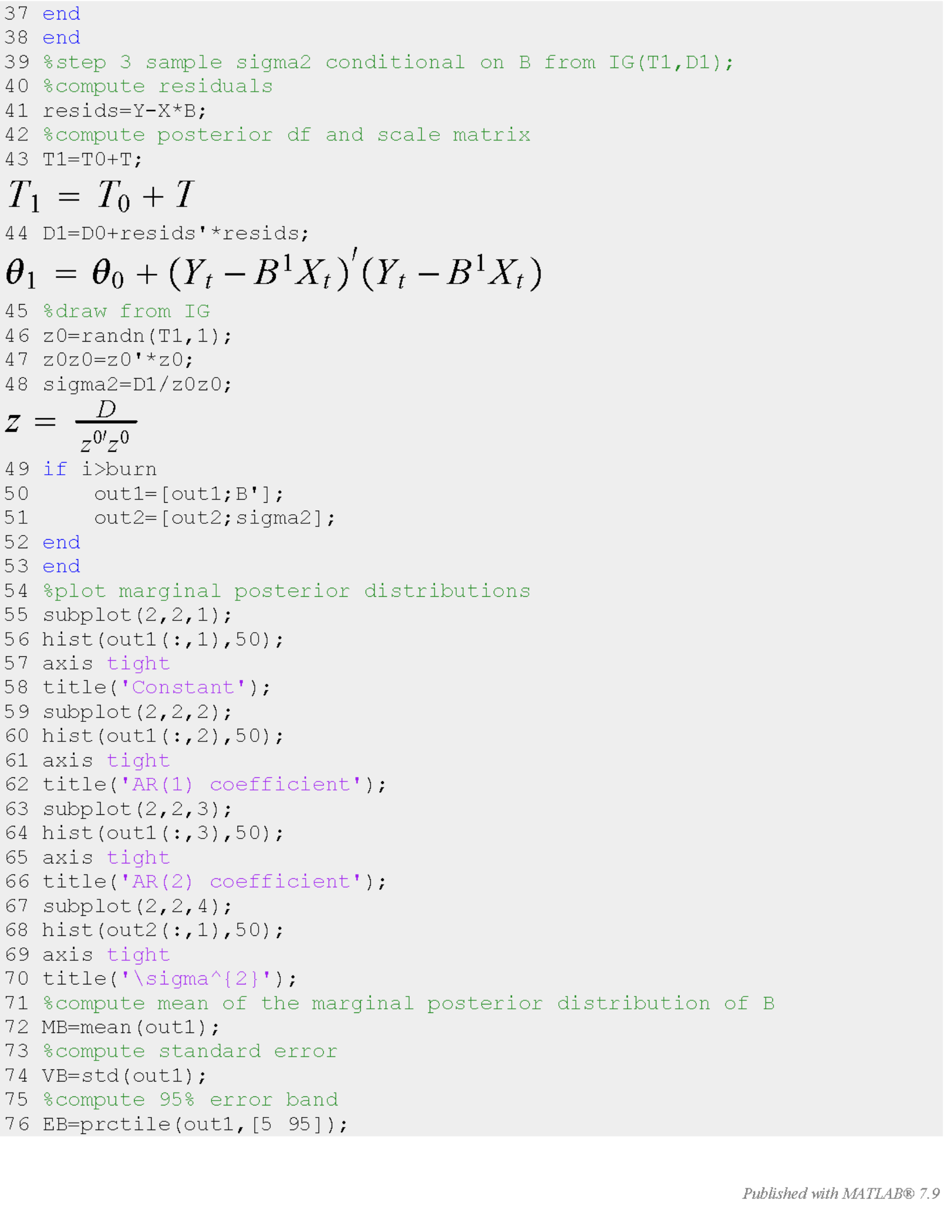
```
30 chck=-1;
31 while chck<0                    %check for stability
32 B=M+(randn(1,3)*chol(V))';
```

$$\underset{(3\times1)}{B^1}=\underset{(3\times1)}{M^*}+\left[\underset{(1\times3)}{\bar{B}}\times\underset{(3\times3)}{(V^*)^{1/2}}\right]'$$

```
33 b=[B(2) B(3);1   0];
34 ee=max(abs(eig(b)));
35 if ee<=1
36     chck=1;
```

FIGURE 4. Example 1: Matlab code

number of Gibbs iterations, while line 23 specifies the number to discard (in this example we save 1000 iterations for inference). out1 and out2 on line 24 and 25 are empty matrices that will save the draws of B and σ^2 respectively. Line 26 starts the main loop that carries out the Gibbs iterations. On line 28, we begin the first step of the Gibbs algorithm and calculate the mean of the conditional posterior distribution of B $\left(M^*=\left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0+\frac{1}{\sigma^2}X_t'Y_t\right)\right)$ and on line 29 we calculate the variance of this conditional posterior distribution. Line 32 draws from the normal distribution with this mean and variance. Note that it is standard to restrict the draw of the AR coefficients to be stable. This is why line 31 has a while loop which keeps on drawing from the coefficients from the normal distribution if the draws are unstable. Stability is checked on line 33 by computing the eigenvalues of the coefficient matrix written



```
end
end
%step 3 sample sigma2 conditional on B from IG(T1,D1);
%compute residuals
resids=Y-X*B;
%compute posterior df and scale matrix
T1=T0+T;
```

$$T_1 = T_0 + T$$

```
D1=D0+resids'*resids;
```

$$\theta_1 = \theta_0 + (Y_t - B^1 X_t)'(Y_t - B^1 X_t)$$

```
%draw from IG
z0=randn(T1,1);
z0z0=z0'*z0;
sigma2=D1/z0z0;
```

$$z = \frac{D}{z0'z0}$$

```
if i>burn
    out1=[out1;B'];
    out2=[out2;sigma2];
end
end
%plot marginal posterior distributions
subplot(2,2,1);
hist(out1(:,1),50);
axis tight
title('Constant');
subplot(2,2,2);
hist(out1(:,2),50);
axis tight
title('AR(1) coefficient');
subplot(2,2,3);
hist(out1(:,3),50);
axis tight
title('AR(2) coefficient');
subplot(2,2,4);
hist(out2(:,1),50);
axis tight
title('\sigma^{2}');
%compute mean of the marginal posterior distribution of B
MB=mean(out1);
%compute standard error
VB=std(out1);
%compute 95% error band
EB=prctile(out1,[5 95]);
```

Published with MATLAB® 7.9

FIGURE 5. Example 1: Matlab Code (continued)

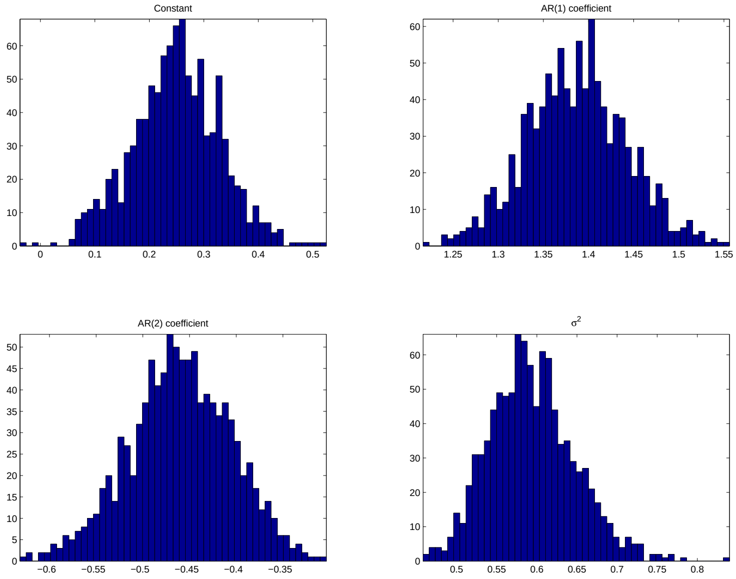


FIGURE 6. Results using example1.m

	Posterior Mean	Standard Deviation	5th and 95th percentiles
α	0.2494	0.0799	$(0.1104, 0.3765)$
B_1	1.3867	0.0557	$(1.2922, 1.4806)$
B_2	-0.4600	0.0550	$(-0.5532, -0.3709)$

TABLE 1. Results using example1.m

in first order companion form. That is the AR(2) model is re-written as (this is the companion form)

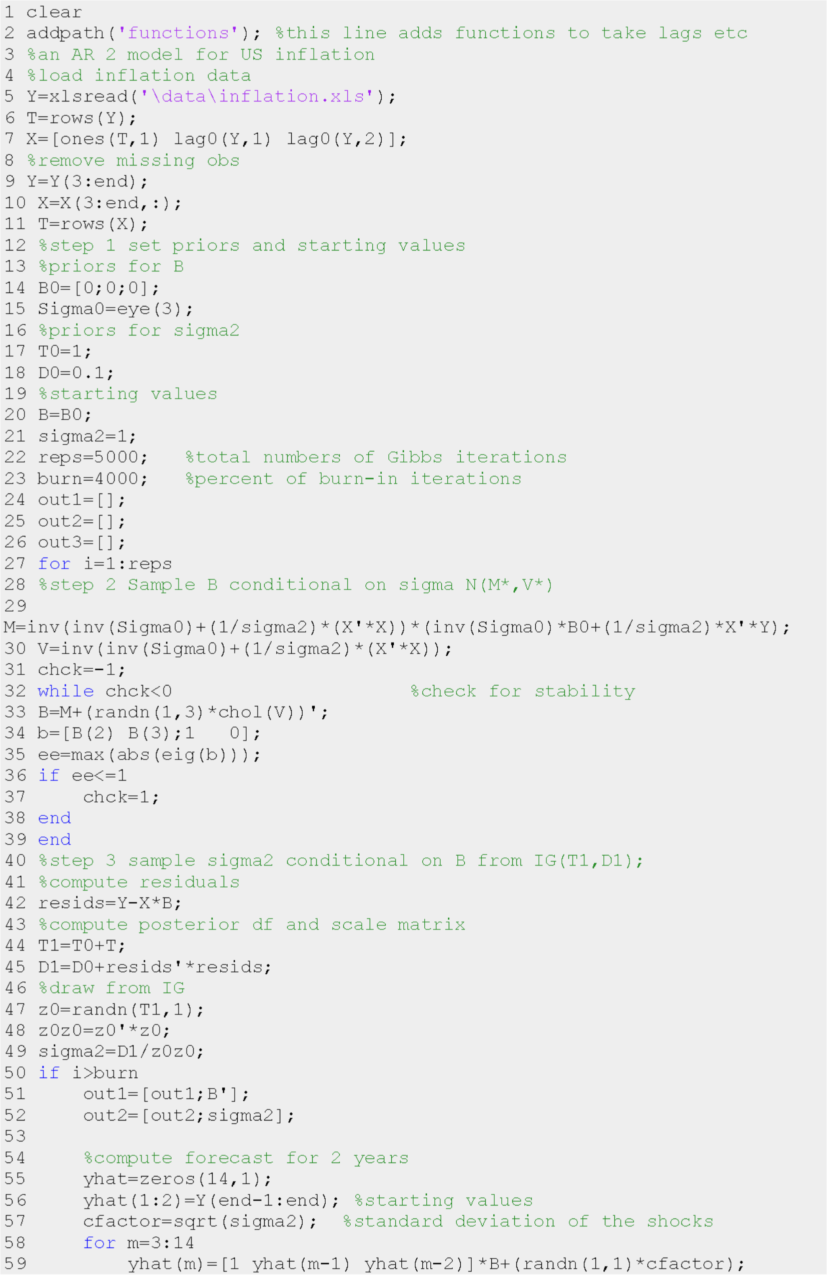
$$\begin{pmatrix} Y_t \\ Y_{t-1} \end{pmatrix} = \begin{pmatrix} \alpha \\ 0 \end{pmatrix} + \begin{pmatrix} B_1 & B_2 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} Y_{t-1} \\ Y_{t-2} \end{pmatrix} + \begin{pmatrix} v_t \\ 0 \end{pmatrix}$$

Then the AR model is stable if the eigenvalues of $\begin{pmatrix} B_1 & B_2 \\ 1 & 0 \end{pmatrix}$ are less than or equal to 1 in absolute value. Note that this check for stability is not required for the Gibbs sampling algorithm but usually added by researchers for practical convenience. Line 41 computes the residuals using the last draw of the coefficients. Line 43 computes the posterior degrees of freedom for the inverse Gamma distribution $T_1 = T_0 + T$. Line 44 computes the posterior scale parameter $\theta_1 = \theta_0 + \left(Y_t - B^1 X_t\right)' \left(Y_t - B^1 X_t\right)$. Line 46 to 48 draw from the inverse Gamma distribution using algorithm 2. Lines 49 to 51 save the draws of B and σ^2 once the number of iterations exceed the burn-in period. Running this file produces the histograms shown in figure 6 (see lines 54 to 70 in example1.m–these histograms are drawn using the retained draws in out1 and out2). These histograms are the Gibbs sampling estimate of the marginal posterior distribution of the coefficients and the variance. Note that the mean of the posterior distribution is easily calculated as the sample mean of these saved draws. Similarly, the sample standard deviation and percentiles provide measures of uncertainty. Researchers usually report the posterior mean, the standard deviation and the 5th and 95th percentiles of the posterior distribution. Example1.m produces the following moments for the coefficients (see table 1).

Note that the percentiles of the distribution are a useful measure of uncertainty. These represent HPDIs, or the posterior belief that the parameter lies within a range (see Canova (2007) page 337 and Koop (2003) pp 43). Suppose that the lower bound for α was less than 0. Then this would indicate that one cannot exclude the possibility that the posterior mean for α is equal to zero.

3.4. Gibbs Sampling for a linear regression in Matlab and forecasting (example2.m). The file example2.m considers the same model as in the previous subsection. However, we know use the AR model to forecast inflation and build the distribution of the forecast. This example shows that one can easily obtain the distribution of functions of the regression coefficients. Note that the forecast from an AR(2) model is easily obtained via simulation. In other words, given a value for the current and lagged data and the regression coefficients, the 1 period ahead forecast is

$$\hat{Y}_{t+1} = \alpha + B_1 Y_t + B_2 Y_{t-1} + (\sigma v^*) \tag{3.12}$$



```
1 clear
2 addpath('functions'); %this line adds functions to take lags etc
3 %an AR 2 model for US inflation
4 %load inflation data
5 Y=xlsread('\data\inflation.xls');
6 T=rows(Y);
7 X=[ones(T,1) lag0(Y,1) lag0(Y,2)];
8 %remove missing obs
9 Y=Y(3:end);
10 X=X(3:end,:);
11 T=rows(X);
12 %step 1 set priors and starting values
13 %priors for B
14 B0=[0;0;0];
15 Sigma0=eye(3);
16 %priors for sigma2
17 T0=1;
18 D0=0.1;
19 %starting values
20 B=B0;
21 sigma2=1;
22 reps=5000;   %total numbers of Gibbs iterations
23 burn=4000;   %percent of burn-in iterations
24 out1=[];
25 out2=[];
26 out3=[];
27 for i=1:reps
28 %step 2 Sample B conditional on sigma N(M*,V*)
29
M=inv(inv(Sigma0)+(1/sigma2)*(X'*X))*(inv(Sigma0)*B0+(1/sigma2)*X'*Y);
30 V=inv(inv(Sigma0)+(1/sigma2)*(X'*X));
31 chck=-1;
32 while chck<0                    %check for stability
33 B=M+(randn(1,3)*chol(V))';
34 b=[B(2) B(3);1   0];
35 ee=max(abs(eig(b)));
36 if ee<=1
37     chck=1;
38 end
39 end
40 %step 3 sample sigma2 conditional on B from IG(T1,D1);
41 %compute residuals
42 resids=Y-X*B;
43 %compute posterior df and scale matrix
44 T1=T0+T;
45 D1=D0+resids'*resids;
46 %draw from IG
47 z0=randn(T1,1);
48 z0z0=z0'*z0;
49 sigma2=D1/z0z0;
50 if i>burn
51     out1=[out1;B'];
52     out2=[out2;sigma2];
53
54     %compute forecast for 2 years
55     yhat=zeros(14,1);
56     yhat(1:2)=Y(end-1:end); %starting values
57     cfactor=sqrt(sigma2);  %standard deviation of the shocks
58     for m=3:14
59         yhat(m)=[1 yhat(m-1) yhat(m-2)]*B+(randn(1,1)*cfactor);
```

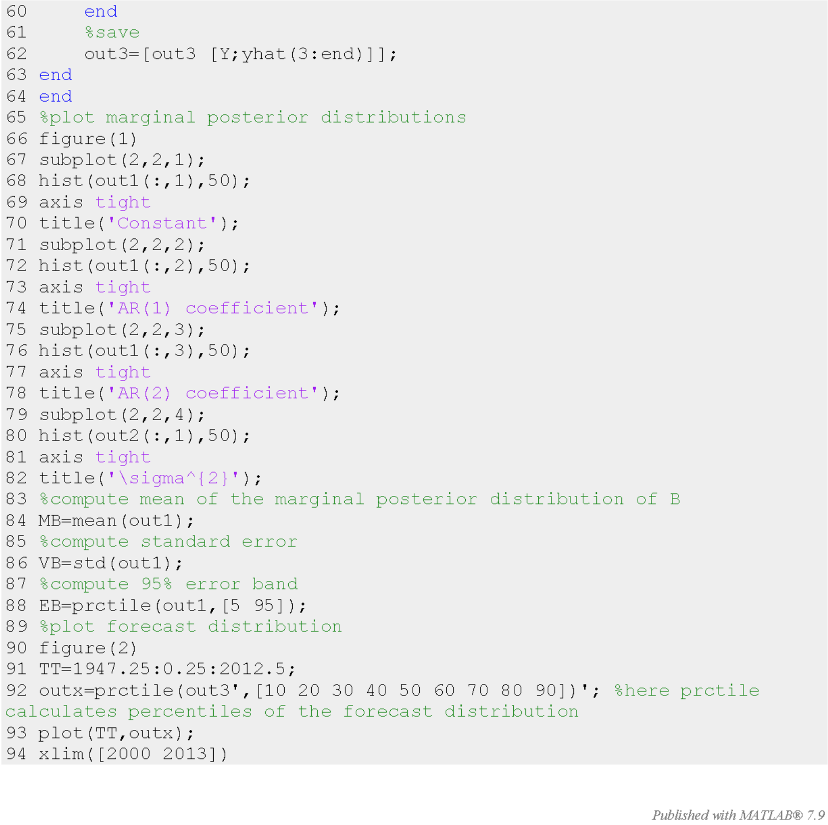
Figure 7. Example 2: Matlab code

where v^* is a scalar drawn from the standard normal distribution. Similarly, the 2 period ahead forecast is

$$\hat{Y}_{t+2} = \alpha + B_1\hat{Y}_{t+1} + B_2Y_t + (\sigma v^*) \tag{3.13}$$

and so forth. Note that we incorporate future shock uncertainty by adding the term σv^* i.e. a draw from the normal distribution with mean 0 and variance σ^2.

The code shown in figures 7 and 8 is identical to example 1 until line 54. Once past the burn in stage, we not only save the draws from the conditional distributions of the coefficients and the variance, but we use these draws to compute a two year ahead forecast for inflation. Line 55 intialises an empty matrix yhat which will save the forecast.



```
60     end
61     %save
62     out3=[out3 [Y;yhat(3:end)]];
63 end
64 end
65 %plot marginal posterior distributions
66 figure(1)
67 subplot(2,2,1);
68 hist(out1(:,1),50);
69 axis tight
70 title('Constant');
71 subplot(2,2,2);
72 hist(out1(:,2),50);
73 axis tight
74 title('AR(1) coefficient');
75 subplot(2,2,3);
76 hist(out1(:,3),50);
77 axis tight
78 title('AR(2) coefficient');
79 subplot(2,2,4);
80 hist(out2(:,1),50);
81 axis tight
82 title('\sigma^{2}');
83 %compute mean of the marginal posterior distribution of B
84 MB=mean(out1);
85 %compute standard error
86 VB=std(out1);
87 %compute 95% error band
88 EB=prctile(out1,[5 95]);
89 %plot forecast distribution
90 figure(2)
91 TT=1947.25:0.25:2012.5;
92 outx=prctile(out3',[10 20 30 40 50 60 70 80 90])'; %here prctile
calculates percentiles of the forecast distribution
93 plot(TT,outx);
94 xlim([2000 2013])
```

Published with MATLAB® 7.9

FIGURE 8. Example 2: Matlab code (continued)

Line 56 fills the first two values of yhat as actual values of inflation in the last two periods of the sample. Line 58 to 60 carries out the recursion shown in equations 3.12 and 3.13 for 12 periods. Line 62 saves actual inflation and the forecast in a matrix out3. The crucial thing to note is that this done for each Gibbs iteration after the burn-in period. Therefore in the end we have a set of 1000 forecasts. This represents an estimate of the posterior density. On line 92 we calculate the percentiles of the 1000 forecasts. The result gives us a fan chart for the inflation forecast shown in figure 9.

3.5. Gibbs Sampling for a linear regression with serial correlation. We now proceed to our second main example involving the linear regression model. We illustrate the power of the Gibbs sampler by considering

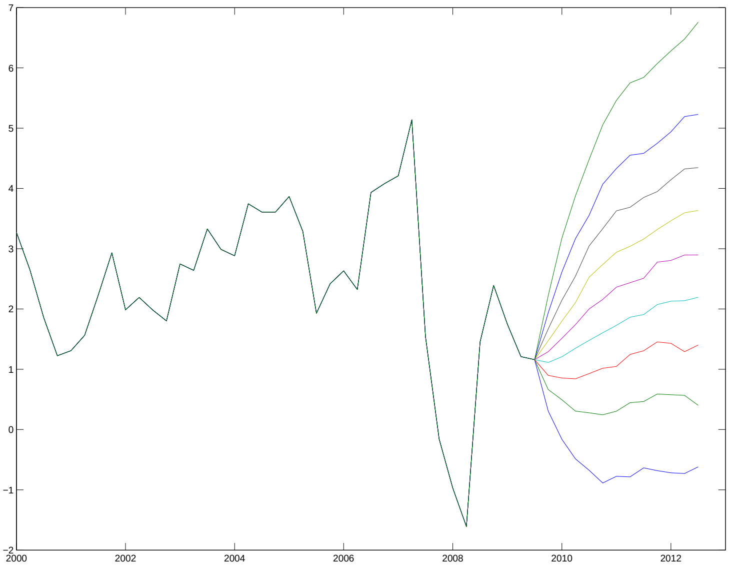


FIGURE 9. The distribution of the forecast of inflation using example2.m

the model in 3.3 but allowing for first order serial correlation in the residuals. We first describe the application of the Gibbs sampling algorithm to the regression. This is followed immediately by a line by line description of Matlab code needed to implement the algorithm. This algorithm was first developed in Chib (1993).

Consider the estimation of the following AR(2) model via Gibbs sampling

$$\begin{aligned} Y_t &= \alpha + B_1 Y_{t-1} + B_2 Y_{t-2} + v_t \\ v_t &= \rho v_{t-1} + \varepsilon_t, \varepsilon_t \sim N(0, \sigma^2) \end{aligned} \tag{3.14}$$

where Y_t is annual CPI inflation for the US over the period 1948Q1 to 2010Q3. Let $X_t = \{1, Y_{t-1},_1 Y_{t-2}\}$ denote the RHS variables in equation 3.3 and $B = \{\alpha, B_1, B_2\}$ the coefficient vector. Our aim is to approximate the marginal posterior distribution of α, B_1, B_2 and $\sigma^2\ and\ \rho.$

The key to seeting up the Gibbs sampler for this model is to make the following two observations

- Suppose we knew the value of ρ. Then the model in equation 3.14 can be transformed to remove the serial correlation. In particular we can re-write the model as

$$\underbrace{(Y_t - \rho Y_{t-1})}_{Y_t^*} = \alpha\,(1-\rho) + B_1 \underbrace{(Y_{t-1} - \rho Y_{t-2})}_{Y_{t-1}^*} + B_2 \underbrace{(Y_{t-2} - \rho Y_{t-3})}_{Y_{t-2}^*} + \underbrace{(v_t - \rho v_{t-1})}_{\varepsilon_t} \tag{3.15}$$

  That is we subtract the lag of each variable times the serial correlation coefficient ρ. Note that the transformed error term $v_t - \rho v_{t-1}$ is serially uncorrelated. Therefore after this transformation we are back to the linear regression framework we saw in the first example (see section 3.2). In other words, after removing the serial correlation, the conditional distribution of the coefficients and of the error variance is exactly as described for the standard linear regression model in section 3.2.
- Suppose we know α, B_1 and B_2. Then we can compute $v_t = Y_t - (\alpha + B_1 Y_{t-1} + B_2 Y_{t-2})$ and treat the equation $v_t = \rho v_{t-1} + \varepsilon_t, \varepsilon_t \sim N(0, \sigma^2)$ as a linear regression model in v_t. Again, this is just a standard linear regression model with an iid error term and the standard formulas for the conditional distribution of the regression coefficient ρ and the error variance σ^2 applies.

These two observations clearly suggest that to estimate this model, the Gibbs sampler needs three steps (instead of two in the previous example). We draw α, B_1 and B_2 conditional on knowing σ^2 and ρ after transforming the model to remove serial correlation (as in equation 3.15). Conditional on α, B_1 and B_2 and σ^2 we draw ρ. Finally, conditional on α, B_1,B_2 and ρ we draw σ^2. The steps are as follows

Step 1 Set priors and starting values. We set a normal prior for the coefficients B.

$$p(B)\tilde{}N\left(\underbrace{\begin{pmatrix} \alpha^0 \\ B_1^0 \\ B_2^0 \end{pmatrix}}_{B_0}, \underbrace{\begin{pmatrix} \Sigma_\alpha & 0 & 0 \\ 0 & \Sigma_{B1} & 0 \\ 0 & 0 & \Sigma_{B2} \end{pmatrix}}_{\Sigma_0}\right) \tag{3.16}$$

In other words, we specify the prior means for each coefficient in B (denoted as B_0 in 3.4) and the prior variance Σ_0. For this example (with three coefficients) B_0 is a 3×1 vector, while Σ_0 is 3×3 matrix with each diagonal element specifying the prior variance of the corresponding element of B_0.

We set a normal prior for the serial correlation coefficient ρ

$$p(\rho)\tilde{}N\left(\rho^0, \Sigma_\rho\right) \tag{3.17}$$

We set an inverse Gamma prior for σ^2 and set the prior degrees of freedom T_0 and the prior scale matrix θ_0 (see equation 3.18).

$$p\left(\sigma^2\right)\tilde{}\Gamma^{-1}\left(\frac{T_0}{2}, \frac{\theta_0}{2}\right) \tag{3.18}$$

To initialise the Gibbs sampler we need a starting value for σ^2 and ρ. In this example we will assume that the starting value for $\sigma^2 = \sigma^2_{OLS}$ where σ^2_{OLS} is the OLS estimate of σ^2. We assume that the starting value for $\rho = 0$.

Step 2 Given a value for σ^2 and ρ we sample from the conditional posterior distribution of B. As discussed above, this is done by first transforming the dependent and independent variables in the model to remove serial correlation. Once this is done we are back to the standard linear regression framework. We create the following transformed variables

$$\begin{aligned} Y_t^* &= Y_t - \rho Y_{t-1} \\ X_t^* &= X_t - \rho X_{t-1} \end{aligned}$$

where X_t^* represent the right hand side variables in our AR model. The conditional distribution of the regression coefficients is then given as

$$H\left(B|\sigma^2, \rho, Y_t\right)\tilde{}N\left(M^*, V^*\right) \tag{3.19}$$

where

$$\begin{aligned} \underset{(3\times1)}{M^*} &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t^{*\prime}X_t^*\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^2}X_t^{*\prime}Y_t^*\right) \qquad (3.20) \\ \underset{(3\times3)}{V^*} &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t^{*\prime}X*_t\right)^{-1} \end{aligned}$$

Note that the mean and variance in equation 3.20 is identical to the expressions in equation 3.7. We have simply replaced the dependent and independent variables with our transformed data.

Step 3 Conditional on σ^2 and B we sample from the conditional distribution of ρ. Given the previous draw of B we can calculate the model residuals $v_t = Y_t - (\alpha + B_1Y_{t-1} + B_2Y_{t-2})$ and treat the equation $v_t = \rho v_{t-1} + \varepsilon_t, \varepsilon_t \sim N(0, \sigma^2)$ as an AR(1) model in v_t. Therefore, the conditional distribution for ρ is simply a normal distribution with the mean and variance derived in section 2.1. That is, the conditional distribution is

$$H\left(\rho|\sigma^2, B, Y_t\right)\tilde{}N\left(\rho^*, z^*\right) \tag{3.21}$$

where

$$\begin{aligned} \underset{(1\times1)}{\rho^*} &= \left(\Sigma_\rho^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1}\left(\Sigma_\rho^{-1}\rho^0 + \frac{1}{\sigma^2}x_t'y_t\right) \qquad (3.22) \\ \underset{(1\times1)}{z^*} &= \left(\Sigma_\rho^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1} \end{aligned}$$

where $y_t = v_t$ and $x_t = v_{t-1}$. With a value for ρ^* and z^* in hand, we simply draw ρ from the normal distribution with this mean and variance

$$\underset{(1\times1)}{\rho^1} = \underset{(1\times1)}{\rho^*} + \left[\underset{(1\times1)}{\bar{\rho}} \times \underset{(1\times1)}{(z^*)^{1/2}}\right]$$

where $\bar{\rho}$ is a draw from the standard normal distribution.

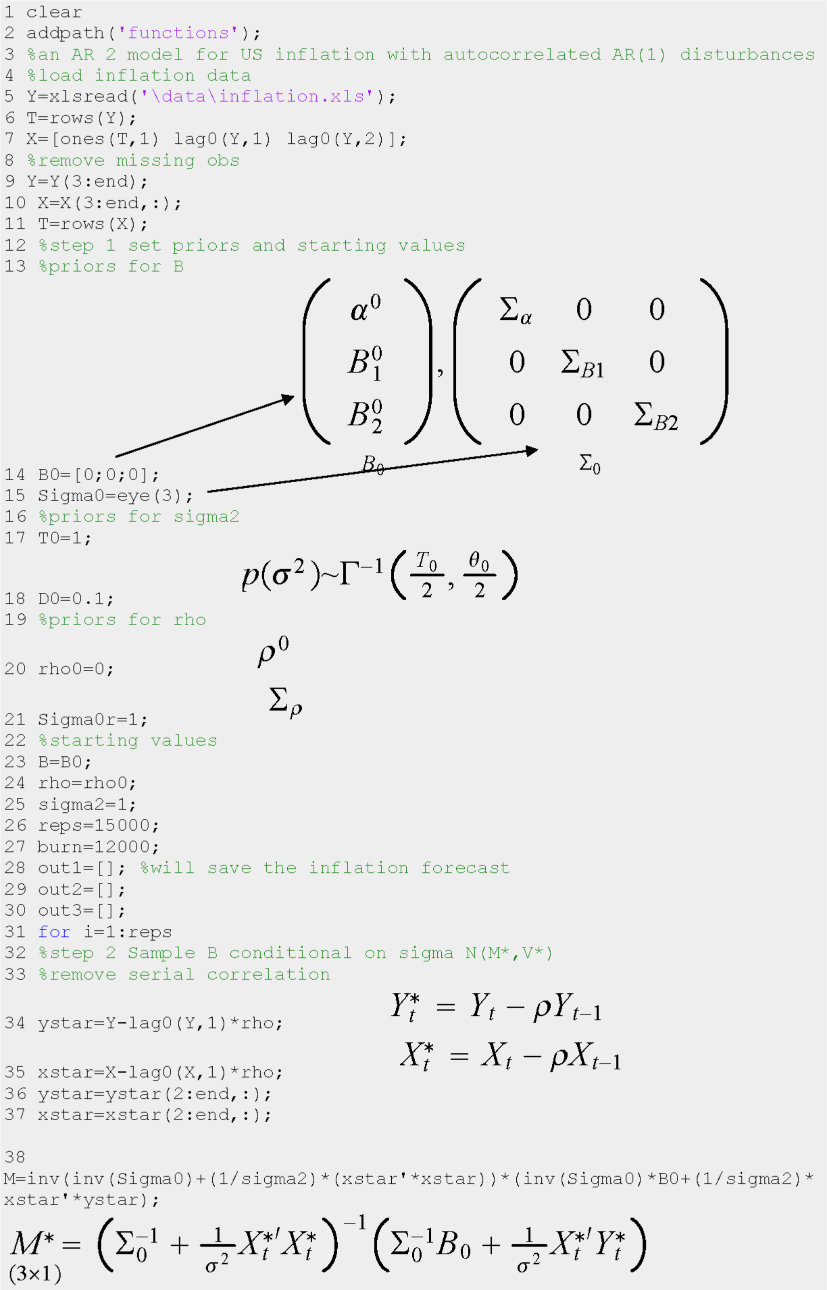
Step 4 Given a draw for B and ρ we draw σ^2 form its conditional posterior distribution. As shown in section 2.2 the conditional posterior distribution for σ^2 is inverse Gamma

$$H\left(\sigma^2|B, Y_t\right)\tilde{}\Gamma^{-1}\left(\frac{T_1}{2}, \frac{\theta_1}{2}\right) \tag{3.23}$$

where

$$\begin{aligned} T_1 &= T_0 + T \qquad (3.24) \\ \theta_1 &= \theta_0 + \left(Y_t^* - B^1X_t^*\right)'\left(Y_t^* - B^1X_t^*\right) \end{aligned}$$

Note that the term $\left(\left(Y_t^* - B^1X_t^*\right)'\left(Y_t^* - B^1X_t^*\right)\right)$ is calculated using the *iid* residuals $Y_t^* - B^1X_t^*$ (where B^1 is the previous draw of the coefficient vector).



```
clear
addpath('functions');
%an AR 2 model for US inflation with autocorrelated AR(1) disturbances
%load inflation data
Y=xlsread('\data\inflation.xls');
T=rows(Y);
X=[ones(T,1) lag0(Y,1) lag0(Y,2)];
%remove missing obs
Y=Y(3:end);
X=X(3:end,:);
T=rows(X);
%step 1 set priors and starting values
%priors for B
```

$$\underbrace{\begin{pmatrix}\alpha^0\\ B_1^0\\ B_2^0\end{pmatrix}}_{B_0},\underbrace{\begin{pmatrix}\Sigma_\alpha & 0 & 0\\ 0 & \Sigma_{B1} & 0\\ 0 & 0 & \Sigma_{B2}\end{pmatrix}}_{\Sigma_0}$$

```
B0=[0;0;0];
Sigma0=eye(3);
%priors for sigma2
T0=1;
```

$$p(\sigma^2)\sim\Gamma^{-1}\left(\frac{T_0}{2},\frac{\theta_0}{2}\right)$$

```
D0=0.1;
%priors for rho
rho0=0;
```

$$\rho^0$$

```
Sigma0r=1;
```

$$\Sigma_\rho$$

```
%starting values
B=B0;
rho=rho0;
sigma2=1;
reps=15000;
burn=12000;
out1=[]; %will save the inflation forecast
out2=[];
out3=[];
for i=1:reps
%step 2 Sample B conditional on sigma N(M*,V*)
%remove serial correlation
ystar=Y-lag0(Y,1)*rho;
```

$$Y_t^* = Y_t - \rho Y_{t-1}$$

```
xstar=X-lag0(X,1)*rho;
```

$$X_t^* = X_t - \rho X_{t-1}$$

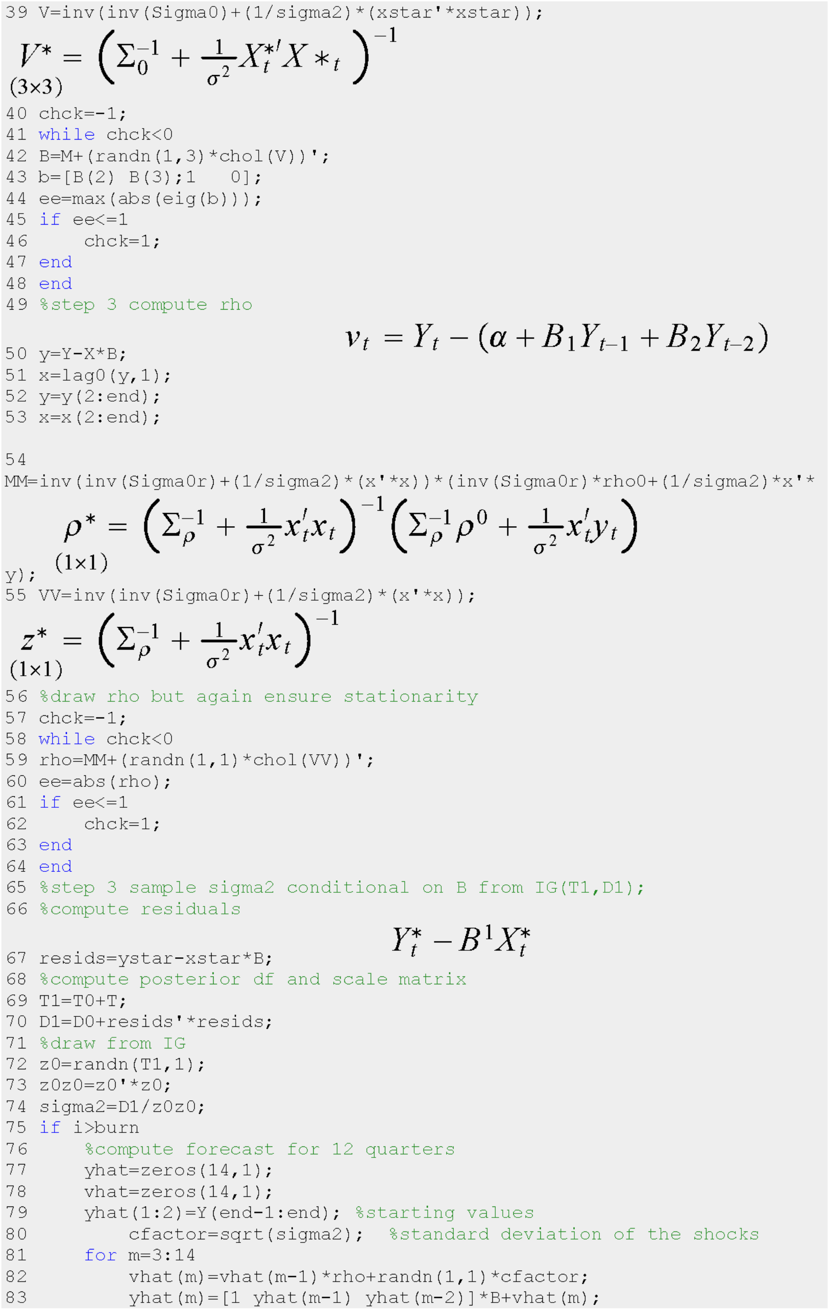
```
ystar=ystar(2:end,:);
xstar=xstar(2:end,:);

M=inv(inv(Sigma0)+(1/sigma2)*(xstar'*xstar))*(inv(Sigma0)*B0+(1/sigma2)*
xstar'*ystar);
```

$$\underset{(3\times 1)}{M^*} = \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t^{*\prime}X_t^*\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^2}X_t^{*\prime}Y_t^*\right)$$

FIGURE 10. Example 3: Matlab code

Step 5 Repeat steps 2 and 4 M times to obtain $B^1...B^M$, $\rho^1...\rho^M$and $\left(\sigma^2\right)^1....\left(\sigma^2\right)^M$. The last H values of B,ρ and σ^2 from these iterations is used to form the empirical distribution of these parameters. This example shows that we reduce a relatively complicated model into three steps, each of which are simple and based on the linear regression framework. As seen in later chapters, Gibbs sampling will operate in exactly the same way in more complicated models–i.e. by breaking the problem down into smaller simpler steps.

3.6. Gibbs Sampling for a linear regression with serial correlation in Matlab (example3.m). The matlab code for this example is a simple extension of example1.m and shown in figures 10, 11 and 12. Note that the



```
39 V=inv(inv(Sigma0)+(1/sigma2)*(xstar'*xstar));
```

$$\underset{(3\times3)}{V^*} = \left(\Sigma_0^{-1} + \frac{1}{\sigma^2}X_t^{*\prime}X*_t\right)^{-1}$$

```
40 chck=-1;
41 while chck<0
42 B=M+(randn(1,3)*chol(V))';
43 b=[B(2) B(3);1   0];
44 ee=max(abs(eig(b)));
45 if ee<=1
46     chck=1;
47 end
48 end
49 %step 3 compute rho
```

$$v_t = Y_t - (\alpha + B_1 Y_{t-1} + B_2 Y_{t-2})$$

```
50 y=Y-X*B;
51 x=lag0(y,1);
52 y=y(2:end);
53 x=x(2:end);

54
MM=inv(inv(Sigma0r)+(1/sigma2)*(x'*x))*(inv(Sigma0r)*rho0+(1/sigma2)*x'*
y);
```

$$\underset{(1\times1)}{\rho^*} = \left(\Sigma_\rho^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1}\left(\Sigma_\rho^{-1}\rho^0 + \frac{1}{\sigma^2}x_t'y_t\right)$$

```
55 VV=inv(inv(Sigma0r)+(1/sigma2)*(x'*x));
```

$$\underset{(1\times1)}{z^*} = \left(\Sigma_\rho^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1}$$

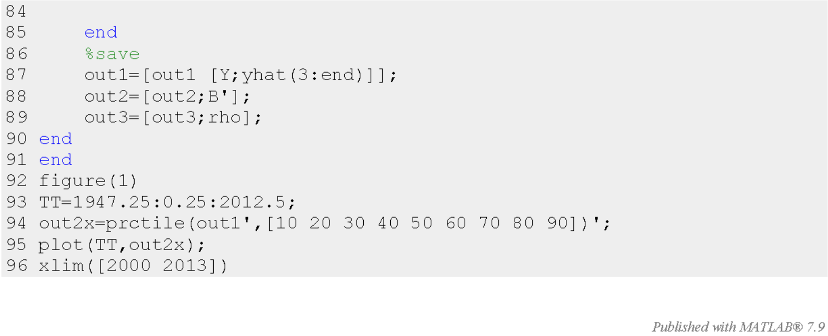
```
56 %draw rho but again ensure stationarity
57 chck=-1;
58 while chck<0
59 rho=MM+(randn(1,1)*chol(VV))';
60 ee=abs(rho);
61 if ee<=1
62     chck=1;
63 end
64 end
65 %step 3 sample sigma2 conditional on B from IG(T1,D1);
66 %compute residuals
```

$$Y_t^* - B^1 X_t^*$$

```
67 resids=ystar-xstar*B;
68 %compute posterior df and scale matrix
69 T1=T0+T;
70 D1=D0+resids'*resids;
71 %draw from IG
72 z0=randn(T1,1);
73 z0z0=z0'*z0;
74 sigma2=D1/z0z0;
75 if i>burn
76     %compute forecast for 12 quarters
77     yhat=zeros(14,1);
78     vhat=zeros(14,1);
79     yhat(1:2)=Y(end-1:end); %starting values
80         cfactor=sqrt(sigma2);  %standard deviation of the shocks
81     for m=3:14
82         vhat(m)=vhat(m-1)*rho+randn(1,1)*cfactor;
83         yhat(m)=[1 yhat(m-1) yhat(m-2)]*B+vhat(m);
```

Figure 11. Example 3: Matlab code (continued)

underlying data is exactly as before. This is loaded and lags etc created using the commands from lines 5 to 11. Lines 14 and 15 set the prior mean and variance for B for lines 17 and lines 18 sets the prior scale parameter and degrees of freedom for the inverse Gamma prior for σ^2. Lines 20 and 21 set the mean and variance for the normal prior for ρ, i.e. $p(\rho)\tilde{}N\left(\rho^0, \Sigma_\rho\right)$ Lines 23 to 25 set starting values for the parameters. The first step of the Gibbs sampling algorithm starts on line 34 and 35 where we create $Y_t^* = Y_t - \rho Y_{t-1}, X_t^* = X_t - \rho X_{t-1}$, the data transformed to remove serial correlation. Lines 38 and 39 calculate the mean and the variance of the conditional distribution of B using this tranformed data. As in the previous example, lines 40 to 48 draw B from its conditional distribution, but ensure that the draw is stable. Line 50 calculates the (serially correlated) residuals $v_t = Y_t - (\alpha + B_1 Y_{t-1} + B_2 Y_{t-2})$



```
84
85     end
86     %save
87     out1=[out1 [Y;yhat(3:end)]];
88     out2=[out2;B'];
89     out3=[out3;rho];
90 end
91 end
92 figure(1)
93 TT=1947.25:0.25:2012.5;
94 out2x=prctile(out1',[10 20 30 40 50 60 70 80 90])';
95 plot(TT,out2x);
96 xlim([2000 2013])
```

Published with MATLAB® 7.9

FIGURE 12. Example 3: Matlab code (continued)

using the previous draw of α, B_1 and B_2 and lines 50 and 51 create $y_t = v_t$ and $x_t = v_{t-1}$. Line 54 calculates the mean of the conditional distribution of ρ, $\underset{(1\times1)}{\rho^{*}} = \left(\Sigma_{\rho}^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1}\left(\Sigma_{\rho}^{-1}\rho^0 + \frac{1}{\sigma^2}x_t'y_t\right)$ while line 55 calculates the variance of the conditional distribution $\underset{(1\times1)}{z^{*}} = \left(\Sigma_{\rho}^{-1} + \frac{1}{\sigma^2}x_t'x_t\right)^{-1}$. Line 59 draws ρ from the normal distribution using $\underset{(1\times1)}{\rho^{1}} = \underset{(1\times1)}{\rho^{*}} + \left[\underset{(1\times1)}{\bar{\rho}} \times \underset{(1\times1)}{(z^{*})^{1/2}}\right]$ and the while loop ensures that ρ is less than or equal to 1 in absolute value. Line 67 calculates the serially uncorrelated residuals $Y_t^{*} - B^1 X_t^{*}$. These are used on lines 69 to 74 to draw σ^2 from

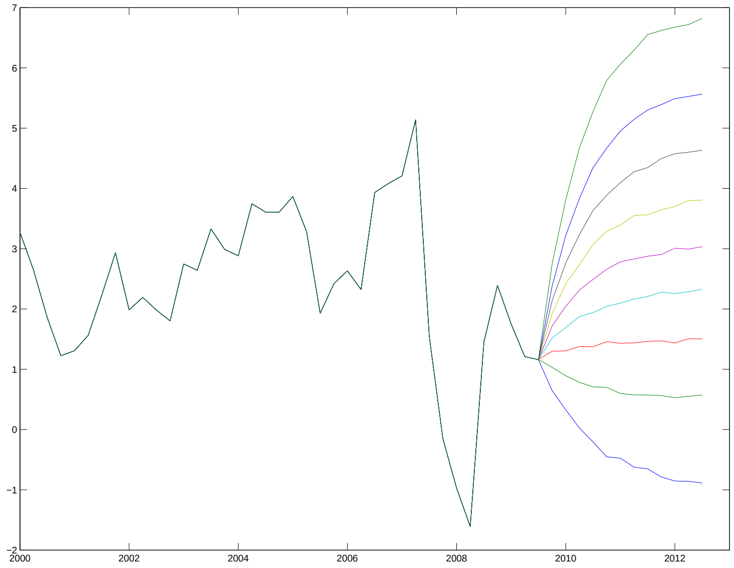


FIGURE 13. The distribution of the inflation forecast using example3.m.

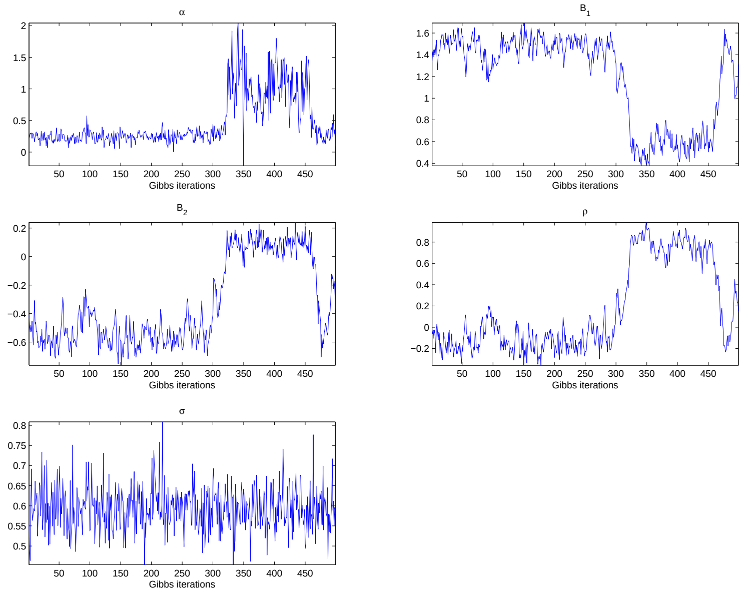


FIGURE 14. Sequence of retained Gibbs draws for the AR(2) model with serial correlation using 500 iterations

the inverse Gamma distribution. After the burn-in stage, the code computes the forecast from this AR(2) model with serial correlation. Line 82 projects forward the equation for the error term i.e. $v_{t+i} = \rho v_{t+i-1} + \sigma v^*$ where v^* is a standard normal shock. Line 83 calculates the projected value of inflation given v_{t+i}. This is done for each retained draw of the Gibbs sampler with the results (along with actual data) stored in the matrix out1 (line 87). The resulting distribution of the forecast is seen in 13.

3.7. Convergence of the Gibbs sampler. A question we have ignored so far is: How many draws of the Gibbs sampling algorithm do we need before we can be confident that the draws from the conditional posterior distributions have converged to the marginal posterior distribution? Generally researchers proceed in two steps

- Choose a minimum number of draws M and run the Gibbs sampler
- Check if the algorithm has converged (using the procedures introduced below). If there is insufficient evidence for convergence, increase M and try again.

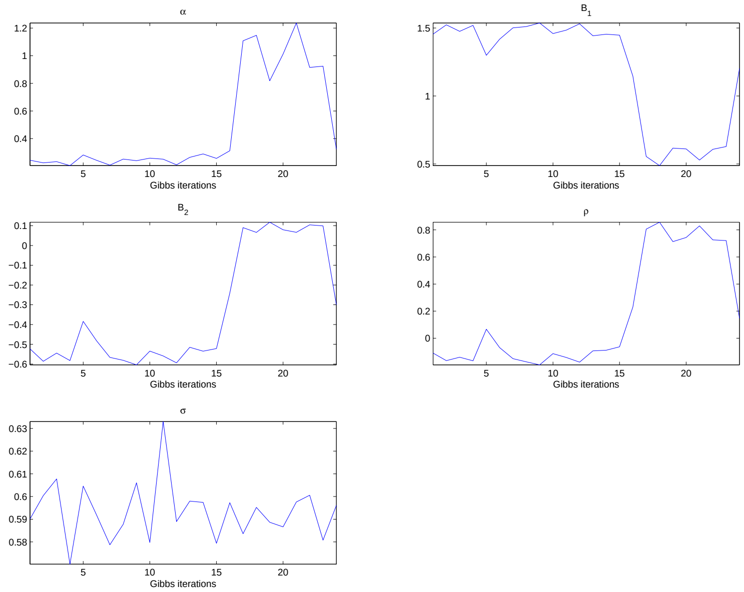


Figure 15. Recursive means of the retained Gibbs draws for the AR(2) model with serial correlation using 500 iterations

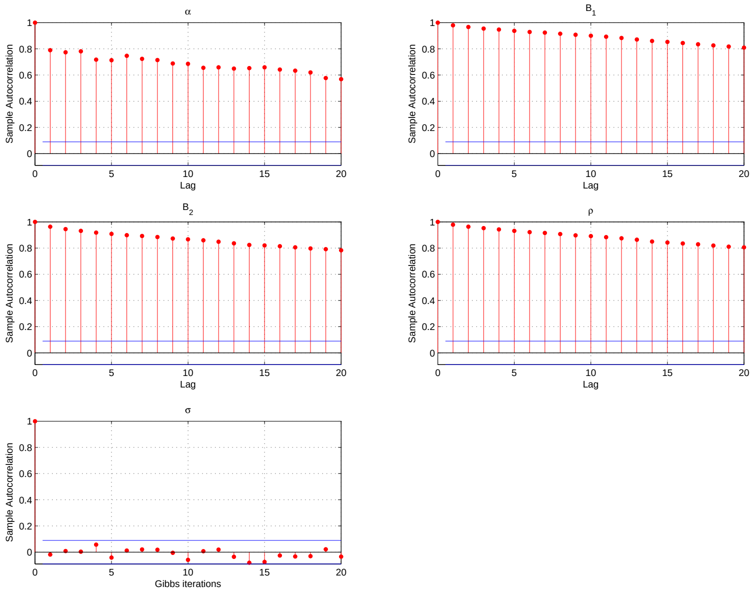


Figure 16. Autocorrelation of the retained Gibbs draws for the AR(2) model with serial correlation using 500 iterations

The simplest way to check convergence is to examine the sequence of retained draws. If the Gibbs sampler has converged to the target distibution, then the retained draws should fluctuate randomly around a stationary mean and not display any trend. This visual inspection is usually easier if one plots the recursive mean of the retained draws. If the Gibbs sampler has converged, then the recursive mean should show little fluctuation. A related method to examine convergence is plot the autocorrelation of the retained draws. If convergence has occurred, the sequence of draws should display little autocorrelation (i.e. they should be fluctuating randomly around a stationary mean).

In order to illustrate these ideas, we plot the sequence of retained draws, the recursive means of those draws and the autocorrelation functions of the retained draws for the parameters of the model examined in section 3.6. In particular, we estimate the AR(2) model with serial correlation using 500 Gibbs iterations (using the file example3.m) and retain all of these draws. Figures 14, 15 and 16 examine the convergence of the model. Figures 14 and 15 clearly show that the Gibbs draws are not stationary with the recursive mean for α, B_1, B_2 and ρ showing a large change

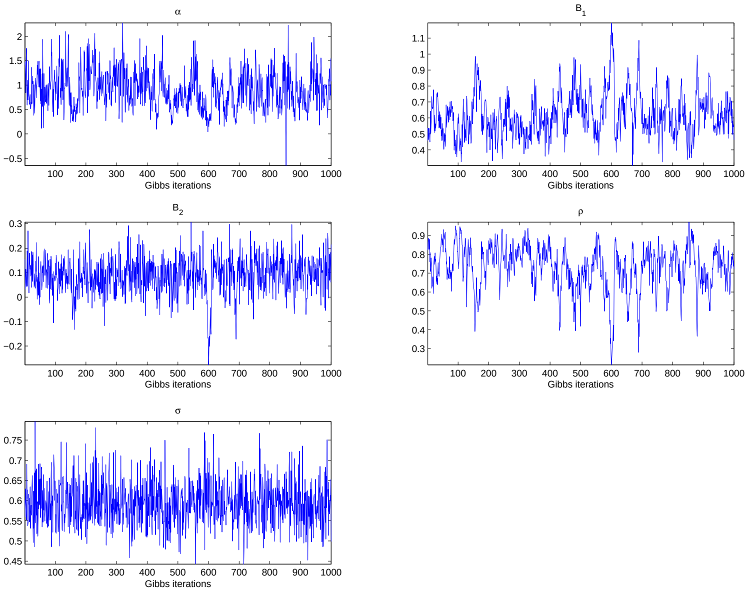


FIGURE 17. Sequence of retained Gibbs draws for the AR(2) model with serial correlation using 25000 iterations

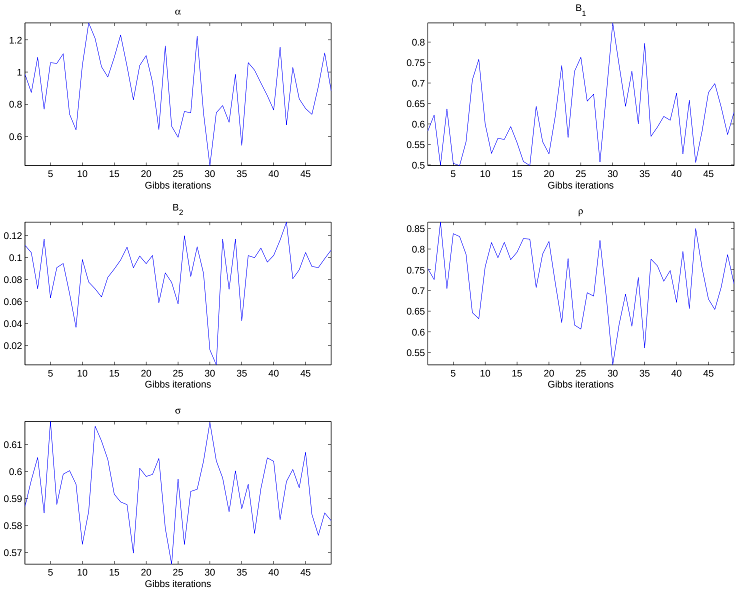


FIGURE 18. Recursive means of retained Gibbs draws for the AR(2) model with serial correlation using 25000 iterations

after 300 iterations (but σ^2 appears to have converged with the draws fluctuating around a stationary mean). This also shows up in the autocorrelation functions, with the autocorrelation high for α, B_1, B_2 and ρ. These figures can be produced using the file example4.m. These results would indicate that a higher number of Gibbs iterations are required. Figures 17, 18 and 19 plot the same objects when 25000 Gibbs iterations are used (with 24000 as the number of burn-in iterations). The sequence of retained draws and the recursive means appear substantially more stable. The autocorrelations for α, B_1, B_2 and ρ decay much faster in figure 19.

These graphical methods to assess convergence are widely used in applied work. A more formal test of convergence has been proposed by Geweke (1991). The intuition behind this test is related to the idea behind the recursive mean plot: If the Gibbs sampler has converged then the mean over different sub-samples of the retained draws should be similar. Geweke (1991) suggests the following procedure:

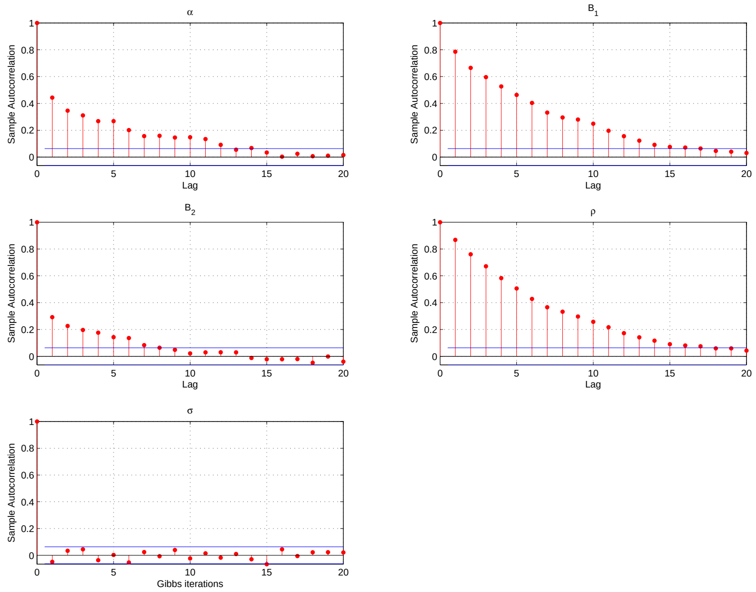


Figure 19. Autocorrelation of retained Gibbs draws for the AR(2) model with serial correlation using 25000 iterations

(1) Divide the retained Gibbs draws of the model parameters θ into two subsamples N_1 N_2 where Geweke (1991) recommends $N_1 = 0.1N, N_2 = 0.5N$ where N denotes the total number of retained draws.
(2) Compute averages $M_1 = \sum_{i=1}^{N_1} \frac{\theta_i}{N_1}$ and $M_2 = \sum_{i=N_2+1}^{N} \frac{\theta_i}{N_2}$
(3) Compute the asymptotic variance $\frac{S_1(0)}{N_1}$ and $\frac{S_2(0)}{N_2}$ where $S(\varpi)$ is the spectral density at frequency ϖ. Note that this estimate of the variance takes into account the possibility that the Gibbs sequence may be autocorrelated. For a description of spectral analysis see Hamilton (1994) and Canova (2007).
(4) Then the test statistic

$$Z = \frac{M_1 - M_2}{\sqrt{\frac{S_1(0)}{N_1} + \frac{S_2(0)}{N_2}}} \tag{3.25}$$

is asymptotically distributed as $N(0,1)$. Large values of this test statistic indicate a significant difference in the mean across the retained draws and suggests that one should increase the number of initial Gibbs iterations (i.e. increase the number of burn-in draws).

Geweke (1991) suggests a related statistic to judge the efficiency of the Gibbs sampler and to gauge the total number of Gibbs iterations to be used. The intuition behind this measure of relative numerical efficiency (RNE) is as follows. Suppose one could take iid draws of $\theta_i \in \{\theta_1, \theta_2 .... \theta_N\}$ directly from the posterior. Then the variance of the posterior mean $E(\theta_i) = \frac{1}{N}\sum_i \theta_i$ is given by

$$\begin{aligned} VAR(E(\theta_i)) &= \frac{1}{N^2}VAR(\theta_1) + \frac{1}{N^2}VAR(\theta_2) + ... \frac{1}{N^2}VAR(\theta_N) \\ &= VAR(\theta_i)/N \end{aligned}$$

However, in practice one uses the Gibbs sampler to approximate draws from the posterior. These Gibbs draws are likely to be autocorrelated and a measure of their variance which takes this into account is $S(0)/N$. Thus a measure of the RNE is

$$RNE = \frac{\widehat{VAR(\theta_i)}}{S(0)} \tag{3.26}$$

where $\widehat{VAR(\theta_i)}$ is the sample variance of the Gibbs draws $\theta_1, \theta_2 .... \theta_N$. If the Gibbs sampler has converged then RNE should be close to 1 as the variance of the iid draws $\widehat{VAR(\theta_i)}$ should be similar to the measure of the variance that takes any possible autocorrelation into account.

The file example5.m illustrates the calculation of the statistics in equation 3.25 and 3.26.

4. Further Reading

- An intuitive description of the Gibbs sampling algorithm for the linear regression model can be found in Kim and Nelson (1999) Chapter 7. Gauss codes for the examples in Kim and Nelson (1999) are available at http://econ.korea.ac.kr/~cjkim/SSMARKOV.htm.

- A more formal treatment of the linear regression model from a Bayesian perspective can be found in Koop (2003), Chapters 2, 3 and 4.
- The appendix in Zellner (1971) provides a detailed description of the Inverse Gamma and Gamma distributions. See Bauwens *et al.* (1999) for a detailed description of algorithms to draw from these distributions.

5. Appendix: Calculating the marginal likelihood for the linear regression model using the Gibbs sampler.

Consider the following linear regression model

$$Y_t = BX_t + v_t, v_t\tilde{\ }N(0, \sigma^2)$$

The prior distributions are assumed to be

$$P(B)\tilde{\ }N(B_0, \Sigma_0)$$
$$P\left(\sigma^2\right)\tilde{\ }IG(V_0, T_0)$$

The posterior distribution of the model parameters $\Phi = B, \sigma^2$ is defined via the Bayes rule

$$H\left(\Phi|Y\right) = \frac{F\left(Y|\Phi\right) \times P\left(\Phi\right)}{F\left(Y\right)} \tag{5.1}$$

where $F\left(Y|\Phi\right) = \left(2\pi\sigma^2\right)^{-\frac{T}{2}} \exp\left(-\frac{1}{2\sigma^2}\left(Y_t - BX_t\right)'\left(Y_t - BX_t\right)\right)$ is the likelihood function, $P\left(\Phi\right)$ is the joint prior distribution while $F\left(Y\right)$ is the marginal likelihood that we want to compute. Chib (1995) suggests computing the marginal likelihood by re-arranging equation 5.1. Note that in logs we can re-write equation 5.1 as

$$\ln F\left(Y\right) = \ln F\left(Y|\Phi\right) + \ln P\left(\Phi\right) - \ln H\left(\Phi|Y\right) \tag{5.2}$$

Note that equation 5.2 can be evaluated at any value of the parameters Φ to calculate $\ln F\left(Y\right)$. In practice a high density point Φ^* such as the posterior mean or posterior mode is used.

The first two terms on the right hand side of equation 9.3 are easy to evaluate at Φ^*. The first term is the log likelihood function. The second term is the joint prior which is the product of a normal density for the coefficients and an inverse Gamma density for the variance (see example below). Evaluating the third term $\ln H\left(\Phi^*|Y\right)$ is more complicated as the posterior distribution is generally not known in closed form. Chib (1995) shows how this term can be evaluated using the output from the Gibbs sampling algorithm used to approximate the posterior distribution for Φ. Recall that $H\left(\Phi^*|Y\right) = H\left(B^*, \sigma^{2*}\right)$ where have dropped the conditioning on y on the right hand side for simplicity. The marginal, conditional decomposition of this distribution is

$$H\left(B^*, \sigma^{2*}\right) = H\left(B^*|\sigma^{2*}\right) \times H\left(\sigma^{2*}\right) \tag{5.3}$$

The first term $H\left(B^*|\sigma^{2*}\right)$ is the conditional posterior distribution for the regression coefficients. Recall that this a normal distribution with mean and variance given by

$$\begin{aligned} M^* &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^{2*}}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0 + \frac{1}{\sigma^{2*}}X_t'Y_t\right) \\ V^* &= \left(\Sigma_0^{-1} + \frac{1}{\sigma^{2*}}X_t'X_t\right)^{-1} \end{aligned}$$

and therefore can be easily evaluated at B^* and σ^{2*}.

The second term in equation 5.3 $H\left(\sigma^{2*}\right)$ can be evaluated using the weak law of large numbers (see Koop (2003) Appendix B). That is

$$H\left(\sigma^{2*}\right) \approx \frac{1}{S}\sum_{s=1}^{S} H\left(\sigma^{2*}|B_s\right)$$

where B_s denotes $s = 1, 2....S$ draws of the Gibbs sampler. Note that the conditional distribution is simply the Inverse Gamma distribution derived for section 2.2 above.

The marginal likelihood is then given by

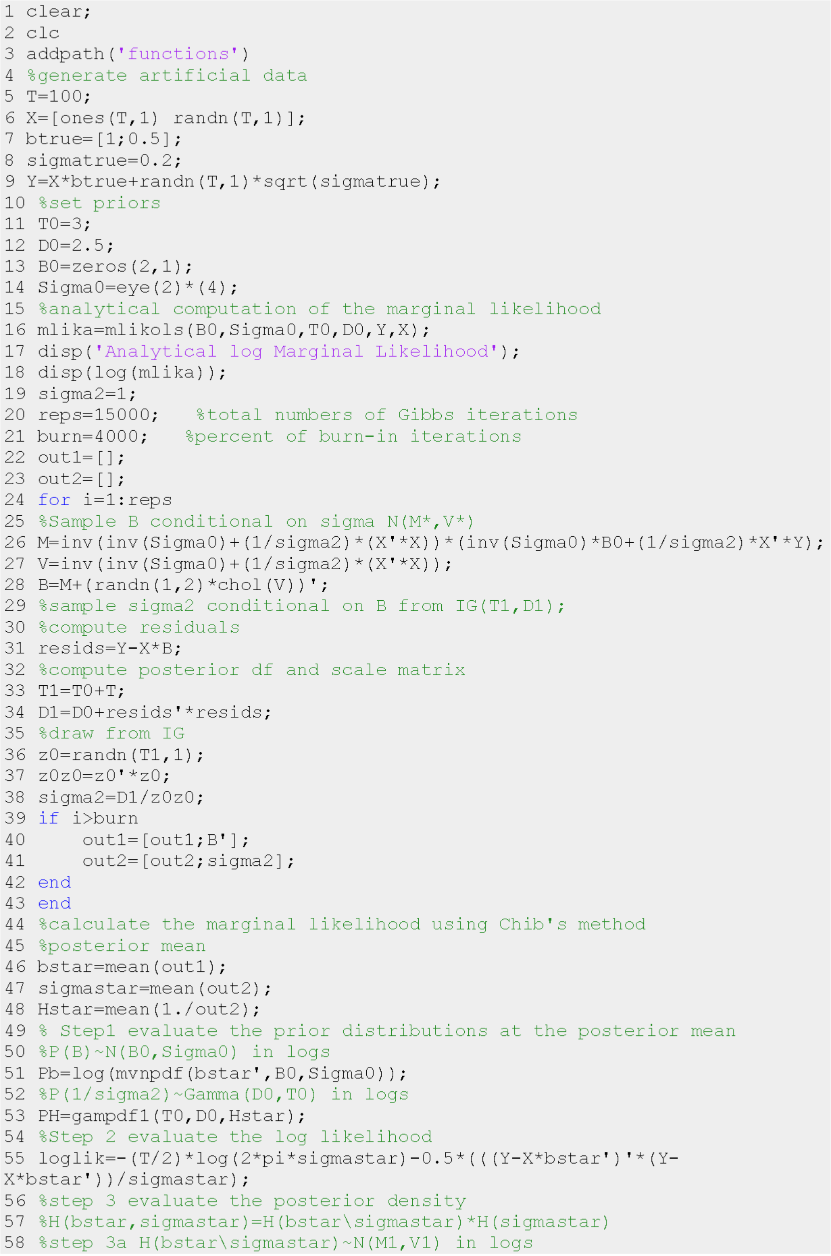
$$\ln F\left(Y\right) = \ln F\left(Y|\Phi\right) + \ln P\left(\Phi\right) - \ln H\left(B^*|\sigma^{2*}\right) - \ln H\left(\sigma^{2*}\right) \tag{5.4}$$

As an example we consider the following linear regression model based on 100 artificial observations

$$y_t = 1 + 0.5x_t + v_t, VAR(v_t) = 0.2$$

where $x_t\tilde{\ }N(0, 1)$. We assume a natural conjugate prior of the form $P(B|\sigma^2)\tilde{\ }N\left(\left(\begin{array}{c} 0 \\ 0 \end{array}\right), 4\sigma^2\left(\begin{array}{cc} 1 & 0 \\ 0 & 1 \end{array}\right)\right)$ and $P\left(\sigma^2\right)\tilde{\ }IG(2.5, 3)$.

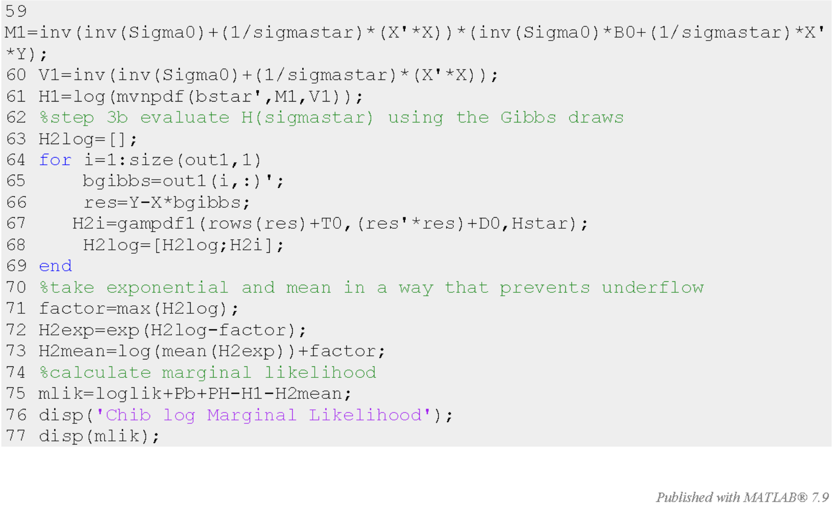
The matlab code for this example is shown in figures 20 and 21. The code on Lines 5 to 9 generates the artificial data. We set the priors on lines 11 to 14. On line 16 we calculate the marginal likelihood for this model analytically using the formula on page 41 in Koop (2003). We can now compare this estimate with the estimate produced using Chib's method. The Gibbs sampler used to estimate the model is coded on lines 19 to 43. Line 46 calculates



```
1 clear;
2 clc
3 addpath('functions')
4 %generate artificial data
5 T=100;
6 X=[ones(T,1) randn(T,1)];
7 btrue=[1;0.5];
8 sigmatrue=0.2;
9 Y=X*btrue+randn(T,1)*sqrt(sigmatrue);
10 %set priors
11 T0=3;
12 D0=2.5;
13 B0=zeros(2,1);
14 Sigma0=eye(2)*(4);
15 %analytical computation of the marginal likelihood
16 mlika=mlikols(B0,Sigma0,T0,D0,Y,X);
17 disp('Analytical log Marginal Likelihood');
18 disp(log(mlika));
19 sigma2=1;
20 reps=15000;   %total numbers of Gibbs iterations
21 burn=4000;   %percent of burn-in iterations
22 out1=[];
23 out2=[];
24 for i=1:reps
25 %Sample B conditional on sigma N(M*,V*)
26 M=inv(inv(Sigma0)+(1/sigma2)*(X'*X))*(inv(Sigma0)*B0+(1/sigma2)*X'*Y);
27 V=inv(inv(Sigma0)+(1/sigma2)*(X'*X));
28 B=M+(randn(1,2)*chol(V))';
29 %sample sigma2 conditional on B from IG(T1,D1);
30 %compute residuals
31 resids=Y-X*B;
32 %compute posterior df and scale matrix
33 T1=T0+T;
34 D1=D0+resids'*resids;
35 %draw from IG
36 z0=randn(T1,1);
37 z0z0=z0'*z0;
38 sigma2=D1/z0z0;
39 if i>burn
40     out1=[out1;B'];
41     out2=[out2;sigma2];
42 end
43 end
44 %calculate the marginal likelihood using Chib's method
45 %posterior mean
46 bstar=mean(out1);
47 sigmastar=mean(out2);
48 Hstar=mean(1./out2);
49 % Step1 evaluate the prior distributions at the posterior mean
50 %P(B)~N(B0,Sigma0) in logs
51 Pb=log(mvnpdf(bstar',B0,Sigma0));
52 %P(1/sigma2)~Gamma(D0,T0) in logs
53 PH=gampdf1(T0,D0,Hstar);
54 %Step 2 evaluate the log likelihood
55 loglik=-(T/2)*log(2*pi*sigmastar)-0.5*(((Y-X*bstar')'*(Y-
X*bstar'))/sigmastar);
56 %step 3 evaluate the posterior density
57 %H(bstar,sigmastar)=H(bstar\sigmastar)*H(sigmastar)
58 %step 3a H(bstar\sigmastar)~N(M1,V1) in logs
```

FIGURE 20. Matlab code for calculating the marginal likelihood

the posterior mean of the coefficients, line 47 calculates the posterior mean of the variance while line 48 calculates the posterior mean of $1/\sigma^2$. For computational convenience, when considering the prior $\ln P(\Phi)$ and the posterior distribution $\ln H(\Phi|Y)$ in the expression for the marginal likelihood (see equation 5.2) we consider the precision $1/\sigma^2$ and use the Gamma distribution. This allows us to use built in matlab functions to evaluate the Gamma PDF. On line 51, we evaluate the log of the prior distribution of the VAR coefficients $P(B|\sigma^2)$ at the posterior mean. Line 53 evaluates the Gamma posterior for the precision. The function gampdf1 converts the two parameters of the distribution: the degrees of freedom T_0 and scale parameter D_0 into the parameters $A = T_0/2$ and $B = 2/D_0$ as expected by the parameterisation of the Gamma distribution used by Matlab in its built in function gampdf.



```
59
M1=inv(inv(Sigma0)+(1/sigmastar)*(X'*X))*(inv(Sigma0)*B0+(1/sigmastar)*X'
*Y);
60 V1=inv(inv(Sigma0)+(1/sigmastar)*(X'*X));
61 H1=log(mvnpdf(bstar',M1,V1));
62 %step 3b evaluate H(sigmastar) using the Gibbs draws
63 H2log=[];
64 for i=1:size(out1,1)
65     bgibbs=out1(i,:)';
66     res=Y-X*bgibbs;
67    H2i=gampdf1(rows(res)+T0,(res'*res)+D0,Hstar);
68     H2log=[H2log;H2i];
69 end
70 %take exponential and mean in a way that prevents underflow
71 factor=max(H2log);
72 H2exp=exp(H2log-factor);
73 H2mean=log(mean(H2exp))+factor;
74 %calculate marginal likelihood
75 mlik=loglik+Pb+PH-H1-H2mean;
76 disp('Chib log Marginal Likelihood');
77 disp(mlik);
```

Published with MATLAB® 7.9

FIGURE 21. Matlab code for calculating the marginal likelihood continued

Line 55 evaluates the log likelihood at the posterior mean. Lines 56 to 61 evaluate the term $H\left(B^{*}|\sigma^{2*}\right)$ in the factorisation of the posterior $H\left(B^{*},1/\sigma^{2*}\right)=H\left(B^{*}|\sigma^{2*}\right)\times H\left(1/\sigma^{2*}\right)$. Lines 63 to 69 evaluate the term $H\left(1/\sigma^{2*}\right)$. Each iteration in the loop evaluates $H\left(1/\sigma^{2*}|B_{j}\right)$. Note that this is simply the Gamma distribution with degrees of freedom T_0+T and scale parameter $D_0+v_t'v_t$ where the residuals v_t are calculated using each Gibbs draw of the regression coefficients $B_j.T_0$ and D_0 denote the prior degrees of freedom and prior scale parameter respectively. Line 73 constructs $H\left(1/\sigma^{2*}\right)\approx\frac{1}{J}\sum_{j=1}^{J}H\left(1/\sigma^{2*}|B_{J}\right)$. The marginal likelihood is calculated using equation 5.2 on line 75 of the code.

CHAPTER 2

Gibbs Sampling for Vector Autoregressions

This chapter introduces Bayesian simulation methods for Vector Autoregressions (VARs). The estimation of these models typically involves a large number of parameters. As a consequence, estimates of objects of interest such as impulse response functions and forecasts can become imprecise in large scale models. By incorporating prior information into the estimation process, the estimates obtained using Bayesian methods are generally more precise than those obtained using the standard classical approach. In addition, bayesian simulation methods such as Gibbs sampling provide an efficient way not only to obtain point estimates but also to characterise the uncertainty around those point estimates. Therefore we focus on estimation of VARs *via Gibbs sampling* in this chapter.

Note, however, that under certain prior distributions, analytical expressions exist for the marginal posterior distribution of the VAR parameters. A more general treatment of Bayesian VARs can be found in Canova (2007) amongst others. See http://apps.eui.eu/Personal/Canova/Courses.html for F.Canova's BVAR code.

This chapter focusses on two key issues

- It states the conditional posterior distributions of the VAR parameters required for Gibbs sampling and discussed the Gibbs sampling algorithm for VARs
- We go through the practical details of setting different type of priors for VAR parameters
- We focus on implementation of Gibbs sampling for VARs in Matlab.
- We discuss how to estimate structural VARs with sign restrictions using Matlab.

1. The Conditional posterior distribution of the VAR parameters and the Gibbs sampling algorithm

Consider the following VAR(p) model

$$\begin{aligned} Y_t &= c + B_1 Y_{t-1} + B_2 Y_{t-2} ... B_P Y_{t-p} + v_t \qquad (1.1)\\ E\left(v_t' v_s\right) &= \Sigma \text{ if } t = s \\ E\left(v_t' v_s\right) &= 0 \text{ if } t \neq s \\ E\left(v_t\right) &= 0 \end{aligned}$$

where Y_t is a $T \times N$ matrix of endogenous variables, c denotes a constant term. The VAR can be written compactly as

$$Y_t = X_t B + v_t \qquad (1.2)$$

with $X_t = \{c_i, Y_{it-1}, Y_{it-2}..., Y_{it-p}\}$.Note that as each equation in the VAR has *identical* regressors, it can be re-written as

$$y = \left(I_N \otimes X\right) b + V \qquad (1.3)$$

where $y = vec(Y_t)$ and $b = vec(B)$ and $V = vec(v_t)$.

Assume that the prior for the VAR coefficients b is normal and given by

$$p(b)\tilde{\ } N\left(\tilde{b}_0, H\right) \qquad (1.4)$$

where $\tilde{b}_0$ is a $(N \times (N \times P + 1)) \times 1$ vector which denotes the prior mean while H is a is a $[N \times (N \times P + 1)] \times [N \times (N \times P + 1)]$ matrix where the diagonal elements denote the variance of the prior. We discuss different ways of setting $\tilde{b}_0$ and H in detail below.

It can be shown that the *posterior distribution of the VAR coefficients* conditional on Σ is normal (see Kadiyala and Karlsson (1997)) . That is the conditional posterior for the coefficients is given by $H\left(b|\Sigma, Y_t\right)\tilde{\ }N\left(M^*, V^*\right)$ where

$$\begin{aligned} M^* &= \left(H^{-1} + \Sigma^{-1} \otimes X_t' X_t\right)^{-1} \left(H^{-1}\tilde{b}_0 + \Sigma^{-1} \otimes X_t' X_t \hat{b}\right) \qquad (1.5)\\ V^* &= \left(H^{-1} + \Sigma^{-1} \otimes X_t' X_t\right)^{-1} \end{aligned}$$

where $\hat{b}$ is a $(N \times (N \times P + 1)) \times 1$ vector which denotes the OLS estimates of the VAR coefficients in vectorised format $\hat{b} = vec\left(\left(X_t' X_t\right)^{-1}\left(X_t' Y_t\right)\right)$. The format of the conditional posterior mean in equation 1.5 is very similar to that discussed for the linear regression model (see section 2.1 in the previous chapter). That is the mean of the conditional posterior distribution is a weighted average of the OLS estimator $\hat{b}$ and the prior $\tilde{b}_0$ with the weights given by the inverse of the variance of each ($\Sigma^{-1} \otimes X_t' X_t$ is the inverse of $\hat{b}$ while H^{-1} is the inverse of the variance of the prior).

The conjugate prior for the VAR covariance matrix is an *inverse Wishart distribution* with prior scale matrix $\bar{S}$ and prior degrees of freedom α.

$$p(\Sigma)\tilde{}IW\left(\bar{S},\alpha\right) \tag{1.6}$$

Definition 3. *If Σ is a $n\times n$ positive definite matrix, it is distributed as an inverse Wishart with the following density $P\left(\Sigma\right)=k\frac{|H|^{v/2}}{|\Sigma|^{(v+n+1)/2}}\exp\left(-0.5tr\Sigma^{-1}H\right)$ where $k^{-1}=2^{vn/2}\pi^{n(n-1)/4}\prod_{i=1}^{n}\Gamma\left[\left(v+1-i\right)/2\right]$, H is the scale matrix and v denotes the degrees of freedom. See Zellner (1971) pp395 for more details.*

Informally, one can think of the inverse Wishart distribution as a multivariate version of the inverse Gamma distribution introduced in the context of the linear regression model in the previous chapter. Given the prior in equation 1.6, the posterior for Σ conditional on b is also inverse Wishart $H\left(\Sigma|b,Y_t\right)\tilde{}IW\left(\bar{\Sigma},T+\alpha\right)$ where T is the sample size and

$$\bar{\Sigma}=\bar{S}+\left(Y_t-X_tB\right)'\left(Y_t-X_tB\right) \tag{1.7}$$

Note that B denotes the VAR coefficients reshaped into $(N\times P+1)$ by N matrix.

1.1. Gibbs sampling algorithm for the VAR model. The Gibbs sampling algorithm for the VAR model consists of the following steps:

Step 1 Set priors for the VAR coefficients and the covariance matrix. As discussed above, the prior for the VAR coefficients is normal and given by $p(b)\tilde{}N\left(\tilde{b}_0,H\right)$. The prior for the covariance matrix of the residuals Σ is inverse Wishart and given by $IW\left(\bar{S},\alpha\right)$. Set a starting value for Σ (e.g. the OLS estimate of Σ).

Step 2 Sample the VAR coefficients from its conditional posterior distribution $H\left(b|\Sigma,Y_t\right)\tilde{}N\left(M^*,V^*\right)$ where

$$\underset{(N\times(N\times P+1))\times 1}{M^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t'X_t\right)^{-1}\left(H^{-1}\tilde{b}_0+\Sigma^{-1}\otimes X_t'X_t\hat{b}\right) \tag{1.8}$$

$$\underset{(N\times(N\times P+1))\times(N\times(N\times P+1))}{V^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t'X_t\right)^{-1} \tag{1.9}$$

Once M^* and V^* are calculated, the VAR coefficients are drawn from the normal distribution (see algorithm 1 in Chapter 1)

$$\underset{((N\times(N\times P+1))\times 1)}{b^1}=\underset{((N\times(N\times P+1))\times 1)}{M^*}+\left[\underset{(1\times(N\times(N\times P+1)))}{\bar{b}}\times\underset{(N\times(N\times P+1))\times(N\times(N\times P+1))}{\left(V^*\right)^{1/2}}\right] \tag{1.10}$$

Step 3 Draw Σ from its conditional distribution $H\left(\Sigma|b,Y_t\right)\tilde{}IW\left(\bar{\Sigma},T+\alpha\right)$ where $\bar{\Sigma}=\bar{S}+\left(Y_t-X_tB^1\right)'\left(Y_t-X_tB^1\right)$ where B^1 is the previous draw of the VAR coefficients reshaped into a matrix with dimensions $(N\times P+1)\times N$ so it is conformable with X_t.

Algorithm 3. *To draw a matrix $\hat{\Sigma}$ from the IW distribution with v degrees of freedom and scale parameter S, draw a matrix Z with dimensions $v\times n$, from the multivariate normal $N(0,S^{-1})$. Then the draw from the inverse Wishart distribution is given by the following transformation:*

$$\hat{\Sigma}=\left(\sum_{i=1}^{v}Z_iZ_i'\right)^{-1}$$

Step 3 (continued) With the parameters of inverse Wishart distribution in hand ($\bar{\Sigma}=\bar{S}+\left(Y_t-X_tB^1\right)'\left(Y_t-X_tB^1\right)$ and $T+\alpha$) one can use algorithm 3 to draw Σ from the inverse Wishart distribution.

Repeat Steps 2 to 3 M times to obtain $B^1...B^M$ and $(\Sigma)^1 \ldots (\Sigma)^M$. The last H values of B and Σ from these iterations is used to form the empirical distribution of these parameters. Note that the draws of the model parameters (after the burn-in period) are typically used to calculate forecasts or impulse response functions and build the distribution for these statistics of interest.

In general, the Gibbs sampling algorithm for VARs is very similar to that employed for the linear regression model in the previous chapter. The key difference turns out to be the fact that setting up the prior in the VAR model is a more structured process than the linear regression case.

We now turn to a few key prior distributions for VARs that have been proposed in the literature and the implementation of the Gibbs sampling algorithm in Matlab. To discuss the form of the priors we will use the following bi-variate VAR(2) model as an example:

$$\begin{pmatrix} y_t \\ x_t \end{pmatrix}=\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}+\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}\begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix}+\begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix}\begin{pmatrix} y_{t-2} \\ x_{t-2} \end{pmatrix}+\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{1.11}$$

where $var\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}=\Sigma=\begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{12} & \Sigma_{22} \end{pmatrix}$

2. The Minnesota prior

The Minnesota prior (named after its origins at the Federal Reserve Bank of Minnesota) incorporates the prior belief that the endogenous variables included in the VAR follow a random walk process or an AR(1) process. In other words, the *mean* of the Minnesota prior for the VAR coefficients in equation 1.11 implies the following form for the VAR

$$\begin{pmatrix} y_t \\ x_t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \begin{pmatrix} b_{11}^0 & 0 \\ 0 & b_{22}^0 \end{pmatrix} \begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} y_{t-2} \\ x_{t-2} \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{2.1}$$

Equation 2.1 states that the Minnesota prior incorporates the belief that both y_t and x_t follow an AR(1) process or a random walk if $b_{11}^0 = b_{22}^0 = 1$. If y_t and x_t are stationary variables then it may be more realistic to incorporate the prior that they follow an AR(1) process. For this example, the mean of the Minnesota prior distribution for the VAR coefficients (i.e. $\tilde{b}_0$ from $p(b)\tilde{}N\left(\tilde{b}_0, H\right)$) is given by the vector

$$\tilde{b}_0 = \begin{pmatrix} 0 \\ b_{11}^0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ b_{22}^0 \\ 0 \\ 0 \end{pmatrix} \tag{2.2}$$

where the first five rows correspond to the coefficients for the first equation and the second five rows correspond to the coefficients for the second equation. The variance of the prior H is a set in a more structured manner (as compared to the examples in chapter 1) and is given by the following relations for the VAR coefficients b_{ij}

$$\begin{aligned} \left(\frac{\lambda_1}{l^{\lambda_3}}\right)^2 \quad if\ i &= j \\ \left(\frac{\sigma_i\lambda_1\lambda_2}{\sigma_j l^{\lambda_3}}\right)^2 \quad if\ i &\neq j \end{aligned} \tag{2.3}$$

$$(\sigma_1\lambda_4)^2 \text{ for the constant}$$

where i refers to the dependent variable in the i^{th} equation and j to the independent variables in that equation. Therefore, if $i = j$ then we are referring to the coefficients on the own lags of variable i. σ_i and σ_j are variances of error terms from AR regressions estimated via OLS using the variables in the VAR. The ratio of σ_i and σ_j in the formulas above controls for the possibility that variable i and j may have different scales. Note that l is the lag length. The $\lambda's$ are parameters set by the researcher that control the tightness of the prior:

- λ_1 controls the standard deviation of the prior on own lags. As $\lambda_1 \to 0$ $b_{11}, b_{22} \to b_{11}^0, b_{22}^0$ respectively and all other lags go to zero in our example VAR in equation 1.11.
- λ_2 controls the standard deviation of the prior on lags of variables other than the dependent variable i.e. b_{12}, b_{21} etc. As $\lambda_2 \to 0$ b_{ij}, d_{ij} go to zero. With $\lambda_2 = 1$ there is no distinction between lags of the dependent variable and other variables.
- λ_3 controls the degree to which coefficients on lags higher than 1 are likely to be zero. As λ_3 increases coefficients on higher lags are shrunk to zero more tightly.
- The prior variance on the constant is controlled by λ_4. As $\lambda_4 \to 0$ the constant terms are shrunk to zero.

It is instructive to look at how the prior variance matrix looks for our example VAR(2) in equation 1.11. This is shown below in equation 2.4

$$H=\begin{pmatrix} (\sigma_1\lambda_4)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & (\lambda_1)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2 2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & (\sigma_2\lambda_4)^2 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1\lambda_2}{\sigma_1}\right)^2 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\lambda_1)^2 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1\lambda_2}{\sigma_1 2^{\lambda_3}}\right)^2 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 \end{pmatrix} \tag{2.4}$$

The matrix H in equation 2.4 is a 10×10 matrix, because for this example we have 10 total coefficients in the VAR model. The diagonal elements of the matrix H are the prior variances for each corresponding coefficient. Consider the the first five elements on the main diagonal correspond to the first equation the VAR model and is re-produced in equation 2.5.

$$\begin{pmatrix} (\sigma_1\lambda_4)^2 & 0 & 0 & 0 & 0\\ 0 & (\lambda_1)^2 & 0 & 0 & 0\\ 0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2}\right)^2 & 0 & 0\\ 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 & 0\\ 0 & 0 & 0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2 2^{\lambda_3}}\right)^2 \end{pmatrix} \tag{2.5}$$

The first diagonal element $(\sigma_1\lambda_4)^2$ controls the prior on the constant term. The second element $(\lambda_1)^2$ controls the prior on b_{11} the coefficient on the first lag of y_t. Note that this element comes from the first expression in equation 2.3 $\left(\frac{\lambda_1}{l^{\lambda_3}}\right)^2$ with the lag length $l = 1$ as we are dealing with the first lag. The third diagonal element controls the prior on b_{12} the coefficient on the first lag of x_t in the equation for y_t. Note that this element comes from the second expression in equation 2.3 i.e. $\left(\frac{\sigma_i\lambda_1\lambda_2}{\sigma_j l^{\lambda_3}}\right)^2$ with $l = 1$. The third and the fourth diagonal elements control the prior on the coefficients d_{11} and d_{12} respectively (and again come from the first and second expression in equation 2.3 with $l = 2$).

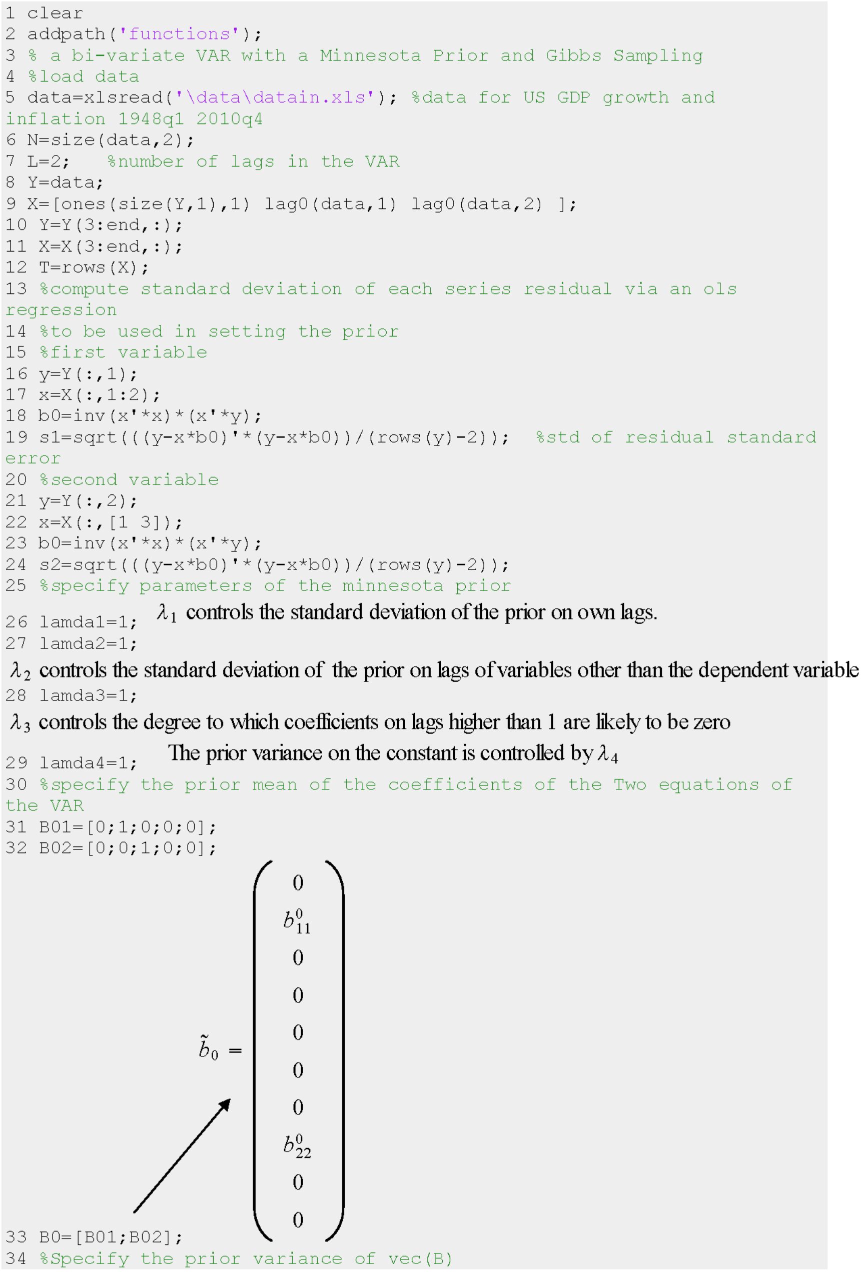
Under a strict interpretation of the Minnesota prior, the covariance matrix of the residuals of the VAR Σ is assumed to be diagonal with the diagonal entries fixed using the error variances from AR regressions σ_i. Under this assumption, the mean of the posterior distribution for the coefficients is available in closed form. For the exact formula, see Kadiyala and Karlsson (1997) Table 1. However, it is common practice amongst some researchers to incorporate the Minnesota prior into the Gibbs sampling framework and draw Σ from the inverse Wishart distribution. We turn to the practical implementation of this algorithm next.

An important question concerns the values of the hyperparameters that control the priors. Canova (2007) pp 380 reports the following values for these parameters typically used in the literature.

$$\begin{aligned} \lambda_1 &= 0.2\\ \lambda_2 &= 0.5\\ \lambda_3 &= 1 \textit{ or } 2\\ \lambda_4 &= 10^5 \end{aligned}$$

Some researchers set the value of these parameters by comparing forecast performance of the VAR across a range of values for these parameters. In addition, the marginal likelihood can be used to select the value of these hyperparameters. The appendix to this chapter shows how to use the procedure in Chib (1995) to calculate the marginal likelihood for a VAR model.

2.1. Gibbs sampling and the Minnesota prior. Matlab code. We consider the estimation of a bi-variate VAR(2) model using quarterly data on annual GDP growth and CPI inflation for the US from 1948Q2 to 2010Q4. We employ a Minnesota prior which incorporates the belief that both variables follow a random walk. Note that while annual CPI inflation may be non-stationary (and hence the random walk prior reasonable), annual GDP growth is likely to be less persistent. Hence one may want to consider incorporating the belief that this variable follows an AR(1) process in actual applications. Note that, we also incorporate a inverse Wishart prior for the covariance matrix and hence depart from the strict form of this model where the covariance matrix is fixed and diagonal. The



```
1 clear
2 addpath('functions');
3 % a bi-variate VAR with a Minnesota Prior and Gibbs Sampling
4 %load data
5 data=xlsread('\data\datain.xls'); %data for US GDP growth and
inflation 1948q1 2010q4
6 N=size(data,2);
7 L=2;   %number of lags in the VAR
8 Y=data;
9 X=[ones(size(Y,1),1) lag0(data,1) lag0(data,2) ];
10 Y=Y(3:end,:);
11 X=X(3:end,:);
12 T=rows(X);
13 %compute standard deviation of each series residual via an ols
regression
14 %to be used in setting the prior
15 %first variable
16 y=Y(:,1);
17 x=X(:,1:2);
18 b0=inv(x'*x)*(x'*y);
19 s1=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));  %std of residual standard
error
20 %second variable
21 y=Y(:,2);
22 x=X(:,[1 3]);
23 b0=inv(x'*x)*(x'*y);
24 s2=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
25 %specify parameters of the minnesota prior
26 lamda1=1;
```

λ_1 controls the standard deviation of the prior on own lags.

```
27 lamda2=1;
```

λ_2 controls the standard deviation of the prior on lags of variables other than the dependent variable

```
28 lamda3=1;
```

λ_3 controls the degree to which coefficients on lags higher than 1 are likely to be zero

```
29 lamda4=1;
```

The prior variance on the constant is controlled by λ_4

```
30 %specify the prior mean of the coefficients of the Two equations of
the VAR
31 B01=[0;1;0;0;0];
32 B02=[0;0;1;0;0];
```

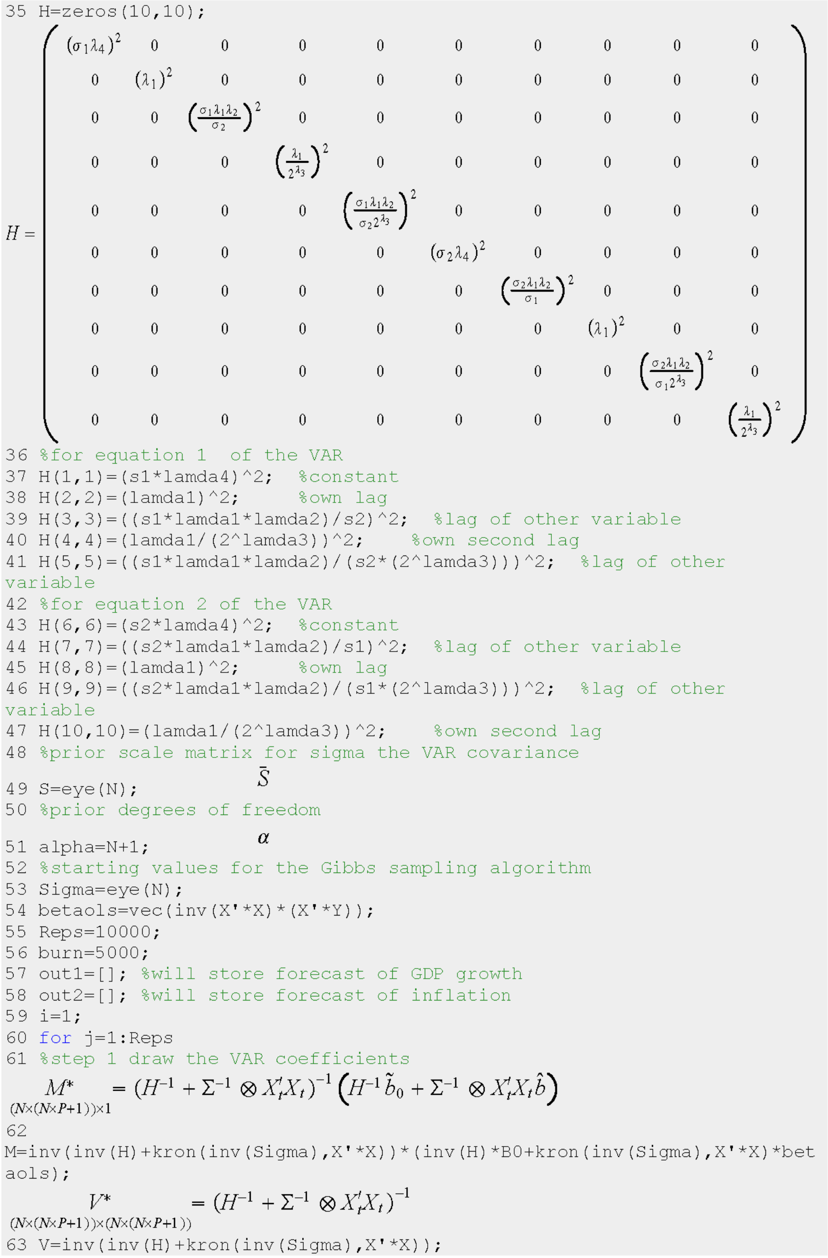
$$\tilde{b}_0 = \begin{pmatrix} 0 \\ b_{11}^0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ b_{22}^0 \\ 0 \\ 0 \end{pmatrix}$$

```
33 B0=[B01;B02];
34 %Specify the prior variance of vec(B)
```

Figure 1. Matlab code for example 1

model is estimated using the Gibbs sampling algorithm described in section 1.1. The code for this model is in the file example1.m in the subfolder chapter 2 under the folder code. The code is also shown in figures 1, 2 and 3. We now go through this code line by line.

Line 5 of the code loads the data for the two variables from an excel file and lines 8 and 9 prepare the matrices Y_t, X_t.Lines 16 to 24 compute σ_1 and σ_2 (to be used to form the Minnesota prior) using AR(1) regressions for each variable. In this example we use the full sample to compute these regressions. Some researchers use a pre-sample (or a training sample) to compute σ_1 and σ_2 and then estimate the VAR on the remaining data points. The argument for using a pre-sample is that the full sample should not really be used to set parameters that affect the prior.



```
35 H=zeros(10,10);
```

$$H=\begin{pmatrix}
(\sigma_1\lambda_4)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & (\lambda_1)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & \left(\frac{\sigma_1\lambda_1\lambda_2}{\sigma_2 2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & (\sigma_2\lambda_4)^2 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1\lambda_2}{\sigma_1}\right)^2 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & (\lambda_1)^2 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1\lambda_2}{\sigma_1 2^{\lambda_3}}\right)^2 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2
\end{pmatrix}$$

```
36 %for equation 1  of the VAR
37 H(1,1)=(s1*lamda4)^2;  %constant
38 H(2,2)=(lamda1)^2;     %own lag
39 H(3,3)=((s1*lamda1*lamda2)/s2)^2;  %lag of other variable
40 H(4,4)=(lamda1/(2^lamda3))^2;    %own second lag
41 H(5,5)=((s1*lamda1*lamda2)/(s2*(2^lamda3)))^2;  %lag of other
variable
42 %for equation 2 of the VAR
43 H(6,6)=(s2*lamda4)^2;  %constant
44 H(7,7)=((s2*lamda1*lamda2)/s1)^2;  %lag of other variable
45 H(8,8)=(lamda1)^2;     %own lag
46 H(9,9)=((s2*lamda1*lamda2)/(s1*(2^lamda3)))^2;  %lag of other
variable
47 H(10,10)=(lamda1/(2^lamda3))^2;    %own second lag
48 %prior scale matrix for sigma the VAR covariance
```

$$\bar{S}$$

```
49 S=eye(N);
50 %prior degrees of freedom
```

$$\alpha$$

```
51 alpha=N+1;
52 %starting values for the Gibbs sampling algorithm
53 Sigma=eye(N);
54 betaols=vec(inv(X'*X)*(X'*Y));
55 Reps=10000;
56 burn=5000;
57 out1=[]; %will store forecast of GDP growth
58 out2=[]; %will store forecast of inflation
59 i=1;
60 for j=1:Reps
61 %step 1 draw the VAR coefficients
```

$$\underset{(N\times(N\times P+1))\times 1}{M^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t'X_t\right)^{-1}\left(H^{-1}\tilde{b}_0+\Sigma^{-1}\otimes X_t'X_t\hat{b}\right)$$

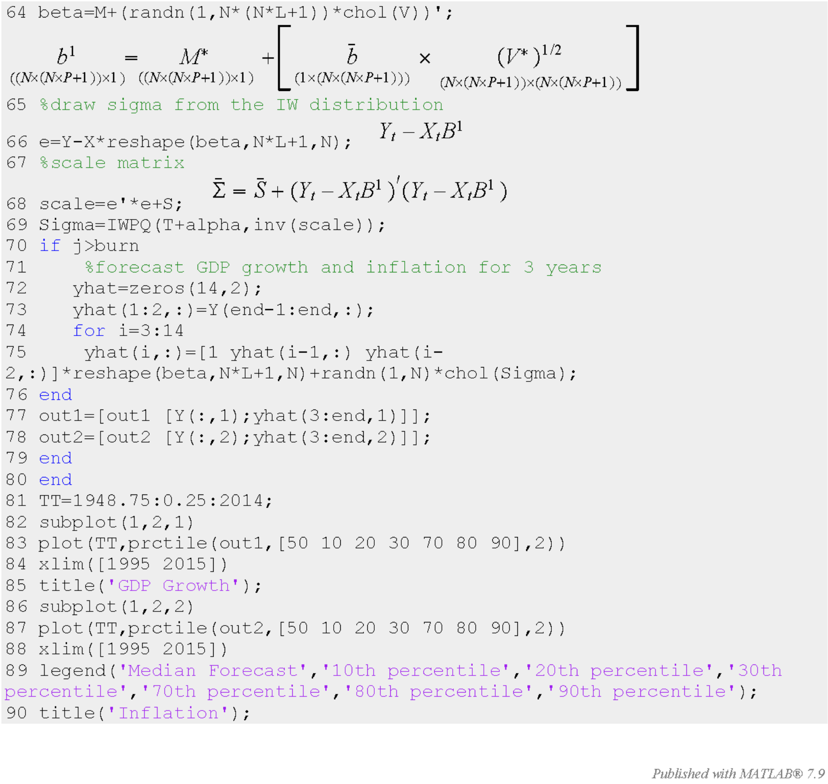
```
62
M=inv(inv(H)+kron(inv(Sigma),X'*X))*(inv(H)*B0+kron(inv(Sigma),X'*X)*bet
aols);
```

$$\underset{(N\times(N\times P+1))\times(N\times(N\times P+1))}{V^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t'X_t\right)^{-1}$$

```
63 V=inv(inv(H)+kron(inv(Sigma),X'*X));
```

FIGURE 2. Matlab code for example 1 (continued)

Lines 27 to 29 specify the parameters λ_1,λ_2, λ_3,λ_4 that control the tightness of the prior and are used to build the prior covariance. Line 33 specifies $\tilde{b}_0$ the prior mean. As mentioned above in this example we simply assume a prior mean of 1 for the coefficients on own first lags. In practice, this choice should depend on the stationarity properties of the series. Line 35 forms the 10×10 prior variance matrix H. Lines 37 to 47 fill the diagonal elements of this matrix as shown in equation 2.4. Line 49 specifies the prior scale matrix for the inverse Wishart distribution as an identity matrix but specifies the prior degrees of freedom as the minimum possible $N+1$ (line 51) hence making this a non-informative prior. Line 53 sets the starting value for Σ as an identity matrix. We use 10,000 Gibbs replications discarding the first 5000 as burn-in. Line 62 is the first step of the Gibbs sampler with the calculation of the mean of the conditional



```
64 beta=M+(randn(1,N*(N*L+1))*chol(V))';
```

$$\underset{((N\times(N\times P+1))\times 1)}{b^1} = \underset{((N\times(N\times P+1))\times 1)}{M^*} + \left[\underset{(1\times(N\times(N\times P+1)))}{\bar{b}} \times \underset{(N\times(N\times P+1))\times(N\times(N\times P+1))}{(V^*)^{1/2}}\right]$$

```
65 %draw sigma from the IW distribution
66 e=Y-X*reshape(beta,N*L+1,N);
```

$$Y_t - X_t B^1$$

```
67 %scale matrix
68 scale=e'*e+S;
```

$$\bar{\Sigma} = \bar{S} + (Y_t - X_t B^1)'(Y_t - X_t B^1)$$

```
69 Sigma=IWPQ(T+alpha,inv(scale));
70 if j>burn
71     %forecast GDP growth and inflation for 3 years
72    yhat=zeros(14,2);
73    yhat(1:2,:)=Y(end-1:end,:);
74    for i=3:14
75     yhat(i,:)=[1 yhat(i-1,:) yhat(i-2,:)]*reshape(beta,N*L+1,N)+randn(1,N)*chol(Sigma);
76 end
77 out1=[out1 [Y(:,1);yhat(3:end,1)]];
78 out2=[out2 [Y(:,2);yhat(3:end,2)]];
79 end
80 end
81 TT=1948.75:0.25:2014;
82 subplot(1,2,1)
83 plot(TT,prctile(out1,[50 10 20 30 70 80 90],2))
84 xlim([1995 2015])
85 title('GDP Growth');
86 subplot(1,2,2)
87 plot(TT,prctile(out2,[50 10 20 30 70 80 90],2))
88 xlim([1995 2015])
89 legend('Median Forecast','10th percentile','20th percentile','30th percentile','70th percentile','80th percentile','90th percentile');
90 title('Inflation');
```

Published with MATLAB® 7.9

FIGURE 3. Matlab code: Example 1 continued

posterior distribution of the VAR coefficients $\underset{(N\times(N\times P+1))\times 1}{M^*} = \left(H^{-1} + \Sigma^{-1} \otimes X_t'X_t\right)^{-1}\left(H^{-1}\tilde{b}_0 + \Sigma^{-1} \otimes X_t'X_t\hat{b}\right)$ while line 63 compute the variance of this distribution as $\underset{(N\times(N\times P+1))\times(N\times(N\times P+1))}{V^*} = \left(H^{-1} + \Sigma^{-1} \otimes X_t'X_t\right)^{-1}$. On line 64 we draw the VAR coefficients from the normal distribution using M^* and V^*. Line 66 calculates the residuals of the VAR. Line 68 calculates the posterior scale matrix $\bar{\Sigma}$. Line 69 draws the covariance matrix from the inverse Wishart distribution where the function IWPQ uses the method in algorithm 3. Once past the burn-in period we build up the predictive density and save the forecast for each variable. The quantiles of the predictive density are shown in figure 4

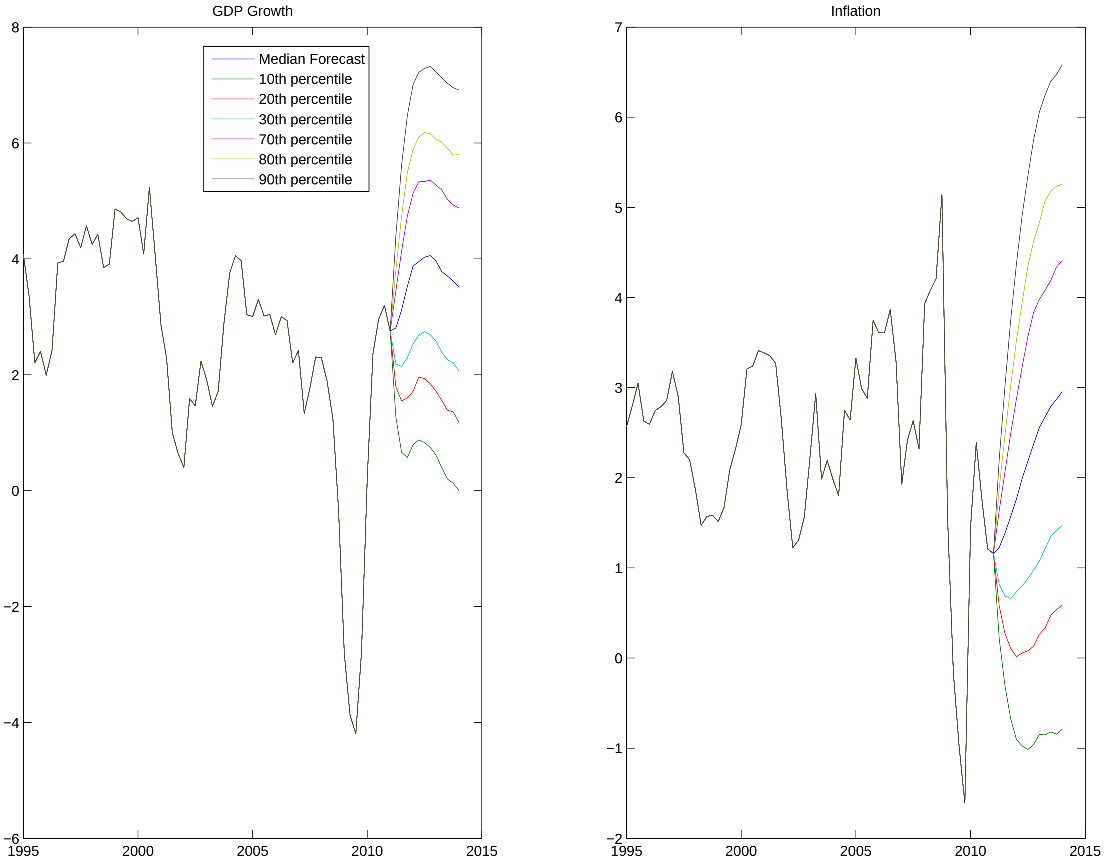


FIGURE 4. Forecast for annual GDP growth and inflation using a VAR with a Minnesota prior

3. The Normal inverse Wishart Prior

3.1. The natural conjugate prior. The normal inverse Wishart prior assumes a normal prior for the VAR coefficients and a inverse Wishart prior for the covariance matrix. This is a conjugate prior for the VAR model. This prior for the VAR parameters can be specified as follows

$$p\left(b|\Sigma\right) \tilde{} N\left(\tilde{b}_0, \Sigma \otimes \bar{H}\right) \tag{3.1}$$

$$p(\Sigma)\tilde{}IW(\bar{S}, \alpha) \tag{3.2}$$

where $\tilde{b}_0$ is specified exactly as in equation 2.1. The matrix $\bar{H}$ is a diagonal matrix where the diagonal elements are defined as

$$\left(\frac{\lambda_0\lambda_1}{l^{\lambda_3}\sigma_i}\right)^2 \text{ for the coefficients on lags} \tag{3.3}$$

$$\left(\lambda_0\lambda_4\right)^2 \text{ for the constant} \tag{3.4}$$

So, for our example VAR(2), this matrix is given as

$$\bar{H} = \begin{pmatrix} (\lambda_0\lambda_4)^2 & 0 & 0 & 0 & 0 \\ 0 & \left(\frac{\lambda_0\lambda_1}{\sigma_1}\right)^2 & 0 & 0 & 0 \\ 0 & 0 & \left(\frac{\lambda_0\lambda_1}{\sigma_2}\right)^2 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{\lambda_0\lambda_1}{2^{\lambda_3}\sigma_1}\right)^2 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{\lambda_0\lambda_1}{2^{\lambda_3}\sigma_2}\right)^2 \end{pmatrix} \tag{3.5}$$

The matrix $\bar{S}$ is defined as a $N \times N$ diagonal matrix with diagonal elements given by

$$\left(\frac{\sigma_i}{\lambda_0}\right)^2 \tag{3.6}$$

For our example VAR this matrix is given by

$$\bar{S} = \begin{pmatrix} \left(\frac{\sigma_1}{\lambda_0}\right)^2 & 0 \\ 0 & \left(\frac{\sigma_2}{\lambda_0}\right)^2 \end{pmatrix} \tag{3.7}$$

The parameters that make up the diagonal elements of $\bar{H}$ and $\bar{S}$ have the following interpretation:

- λ_0 controls the overall tightness of the prior on the covariance matrix.
- λ_1 controls the tightness of the prior on the coefficients on the first lag. As $\lambda_1 \to 0$ the prior is imposed more tightly.
- λ_3 controls the degree to which coefficients on lags higher than 1 are likely to be zero. As λ_3 increases coefficients on higher lags are shrunk to zero more tightly.
- The prior variance on the constant is controlled by λ_4. As $\lambda_4 \to 0$ the constant is shrunk to zero.

To consider the interpretation of this prior (i.e. equations 3.1 and 3.2), consider calculating the prior covariance matrix for the coefficients. This will involve the following operation

$$\bar{S} \otimes \bar{H} \tag{3.8}$$

That is the matrix H or the prior variance of all the VAR coefficients is obtained by a kronecker product in 3.8. Consider calculating this kronecker product in our bi-variate VAR example

$$\begin{pmatrix} \left(\frac{\sigma_1}{\lambda_0}\right)^2 & 0 \\ 0 & \left(\frac{\sigma_2}{\lambda_0}\right)^2 \end{pmatrix} \otimes \begin{pmatrix} (\lambda_0\lambda_4)^2 & 0 & 0 & 0 & 0 \\ 0 & \left(\frac{\lambda_0\lambda_1}{\sigma_1}\right)^2 & 0 & 0 & 0 \\ 0 & 0 & \left(\frac{\lambda_0\lambda_1}{\sigma_2}\right)^2 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{\lambda_0\lambda_1}{2^{\lambda_3}\sigma_1}\right)^2 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{\lambda_0\lambda_1}{2^{\lambda_3}\sigma_2}\right)^2 \end{pmatrix}$$

This kronecker product involves each element of $\bar{S}$ being multiplied by the entire $\bar{H}$. If one does one obtains equation 3.9

$$H = \begin{pmatrix} (\sigma_1\lambda_4)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (\lambda_1)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \left(\frac{\sigma_1\lambda_1}{\sigma_2}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \left(\frac{\sigma_1\lambda_1}{\sigma_2 2^{\lambda_3}}\right)^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & (\sigma_2\lambda_4)^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1}{\sigma_1}\right)^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & (\lambda_1)^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\sigma_2\lambda_1}{\sigma_1 2^{\lambda_3}}\right)^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \left(\frac{\lambda_1}{2^{\lambda_3}}\right)^2 \end{pmatrix} \tag{3.9}$$

Note that this is just the Minnesota prior variance with the parameter $\lambda_2 = 1$. Therefore the structure of the natural conjugate prior implies that we treat lags of dependent variable and lags of other variables in each equation of the VAR in exactly the same manner. This is in contrast to the Minnesota prior where the parameter λ_2 governs the tightness of the prior on lags of variables other than the dependent variable.

Given the natural conjugate prior, analytical results exist for the posterior distribution for the coefficients and the covariance matrix. Therefore one clear advantage of this set up over the Minnesota prior is that it allows the derivation of these analytical results without the need for a fixed and diagonal error covariance matrix. The exact formulas for the posteriors are listed in table 1 in Kadiyala and Karlsson (1997).

The Gibbs sampling algorithm for this model is identical to that described in section 2.1. As explained above the only difference is that the variance of the prior distribution is set equalt to H as described in equation 3.9.

3.2. The independent Normal inverse Wishart prior. The restrictions inherent in the natural conjugate prior may be restrictive in many practical circumstances. That is, in many practical applications one may want to treat the coefficients of the lagged dependent variables differently from those of other variables. An example is a situation where the researcher wants impose that some coefficients in a VAR equation are close to zero (e.g. to impose money neutrality or small open economy type restrictions). This can be acheived via the independent Normal inverse Wishart prior. As the name suggests, this prior involves setting the prior for the VAR coefficients and the error covariance independently (unlike the natural conjugate prior)

$$p(b)\,\tilde{}N\left(\tilde{b}_0, H\right) \tag{3.10}$$

$$p(\Sigma)\tilde{}IW(\bar{S}, \alpha) \tag{3.11}$$

where the elements of $\tilde{b}_0$, H and $\bar{S}$ are set by the researcher to suit the empirical question at hand. Under this prior analytical expressions for the marginal posterior distributions are not available. Therefore, the Gibbs sampling algorithm outlined in section 1.1 has to be used.

As an example, consider estimating the following VAR(2) model for the US,

$$\begin{pmatrix} R_t \\ GB_t \\ U_t \\ P_t \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} \begin{pmatrix} R_{t-1} \\ GB_{t-1} \\ U_{t-1} \\ P_{t-1} \end{pmatrix} \tag{3.12}$$

$$+ \begin{pmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & d_{34} \\ d_{41} & d_{42} & d_{43} & d_{44} \end{pmatrix} \begin{pmatrix} R_{t-2} \\ GB_{t-2} \\ U_{t-2} \\ P_{t-2} \end{pmatrix} + \begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ v_{4t} \end{pmatrix}$$

where

$$var\begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ v_{4t} \end{pmatrix} = \Sigma$$

and R_t is the federal funds rate, GB_t is the 10 year government bond yield, U_t is the unemployment rate and P_t is annual CPI inflation. Suppose that one is interested in estimating the response of these variables to a decrease in the government bond yield. This shock may proxy the impact of quantitative easing polices recently adopted. Note, that given the recession in 2010/2011 it is reasonable to assume that the federal funds rate is unlikely to respond to changes in other variables. The standard way to impose this restriction on the contemporaneous period is to identify the yield shock using a Cholesky decomposition of Σ

$$\Sigma = A_0 A_0'$$

where A_0 is a lower triangular matrix. Note, however, that one may also want impose the restriction that the Federal Funds rate does not respond with a lag to changes in the other variables. Given that the Federal Funds rate is near the zero lower bound during the crisis period this restriction can be justified.

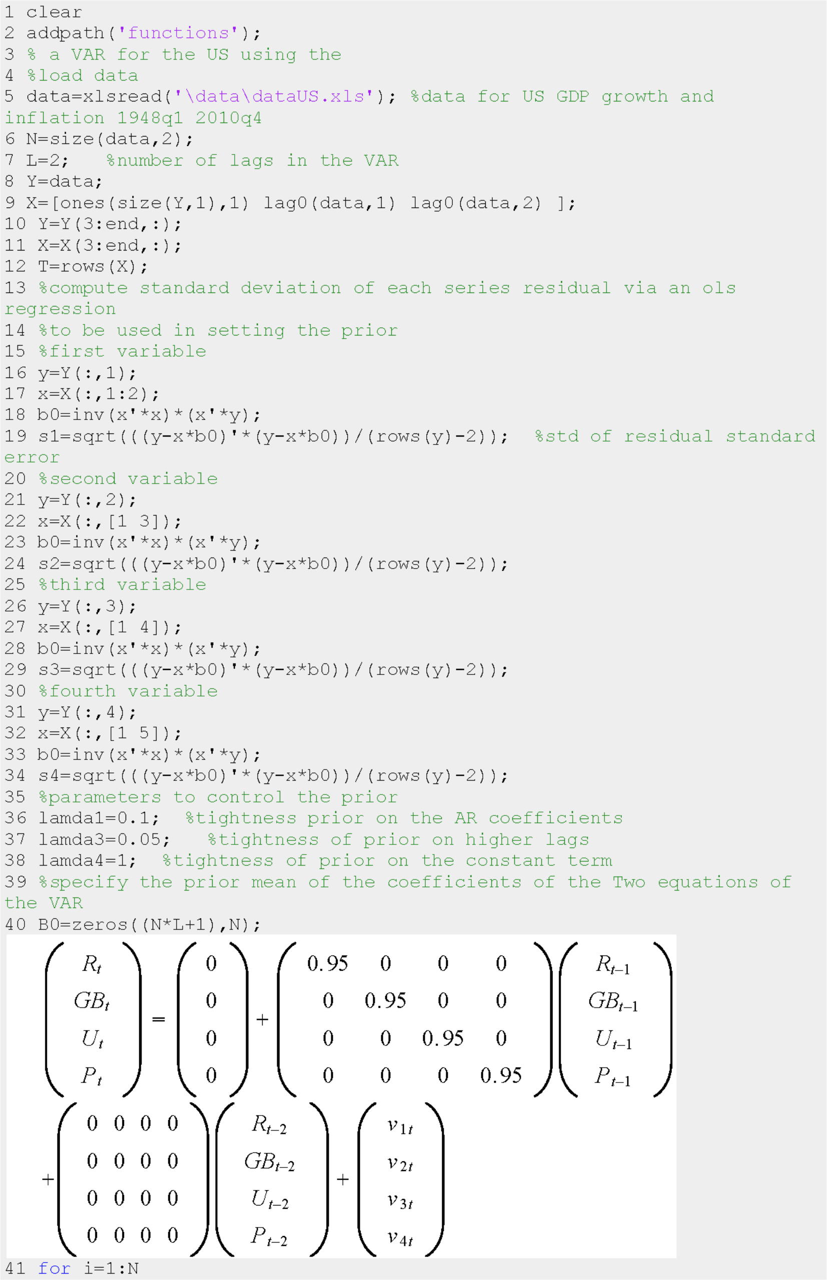
The independent Normal Wishart prior offers a convenient way to incorporate these restrictions into the VAR model. One can specify the prior mean for all coefficients equal to zero i.e. $\tilde{b}_0 = 0_{(N\times(N\times P+1))\times 1}$ and the covariance of this prior H as a diagonal marix with diagonal elements equal to a very large number *except for the elements corresponding to the coefficients* b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14}. The elements of H corresponding to these coefficients are instead set to a very small number and the prior mean of zero is imposed very tightly for them. Therefore the posterior estimates of b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14} will be very close to zero. We now turn to a matlab implementation of this example using Gibbs sampling.

3.2.1. *Gibbs sampling and the independent normal Wishart prior.* We estimate the VAR model in equation 3.12 using data for the US over the period 2007m1 to 2010m12, the period associated with the financial crisis. We employ a prior which sets the coefficients b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14} close to zero– i.e. the prior mean for these equals zero and the prior variance is a very small number. Given the very short sample period, we also set a prior for the remaining VAR coefficients. For these remaining coefficients, we assume that the prior mean for coefficients on own first lags are equal to 0.95 and all others equal zero. The prior variance for these is set according to equation 3.9. We set a prior independently for error covariance. We use a Gibbs sampling algorithm to approximate the posterior. The matlab code (example2.m) can be seen in figures 5, 6 and 7.

Lines 16 to 34 of the code calculate σ_1, σ_2, σ_3, σ_4 the variances used to scale the prior variance for the VAR coefficients other than b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14}. Lines 36 to 38 specify the parameters that will control the variance of the prior on these parameters. Lines 40 to 44 set the prior mean for the VAR coefficients. Under the prior the VAR has the following form:

$$\begin{pmatrix} R_t \\ GB_t \\ U_t \\ P_t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.95 & 0 & 0 & 0 \\ 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0 \\ 0 & 0 & 0 & 0.95 \end{pmatrix} \begin{pmatrix} R_{t-1} \\ GB_{t-1} \\ U_{t-1} \\ P_{t-1} \end{pmatrix}$$

$$+ \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} R_{t-2} \\ GB_{t-2} \\ U_{t-2} \\ P_{t-2} \end{pmatrix} + \begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ v_{4t} \end{pmatrix}$$



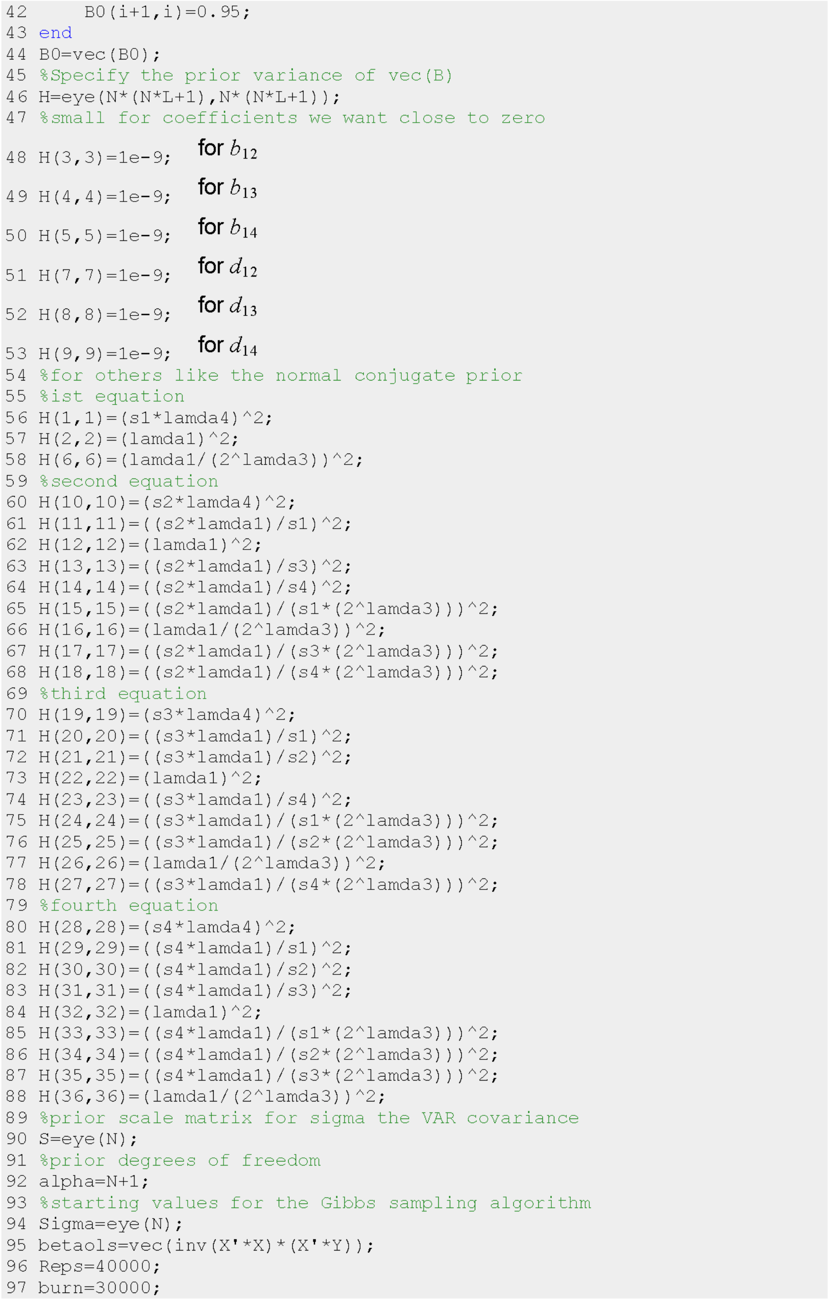
```
1 clear
2 addpath('functions');
3 % a VAR for the US using the
4 %load data
5 data=xlsread('\data\dataUS.xls'); %data for US GDP growth and
inflation 1948q1 2010q4
6 N=size(data,2);
7 L=2;   %number of lags in the VAR
8 Y=data;
9 X=[ones(size(Y,1),1) lag0(data,1) lag0(data,2) ];
10 Y=Y(3:end,:);
11 X=X(3:end,:);
12 T=rows(X);
13 %compute standard deviation of each series residual via an ols
regression
14 %to be used in setting the prior
15 %first variable
16 y=Y(:,1);
17 x=X(:,1:2);
18 b0=inv(x'*x)*(x'*y);
19 s1=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));  %std of residual standard
error
20 %second variable
21 y=Y(:,2);
22 x=X(:,[1 3]);
23 b0=inv(x'*x)*(x'*y);
24 s2=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
25 %third variable
26 y=Y(:,3);
27 x=X(:,[1 4]);
28 b0=inv(x'*x)*(x'*y);
29 s3=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
30 %fourth variable
31 y=Y(:,4);
32 x=X(:,[1 5]);
33 b0=inv(x'*x)*(x'*y);
34 s4=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
35 %parameters to control the prior
36 lamda1=0.1;  %tightness prior on the AR coefficients
37 lamda3=0.05;   %tightness of prior on higher lags
38 lamda4=1;  %tightness of prior on the constant term
39 %specify the prior mean of the coefficients of the Two equations of
the VAR
40 B0=zeros((N*L+1),N);
```

$$\begin{pmatrix} R_t \\ GB_t \\ U_t \\ P_t \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0.95 & 0 & 0 & 0 \\ 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0.95 & 0 \\ 0 & 0 & 0 & 0.95 \end{pmatrix} \begin{pmatrix} R_{t-1} \\ GB_{t-1} \\ U_{t-1} \\ P_{t-1} \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} R_{t-2} \\ GB_{t-2} \\ U_{t-2} \\ P_{t-2} \end{pmatrix} + \begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ v_{4t} \end{pmatrix}$$

```
41 for i=1:N
```

FIGURE 5. Matlab code for example 2

Lines 48 to 53 set the variance around the prior for b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14}. Note that the variance is set to a very small number implying that we incorporate the belief that these coefficients equal zero very strongly. Lines 56 to 88 set the prior variance for the remaining VAR coefficients according to equation 3.9. This is an ad hoc way of incorporating prior information about these coefficients but is important given the small sample. Lines 90 and 92 set the prior for the error covariance as in example 1. Given these priors the Gibbs algorithm is exactly the same as in the previous example. However, we incorporate one change usually adopted by researchers. On lines 108 to 115 we draw the VAR coefficients from its conditional posterior but ensure that the draw is stable. In other words the function stability re-writes the VAR coefficient matrix in companion form and checks if the eigenvalues of this

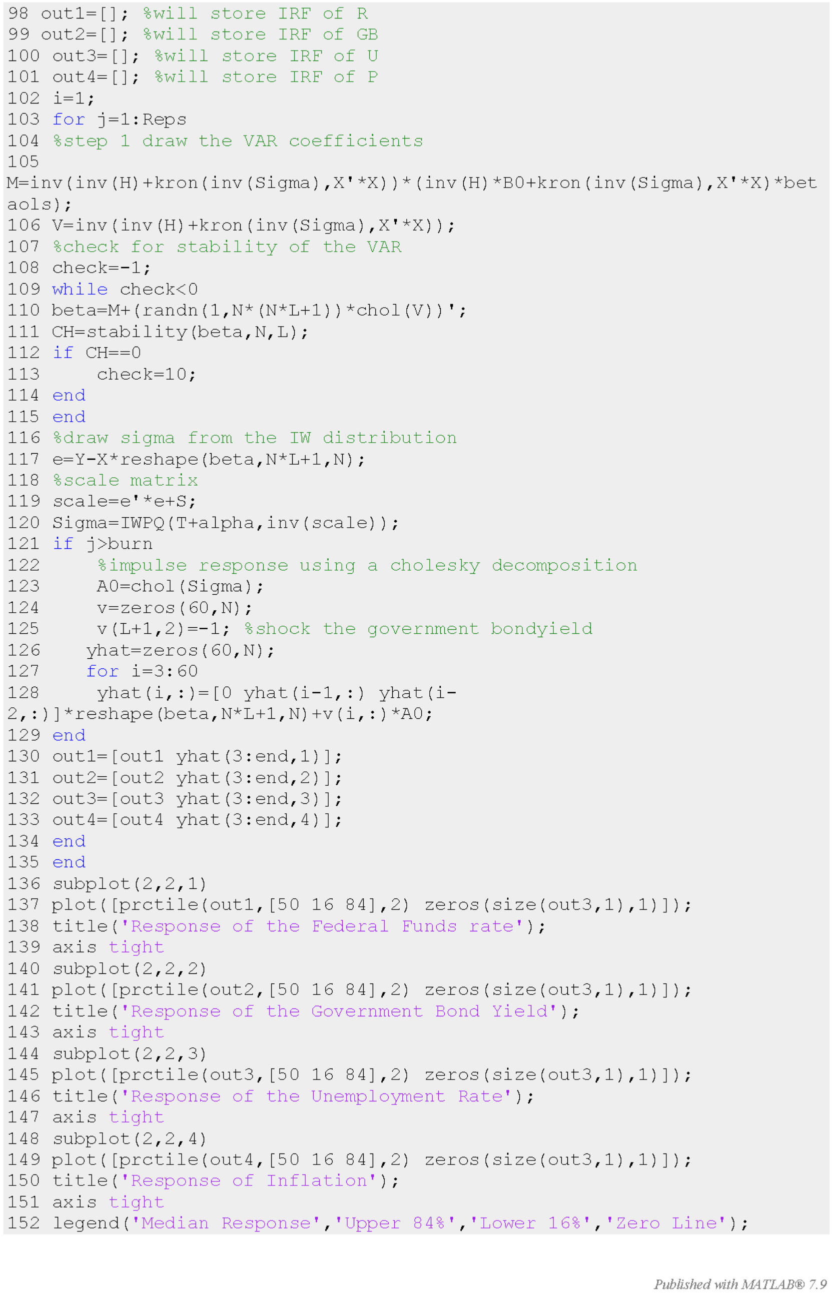


```
42     B0(i+1,i)=0.95;
43 end
44 B0=vec(B0);
45 %Specify the prior variance of vec(B)
46 H=eye(N*(N*L+1),N*(N*L+1));
47 %small for coefficients we want close to zero
48 H(3,3)=1e-9;    for b12
49 H(4,4)=1e-9;    for b13
50 H(5,5)=1e-9;    for b14
51 H(7,7)=1e-9;    for d12
52 H(8,8)=1e-9;    for d13
53 H(9,9)=1e-9;    for d14
54 %for others like the normal conjugate prior
55 %ist equation
56 H(1,1)=(s1*lamda4)^2;
57 H(2,2)=(lamda1)^2;
58 H(6,6)=(lamda1/(2^lamda3))^2;
59 %second equation
60 H(10,10)=(s2*lamda4)^2;
61 H(11,11)=((s2*lamda1)/s1)^2;
62 H(12,12)=(lamda1)^2;
63 H(13,13)=((s2*lamda1)/s3)^2;
64 H(14,14)=((s2*lamda1)/s4)^2;
65 H(15,15)=((s2*lamda1)/(s1*(2^lamda3)))^2;
66 H(16,16)=(lamda1/(2^lamda3))^2;
67 H(17,17)=((s2*lamda1)/(s3*(2^lamda3)))^2;
68 H(18,18)=((s2*lamda1)/(s4*(2^lamda3)))^2;
69 %third equation
70 H(19,19)=(s3*lamda4)^2;
71 H(20,20)=((s3*lamda1)/s1)^2;
72 H(21,21)=((s3*lamda1)/s2)^2;
73 H(22,22)=(lamda1)^2;
74 H(23,23)=((s3*lamda1)/s4)^2;
75 H(24,24)=((s3*lamda1)/(s1*(2^lamda3)))^2;
76 H(25,25)=((s3*lamda1)/(s2*(2^lamda3)))^2;
77 H(26,26)=(lamda1/(2^lamda3))^2;
78 H(27,27)=((s3*lamda1)/(s4*(2^lamda3)))^2;
79 %fourth equation
80 H(28,28)=(s4*lamda4)^2;
81 H(29,29)=((s4*lamda1)/s1)^2;
82 H(30,30)=((s4*lamda1)/s2)^2;
83 H(31,31)=((s4*lamda1)/s3)^2;
84 H(32,32)=(lamda1)^2;
85 H(33,33)=((s4*lamda1)/(s1*(2^lamda3)))^2;
86 H(34,34)=((s4*lamda1)/(s2*(2^lamda3)))^2;
87 H(35,35)=((s4*lamda1)/(s3*(2^lamda3)))^2;
88 H(36,36)=(lamda1/(2^lamda3))^2;
89 %prior scale matrix for sigma the VAR covariance
90 S=eye(N);
91 %prior degrees of freedom
92 alpha=N+1;
93 %starting values for the Gibbs sampling algorithm
94 Sigma=eye(N);
95 betaols=vec(inv(X'*X)*(X'*Y));
96 Reps=40000;
97 burn=30000;
```

FIGURE 6. example 2: Matlab code continued

matrix are less than or equal to1–i.e. that the VAR is stable (see Hamilton (1994) page 259). Once past the burn-in stage line 123 calculates the structural impact matrix A_0 as the Cholesky decomposition of the draw of Σ and lines 124 to 129 calculate the impulse response to a negative shock in the Government bond yield using this A_0. We save the impulse response functions for each remaining draw of the Gibbs sampler. Quantiles of the saved draws of the impulse response are error bands for the impulse responses.

The resulting median impulse responses and the 68% error bands are shown in figure 8. Note that 68% error bands are typically shown as the 90% or 95% bands can be misleading if the distribution of the impulse response function is skewed due to non-linearity. The response of the Federal Funds rate to this shock is close to zero as



```
out1=[]; %will store IRF of R
out2=[]; %will store IRF of GB
out3=[]; %will store IRF of U
out4=[]; %will store IRF of P
i=1;
for j=1:Reps
%step 1 draw the VAR coefficients
M=inv(inv(H)+kron(inv(Sigma),X'*X))*(inv(H)*B0+kron(inv(Sigma),X'*X)*betaols);
V=inv(inv(H)+kron(inv(Sigma),X'*X));
%check for stability of the VAR
check=-1;
while check<0
beta=M+(randn(1,N*(N*L+1))*chol(V))';
CH=stability(beta,N,L);
if CH==0
    check=10;
end
end
%draw sigma from the IW distribution
e=Y-X*reshape(beta,N*L+1,N);
%scale matrix
scale=e'*e+S;
Sigma=IWPQ(T+alpha,inv(scale));
if j>burn
    %impulse response using a cholesky decomposition
    A0=chol(Sigma);
    v=zeros(60,N);
    v(L+1,2)=-1; %shock the government bondyield
   yhat=zeros(60,N);
   for i=3:60
    yhat(i,:)=[0 yhat(i-1,:) yhat(i-2,:)]*reshape(beta,N*L+1,N)+v(i,:)*A0;
end
out1=[out1 yhat(3:end,1)];
out2=[out2 yhat(3:end,2)];
out3=[out3 yhat(3:end,3)];
out4=[out4 yhat(3:end,4)];
end
end
subplot(2,2,1)
plot([prctile(out1,[50 16 84],2) zeros(size(out3,1),1)]);
title('Response of the Federal Funds rate');
axis tight
subplot(2,2,2)
plot([prctile(out2,[50 16 84],2) zeros(size(out3,1),1)]);
title('Response of the Government Bond Yield');
axis tight
subplot(2,2,3)
plot([prctile(out3,[50 16 84],2) zeros(size(out3,1),1)]);
title('Response of the Unemployment Rate');
axis tight
subplot(2,2,4)
plot([prctile(out4,[50 16 84],2) zeros(size(out3,1),1)]);
title('Response of Inflation');
axis tight
legend('Median Response','Upper 84%','Lower 16%','Zero Line');
```

Published with MATLAB® 7.9

Figure 7. example2: Matlab code (continued)

implied by the Cholesky decomposition and the prior on b_{12}, b_{13}, b_{14} and d_{12}, d_{13}, d_{14}. A 0.3% fall in the Government bond yield lowers unemployment by 0.1% after 10 months (but the impact is quite uncertain as evident from the wide error bands). The impact on inflation is much more imprecise with the zero line within the error bands for most of the impulse horizon.

4. Steady State priors

In some circumstances it is useful to incorporate priors about the long run behaviour of the variables included in the VAR. For example one may be interested in forecasting inflation using a VAR model. It can be argued that

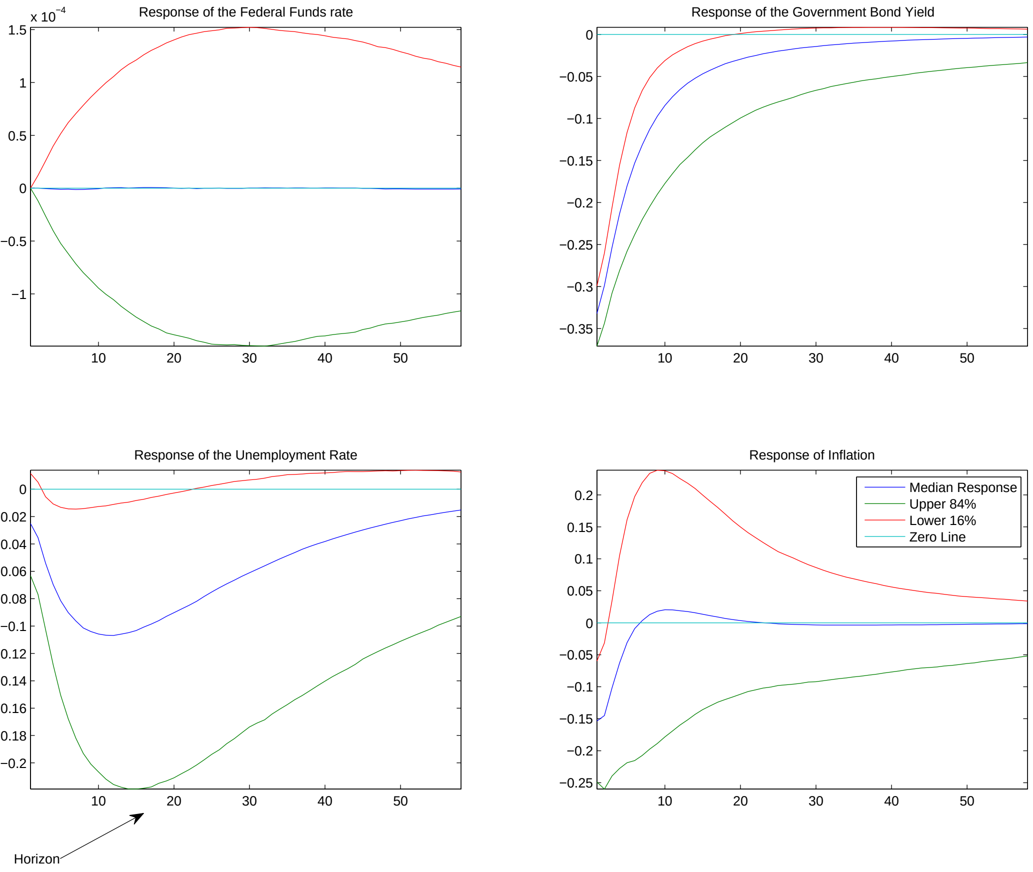


FIGURE 8. Impulse response to a fall in the Government bond yield

inflation in the long run will be close to the target set by the central bank. This information is a potentially useful input as a prior.

Note that while the priors introduced above allow the researcher to have an impact on the value of the constant terms in the VAR, there is no direct way to affect the long run mean (note that forecasts converge to the long run unconditional mean). Consider our example bi-variate VAR re-produced below

$$\begin{pmatrix} y_t \\ x_t \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix} + \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \begin{pmatrix} y_{t-2} \\ x_{t-2} \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, VAR \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \Sigma \tag{4.1}$$

The Minnesota and the Normal inverse Wishart priors place a prior on the constants $\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$. The long run or steady state means for y_t and x_t denoted by μ_1 and μ_2 however, is defined as (see Hamilton (1994) page 258)

$$\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} = \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} - \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \right)^{-1} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \tag{4.2}$$

Villani (2009) proposes a prior distribution for the unconditional means $\mu = \{\mu_1, \mu_2\}$ along with coefficients of the VAR model. This requires one to re-write the model in terms of $\mu = \{\mu_1, \mu_2\}$ rather than the constants c_1 and c_2. This can be done in our example VAR by substituting for c_1 and c_2 in equation 4.1 using the values of these constants from equation 4.2 to obtain

$$\begin{aligned} \begin{pmatrix} y_t \\ x_t \end{pmatrix} &= \left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} - \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \right) \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \\ &+ \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} \begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix} + \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} \begin{pmatrix} y_{t-2} \\ x_{t-2} \end{pmatrix} + \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \end{aligned}$$

or more compactly in terms of lag operators as

$$B(L)(Z_t - \mu) = v \tag{4.3}$$

where $Z_t = \{y_t, x_t\}$, $v = \{v_1, v_2\}$and $B(L) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} L - \begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix} L^2$

Villani (2009) proposes a normal prior for μ

$$p(\mu)\sim N(\mu_0,\Sigma_\mu) \tag{4.4}$$

The priors for the autoregressive coefficients and the error covariance are specified independently. For example, one can specify the Minnesota prior for the autoregressive coefficients and an inverse Wishart prior for the error covariance.

Note that there are three sets of parameters to be estimated in this VAR model: (1) The VAR coefficients, the error covariance and the long run means μ.Villani (2009) describes a Gibbs sampling algorithm to estimate the model and we turn to this next.

4.1. Gibbs sampling algorithm. The Gibbs sampling algorithm for this model is an extension of the algorithm described in section 1.1. Conditional on knowing μ the reparametrised model is a just a standard VAR and standard methods apply. The algorithm works in the following steps

Step 1 Set a normal prior for the VAR coefficients $p(\bar{b})\tilde{}N\left(\tilde{b}_0,H\right)$ where $\bar{b}$ the (vectorised) VAR coefficients except for the constant terms. The prior for the covariance matrix of the residuals Σ is inverse Wishart and given by $IW\left(\bar{S},\alpha\right)$. The prior for the long run means is $p(\mu)\sim N(\mu_0,\Sigma_\mu)$. Set a starting value for μ. A starting value can be set via OLS estimates of the VAR coefficients as

$$\mu_{ols}=\left(I-\tilde{B}\right)^{-1}\hat{C}$$

where $\tilde{B}$ are the OLS estimates of the VAR coefficients in companion form and C denotes the OLS estimates of the constant terms in a comformable matrix. For the bi-variate VAR in equation 4.1 this looks as follows

$$\begin{pmatrix}\mu_{OLS,1}\\ \mu_{OLS,2}\\ \\ \end{pmatrix}=\left(\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}-\begin{pmatrix}\hat{b}_{11}&\hat{b}_{12}&\hat{d}_{11}&\hat{d}_{12}\\ \hat{b}_{21}&\hat{b}_{22}&\hat{d}_{21}&\hat{d}_{22}\\1&0&0&0\\0&1&0&0\end{pmatrix}\right)^{-1}\begin{pmatrix}\hat{c}_1\\ \hat{c}_2\\0\\0\end{pmatrix}$$

Step 2 Sample the VAR coefficients from their conditonal distribution. Conditional on μ, equation 4.3 implies that the model is a VAR in the transformed (or de-meaned) variables $Y_t^0=Z_t-\mu$. The conditional posterior distribution of the VAR coefficients is normal distribution $H\left(\bar{b}|\Sigma,\mu,Z_t^*\right)\tilde{}N\left(M^*,V^*\right)$ where

$$\underset{(N\times(N\times P))\times 1}{M^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t^{0\prime}X_t^0\right)^{-1}\left(H^{-1}\tilde{b}_0+\Sigma^{-1}\otimes X_t^{0\prime}X_t^0\hat{b}\right) \tag{4.5}$$

$$\underset{(N\times(N\times P))\times(N\times(N\times P))}{V^*}=\left(H^{-1}+\Sigma^{-1}\otimes X_t^{0\prime}X_t^0\right)^{-1} \tag{4.6}$$

where $X_t^0=[Y_{t-1}^0,...,Y_{t-p}^0]$ and $\hat{b}=vec\left(\left(X_t^{0\prime}X_t^0\right)^{-1}\left(X_t^{0\prime}Y_t^0\right)\right)$. Note that the dimensions of M^* and V^* are different relative to those shown in section 1.1 because X_t^0 does not contain a constant term. Once M^* and V^* are calculated, the VAR coefficients are drawn from the normal distribution as before.

Step 3 Draw Σ from its conditional distribution $H\left(\Sigma|\bar{b},\mu,Z_t^*\right)\tilde{}IW\left(\bar{\Sigma},T+\alpha\right)$ where $\bar{\Sigma}=\bar{S}+\left(Y_t^0-X_t^0B^1\right)'\left(Y_t^0-X_t^0B^1\right)$ where B^1 is the previous draw of the VAR coefficients reshaped into a matrix with dimensions $(N\times P)\times N$ so it is conformable with X_t^*.

Step 4 Draw μ from its conditional distribution. Villani (2009) shows that the conditional distribution of μ is given as $H\left(\mu|\bar{b},\Sigma,Z_t^*\right)\sim N\left(\mu^*,\Omega^*\right)$ where

$$\Omega^*=\left(\Sigma_\mu^{-1}+U'\left(D'D\otimes\Sigma^{-1}\right)U'\right)^{-1} \tag{4.7}$$

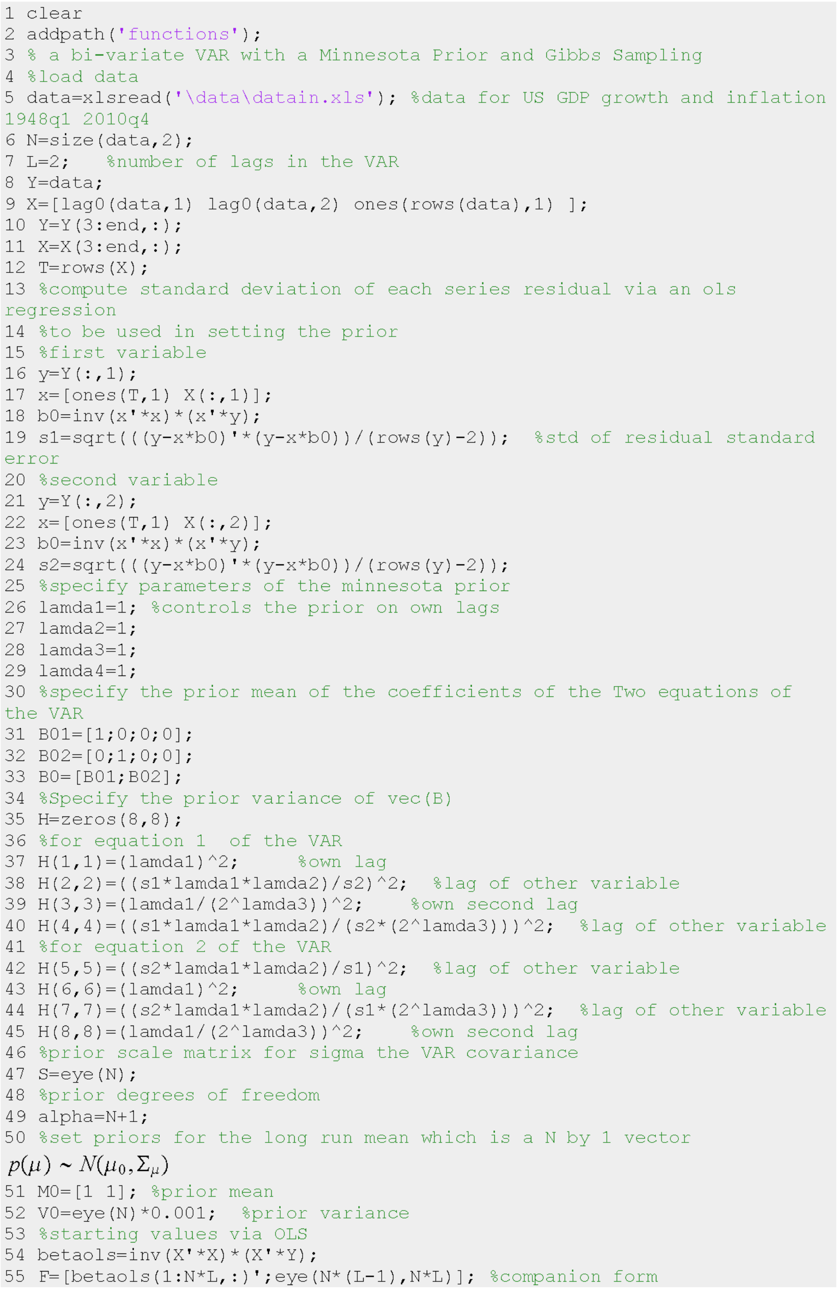
$$\mu^*=\Omega^*\left(U'vec\left(\Sigma^{-1}Y'D\right)+\Sigma_\mu^{-1}\mu_0\right) \tag{4.8}$$

where D is a $T\times(P+1)$ matrix $D=[c_t,-c_{t-1},...-c_{t-p}]$ where c_t is the constant term (a $T\times 1$) vector equal to one). U is a matrix with the following structure

$$U=\begin{pmatrix}I_N\\B_1\\.\\B_P\end{pmatrix}$$

For our two variable VAR U looks as follows

$$U=\begin{pmatrix}1&0\\0&1\\b_{11}&b_{12}\\b_{21}&b_{22}\\d_{11}&d_{12}\\d_{21}&d_{22}\end{pmatrix} \tag{4.9}$$



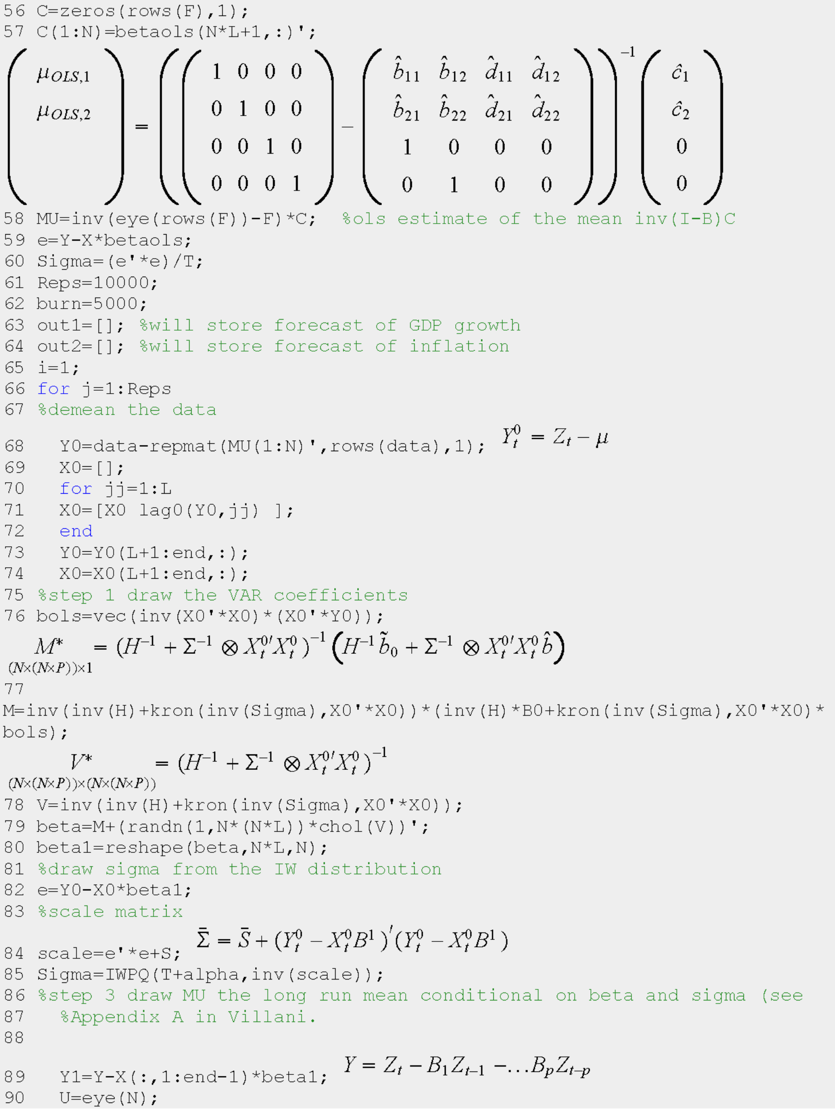
```
clear
addpath('functions');
% a bi-variate VAR with a Minnesota Prior and Gibbs Sampling
%load data
data=xlsread('\data\datain.xls'); %data for US GDP growth and inflation 1948q1 2010q4
N=size(data,2);
L=2;   %number of lags in the VAR
Y=data;
X=[lag0(data,1) lag0(data,2) ones(rows(data),1) ];
Y=Y(3:end,:);
X=X(3:end,:);
T=rows(X);
%compute standard deviation of each series residual via an ols regression
%to be used in setting the prior
%first variable
y=Y(:,1);
x=[ones(T,1) X(:,1)];
b0=inv(x'*x)*(x'*y);
s1=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));  %std of residual standard error
%second variable
y=Y(:,2);
x=[ones(T,1) X(:,2)];
b0=inv(x'*x)*(x'*y);
s2=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
%specify parameters of the minnesota prior
lamda1=1; %controls the prior on own lags
lamda2=1;
lamda3=1;
lamda4=1;
%specify the prior mean of the coefficients of the Two equations of the VAR
B01=[1;0;0;0];
B02=[0;1;0;0];
B0=[B01;B02];
%Specify the prior variance of vec(B)
H=zeros(8,8);
%for equation 1  of the VAR
H(1,1)=(lamda1)^2;     %own lag
H(2,2)=((s1*lamda1*lamda2)/s2)^2;  %lag of other variable
H(3,3)=(lamda1/(2^lamda3))^2;    %own second lag
H(4,4)=((s1*lamda1*lamda2)/(s2*(2^lamda3)))^2;  %lag of other variable
%for equation 2 of the VAR
H(5,5)=((s2*lamda1*lamda2)/s1)^2;  %lag of other variable
H(6,6)=(lamda1)^2;     %own lag
H(7,7)=((s2*lamda1*lamda2)/(s1*(2^lamda3)))^2;  %lag of other variable
H(8,8)=(lamda1/(2^lamda3))^2;    %own second lag
%prior scale matrix for sigma the VAR covariance
S=eye(N);
%prior degrees of freedom
alpha=N+1;
%set priors for the long run mean which is a N by 1 vector
p(μ) ~ N(μ0,Σμ)
M0=[1 1]; %prior mean
V0=eye(N)*0.001;  %prior variance
%starting values via OLS
betaols=inv(X'*X)*(X'*Y);
F=[betaols(1:N*L,:)';eye(N*(L-1),N*L)]; %companion form
```

Figure 9. Matlab code for VAR with steady state priors

Finally $Y = Z_t - B_1 Z_{t-1} - ...B_p Z_{t-p}$ where B_i denotes the VAR coefficients on the ith lag from the previous Gibbs iteration.

Step 5 Repeat steps 2 to 4 M times to obtain $B^1...B^M$ and $(\Sigma)^1 .... (\Sigma)^M$ and $\mu^1...\mu^M$ The last H values of B, μ and Σ from these iterations is used to form the empirical distribution of these parameters.

4.2. Gibbs sampling algorithm for the VAR with steady state priors. The matlab code. We estimate the VAR with steady state priors using the same data used in the first example (quarterly data on annual GDP growth and CPI inflation for the US from 1948Q2 to 2010Q4) and consider a long term forecast of these variables. The code



```
56 C=zeros(rows(F),1);
57 C(1:N)=betaols(N*L+1,:)';
```

$$\begin{pmatrix} \mu_{OLS,1} \\ \mu_{OLS,2} \\ \\ \\ \end{pmatrix} = \left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \hat{b}_{11} & \hat{b}_{12} & \hat{d}_{11} & \hat{d}_{12} \\ \hat{b}_{21} & \hat{b}_{22} & \hat{d}_{21} & \hat{d}_{22} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \right)^{-1} \begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ 0 \\ 0 \end{pmatrix}$$

```
58 MU=inv(eye(rows(F))-F)*C;  %ols estimate of the mean inv(I-B)C
59 e=Y-X*betaols;
60 Sigma=(e'*e)/T;
61 Reps=10000;
62 burn=5000;
63 out1=[]; %will store forecast of GDP growth
64 out2=[]; %will store forecast of inflation
65 i=1;
66 for j=1:Reps
67 %demean the data
68   Y0=data-repmat(MU(1:N)',rows(data),1);
```

$$Y_t^0 = Z_t - \mu$$

```
69   X0=[];
70   for jj=1:L
71   X0=[X0 lag0(Y0,jj) ];
72   end
73   Y0=Y0(L+1:end,:);
74   X0=X0(L+1:end,:);
75 %step 1 draw the VAR coefficients
76 bols=vec(inv(X0'*X0)*(X0'*Y0));
```

$$\underset{(N\times(N\times P))\times 1}{M^*} = (H^{-1} + \Sigma^{-1} \otimes X_t^{0\prime} X_t^0)^{-1} \left(H^{-1} \tilde{b}_0 + \Sigma^{-1} \otimes X_t^{0\prime} X_t^0 \hat{b} \right)$$

```
77
M=inv(inv(H)+kron(inv(Sigma),X0'*X0))*(inv(H)*B0+kron(inv(Sigma),X0'*X0)*
bols);
```

$$\underset{(N\times(N\times P))\times(N\times(N\times P))}{V^*} = (H^{-1} + \Sigma^{-1} \otimes X_t^{0\prime} X_t^0)^{-1}$$

```
78 V=inv(inv(H)+kron(inv(Sigma),X0'*X0));
79 beta=M+(randn(1,N*(N*L))*chol(V))';
80 beta1=reshape(beta,N*L,N);
81 %draw sigma from the IW distribution
82 e=Y0-X0*beta1;
83 %scale matrix
84 scale=e'*e+S;
```

$$\bar{\Sigma} = \bar{S} + (Y_t^0 - X_t^0 B^1)'(Y_t^0 - X_t^0 B^1)$$

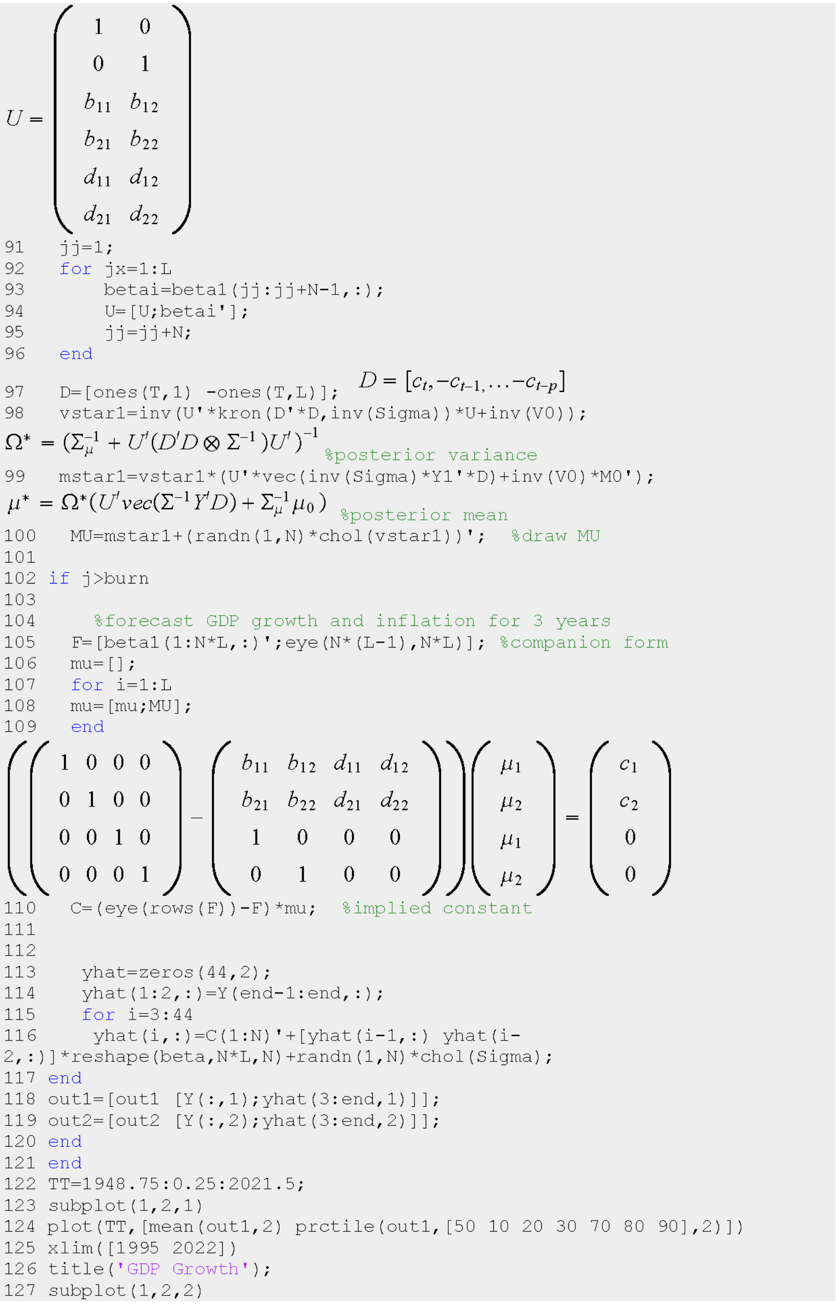
```
85 Sigma=IWPQ(T+alpha,inv(scale));
86 %step 3 draw MU the long run mean conditional on beta and sigma (see
87   %Appendix A in Villani.
88
89   Y1=Y-X(:,1:end-1)*beta1;
```

$$Y = Z_t - B_1 Z_{t-1} - \ldots B_p Z_{t-p}$$

```
90   U=eye(N);
```

FIGURE 10. Matlab code for steady state VAR (continued)

for the model (example3.m) is presented in figures 9, 10, 11 and 12. The code is identical to the first matlab example until line 50 where we set the prior for the long run means of the two variables. As an example we set the prior mean equal to 1 for both μ_1 and μ_2 and a tight prior variance. Lines 54 to 58 estimate the VAR coefficients via OLS and estimate a starting value for μ_1 and μ_2 as described in Step 1 of the Gibbs sampling algorithm above. Line 68 is the first step of the Gibbs algorithm and computes the demeaned data $Y_t^0 = Z_t - \mu$ and uses this on line 77 and 78 to compute the mean and the variance of the conditional posterior distribution $H\left(\bar{b}|\Sigma, \mu, Z_t^*\right)$ and samples the VAR coefficients from the normal distribution. Lines 81 to 85 draw Σ from the inverse Wishart distribution. Lines 89 to 100 draw μ_1 and μ_2 from the normal distribution. On line 89 the code creates the matrix $Y = Z_t - B_1 Z_{t-1} - ...B_p Z_{t-p}$.



$$U=\begin{pmatrix}1&0\\0&1\\b_{11}&b_{12}\\b_{21}&b_{22}\\d_{11}&d_{12}\\d_{21}&d_{22}\end{pmatrix}$$

```
91   jj=1;
92   for jx=1:L
93       betai=beta1(jj:jj+N-1,:);
94       U=[U;betai'];
95       jj=jj+N;
96   end
```

$$D=[c_t,-c_{t-1},\ldots-c_{t-p}]$$

```
97   D=[ones(T,1) -ones(T,L)];
98   vstar1=inv(U'*kron(D'*D,inv(Sigma))*U+inv(V0));
```

$$\Omega^* = (\Sigma_\mu^{-1} + U'(D'D\otimes\Sigma^{-1})U')^{-1}$$

```
%posterior variance
99   mstar1=vstar1*(U'*vec(inv(Sigma)*Y1'*D)+inv(V0)*M0');
```

$$\mu^* = \Omega^*(U'vec(\Sigma^{-1}Y'D)+\Sigma_\mu^{-1}\mu_0)$$

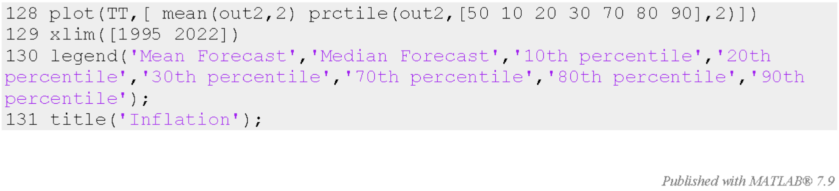
```
%posterior mean
100   MU=mstar1+(randn(1,N)*chol(vstar1))';  %draw MU
101
102 if j>burn
103
104      %forecast GDP growth and inflation for 3 years
105   F=[beta1(1:N*L,:)';eye(N*(L-1),N*L)]; %companion form
106   mu=[];
107   for i=1:L
108   mu=[mu;MU];
109   end
```

$$\left(\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}-\begin{pmatrix}b_{11}&b_{12}&d_{11}&d_{12}\\b_{21}&b_{22}&d_{21}&d_{22}\\1&0&0&0\\0&1&0&0\end{pmatrix}\right)\begin{pmatrix}\mu_1\\\mu_2\\\mu_1\\\mu_2\end{pmatrix}=\begin{pmatrix}c_1\\c_2\\0\\0\end{pmatrix}$$

```
110   C=(eye(rows(F))-F)*mu;  %implied constant
111
112
113    yhat=zeros(44,2);
114    yhat(1:2,:)=Y(end-1:end,:);
115    for i=3:44
116     yhat(i,:)=C(1:N)'+[yhat(i-1,:) yhat(i-
2,:)]*reshape(beta,N*L,N)+randn(1,N)*chol(Sigma);
117 end
118 out1=[out1 [Y(:,1);yhat(3:end,1)]];
119 out2=[out2 [Y(:,2);yhat(3:end,2)]];
120 end
121 end
122 TT=1948.75:0.25:2021.5;
123 subplot(1,2,1)
124 plot(TT,[mean(out1,2) prctile(out1,[50 10 20 30 70 80 90],2)])
125 xlim([1995 2022])
126 title('GDP Growth');
127 subplot(1,2,2)
```

Figure 11. Matlab code for VAR with steady state priors (continued)

Lines 90 to 96 create the matrix $U=\begin{pmatrix}1&0\\0&1\\b_{11}&b_{12}\\b_{21}&b_{22}\\d_{11}&d_{12}\\d_{21}&d_{22}\end{pmatrix}$. Line 97 creates the matrix $D=[c_t,-c_{t-1},\ldots-c_{t-p}]$. Lines 98 and 99 compute the variance and mean of the conditional posterior distribution of μ (see equation 4.7 and 4.8)while line 100 draws μ from the normal distribution. After the burn-in stage the VAR is used to do a forecast for



```
128 plot(TT,[ mean(out2,2) prctile(out2,[50 10 20 30 70 80 90],2)])
129 xlim([1995 2022])
130 legend('Mean Forecast','Median Forecast','10th percentile','20th
percentile','30th percentile','70th percentile','80th percentile','90th
percentile');
131 title('Inflation');
```

Published with MATLAB® 7.9

FIGURE 12. Matlab code for VAR with Steady State priors.

40 quarters. It is convenient to parameterise the VAR in the usual form i.e as in equation 4.1. On line 110 the code calculates the implied constants in the VAR using the fact that

$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} b_{11} & b_{12} & d_{11} & d_{12} \\ b_{21} & b_{22} & d_{21} & d_{22} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\right) \begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_1 \\ \mu_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ 0 \\ 0 \end{pmatrix}$$

and lines 113 to 117 calculate the forecast for each retained draw. The resulting forecast distribution in figure 13 is centered around the long run mean close to 1 for both variables.

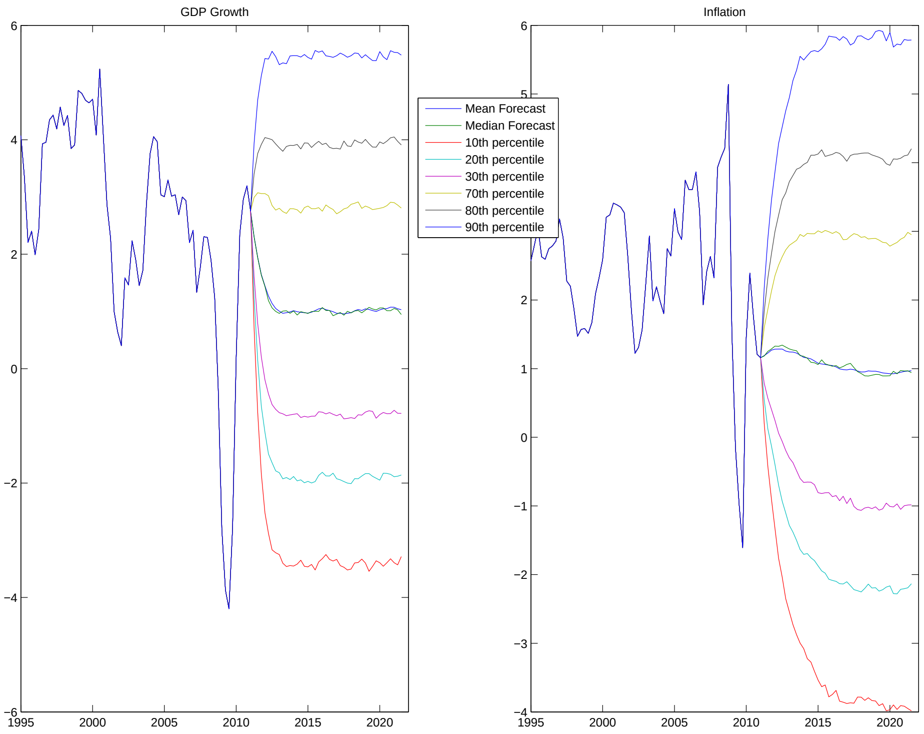


FIGURE 13. Forecast distribution for the VAR with steady state priors.

5. Implementing priors using dummy observations

The computation of the mean of the conditional posterior distribution (see equation 1.5) requires the inversion of $(N\times(N\times(P+1)))\times(N\times(N\times(P+1)))$ matrix $\left(H^{-1}+\Sigma^{-1}\otimes X_t'X_t\right)^{-1}$. For large VARs $(N\geq 20)$ this matrix has very large dimensions (e.g for $N=20$ and $P=2$ this is a 820×820 matrix). This can slow down the Gibbs sampling algorithm considerably. This computational constraint can be thought of as one potential disadvantage of the way we have incorporated the prior, i.e. via the covariance matrix H which has the dimensions $(N\times(N\times(P+1)))\times(N\times(N\times(P+1)))$.

Note also that our method of implementing the prior makes it difficult to incorporate priors about combination of coefficients in each equation or across equations. For instance, if one is interested in a prior that incorporates the belief that the sum of the coefficients on lags of the dependent variable in each equation sum to 1 (i.e. each variable has a unit root) this is very difficult to implement using a prior covariance matrix. Priors on combinations of coefficients across equations may arise from the implications of DSGE models (see Negro and Schorfheide (2004)). Again these are difficult to implement using the standard approach.

An alternative approach to incorporating prior information into the VAR is via dummy observations or artificial data. Informally speaking this involves generating artificial data from the model assumed under the prior and mixing this with the actual data. The weight placed on the artificial data determines how tightly the prior is imposed.

5.1. The Normal Wishart (Natural Conjugate) prior using dummy observations.

Consider artificial data denoted Y_D and X_D (we consider in detail below how to generate this data) such that

$$\begin{aligned} b_0 &= \left(X_D'X_D\right)^{-1}\left(X_D'Y_D\right) \\ S &= \left(Y_D-X_Db_0\right)'\left(Y_D-X_Db_0\right) \end{aligned} \tag{5.1}$$

where $\tilde{b}_0=vec(b_0)$. In other words a regression of Y_D on X_D gives the prior mean for the VAR coefficients and sum of squared residuals give the prior scale matrix for the error covariance matrix. The prior is of the normal inverse Wishart form

$$\begin{gathered} p(B|\Sigma)\tilde{}N\left(\tilde{b}_0,\Sigma\otimes\left(X_D'X_D\right)^{-1}\right) \\ p\left(\Sigma\right)\tilde{}IW(S,T_D-K) \end{gathered} \tag{5.2}$$

where T_D is the length of the artificial data and K denotes the number of regressors in each equation.

Given this artificial data, the conditional posterior distributions for the VAR parameters are given by

$$\begin{aligned} &H\left(b|\Sigma,Y_t\right)\tilde{\ }N(vec(B^*),\Sigma\otimes\left(X^{*\prime}X^*\right)^{-1}) \\ &H\left(\Sigma|b,Y_t\right)\tilde{\ }IW(S^*,T^*) \end{aligned} \tag{5.3}$$

where $Y^*=[Y;Y_D], X^*=[X;X_D]$ i.e. the actual VAR left and right hand side variables appended by the artificial data and T^* denotes the number of rows in Y^* and

$$B^*=\left(X^{*\prime}X^*\right)^{-1}\left(X^{*\prime}Y^*\right)$$

$$S^*=\left(Y^*-X^*b\right)'\left(Y^*-X^*b\right)$$

Note that the conditional posterior distribution has a simple form and the variance of $H\left(b|\Sigma,Y_t\right)$ only involves the inversion of $N\times P+1$ matrix making a Gibbs sampler based on this formulation much more computationally efficient in large models.

5.1.1. *Creating the dummy observations for the Normal Wishart prior.* The key question however is, where do Y_D and X_D come from? The artificial observations are formed by the researcher and are created using the following hyper-parameters:

- τ controls the overall tightness of the prior
- d controls the tightness of the prior on higher lags
- c controls the tightness of the prior on constants
- σ_i are standard deviation of error terms from OLS estimates of AR regression for each variable in the model

To discuss the creation of the dummy observations we are going to use the bi-variate VAR given below as an example:

$$\begin{pmatrix} y_t \\ x_t \end{pmatrix}=\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}+\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}\begin{pmatrix} y_{t-1} \\ x_{t-1} \end{pmatrix}+\begin{pmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{pmatrix}\begin{pmatrix} y_{t-2} \\ x_{t-2} \end{pmatrix}+\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, VAR\begin{pmatrix} v_1 \\ v_2 \end{pmatrix}=\Sigma \tag{5.4}$$

Consider dummy observations that implement the prior on the coefficients on the first lag of y_t and x_t. The artificial data (denoted by $Y_{D,1}$ and $X_{D,1}$) is given by

$$\begin{aligned} Y_{D,1} &= \begin{pmatrix} (1/\tau)\,\sigma_1 & 0 \\ 0 & (1/\tau)\,\sigma_2 \end{pmatrix} \\ X_{D,1} &= \begin{pmatrix} 0 & (1/\tau)\,\sigma_1 & 0 & 0 & 0 \\ 0 & 0 & (1/\tau)\,\sigma_2 & 0 & 0 \end{pmatrix} \end{aligned} \tag{5.5}$$

To see the intuition behind this formulation consider the VAR model using the artificial data

$$\underbrace{\begin{pmatrix} (1/\tau)\,\sigma_1 & 0 \\ 0 & (1/\tau)\,\sigma_2 \end{pmatrix}}_{Y_{D,1}}=\underbrace{\begin{pmatrix} 0 & (1/\tau)\,\sigma_1 & 0 & 0 & 0 \\ 0 & 0 & (1/\tau)\,\sigma_2 & 0 & 0 \end{pmatrix}}_{X_{D,1}}\underbrace{\begin{pmatrix} c_1 & c_2 \\ b_{11} & b_{21} \\ b_{12} & b_{22} \\ d_{11} & d_{21} \\ d_{12} & d_{22} \end{pmatrix}}_{B}+\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{5.6}$$

Expanding the equation above gives the following

$$\begin{pmatrix} (1/\tau)\,\sigma_1 & 0 \\ 0 & (1/\tau)\,\sigma_2 \end{pmatrix}=\begin{pmatrix} (1/\tau)\,\sigma_1 b_{11} & (1/\tau)\,\sigma_1 b_{21} \\ (1/\tau)\,\sigma_2 b_{12} & (1/\tau)\,\sigma_2 b_{22} \end{pmatrix}+\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{5.7}$$

Consider the first equation in the expression above $(1/\tau)\,\sigma_1=(1/\tau)\,\sigma_1 b_{11}+v_1$ or $b_{11}=1-\frac{\tau v_1}{\sigma_1}$. Taking the expected value of this gives $E\left(b_{11}\right)=1-E\left(\frac{\tau v_1}{\sigma_1}\right)$ which equals 1 as $E\left(v_1\right)=0$. In other words, the dummy variables imply a prior mean of 1 for b_{11}. Similarly, the variance of b_{11} is $\frac{\tau^2 var(v_1)}{\sigma_1^2}$. Note that the implied prior mean and variance for b_{11} is identical to the Natural conjugate prior discussed above. That is under the prior $b_{11}\tilde{\ }N\left(1,\frac{\tau^2 var(v_1)}{\sigma_1^2}\right)$. As $\tau\rightarrow 0$ the prior is imlemented more tightly.

Consider the second equation implied by expression 5.7: $0=(1/\tau)\,\sigma_1 b_{21}+v_2$ or $b_{21}=-\frac{\tau v_2}{\sigma_1}$. This implies that $E\left(b_{21}\right)=0$ and $var\left(b_{21}\right)=\frac{\tau^2 var(v_2)}{\sigma_1^2}$. Thus $b_{21}\tilde{\ }N\left(0,\frac{\tau^2 var(v_2)}{\sigma_1^2}\right)$ where the variance is of the same form as the corresponding element in equation 3.9.

Thus, the artificial observations in 5.5 implement the Normal inverse Wishart prior for the coefficients on the first lags of the two variables. We need to create artificial observations to implement the prior on the second lags. These are given by the following matrices

$$\begin{aligned} Y_{D,2} &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ X_{D,2} &= \begin{pmatrix} 0 & 0 & 0 & (1/\tau)\,\sigma_1 2^d & 0 \\ 0 & 0 & 0 & 0 & (1/\tau)\,\sigma_2 2^d \end{pmatrix} \end{aligned} \tag{5.8}$$

Proceeding as in equation 5.7 one can show that these dummy variables imply a prior mean of 0 for the second lag with the prior variance of the same form as in equation 3.9. For example, the prior variance associated with d_{11} is $\frac{\tau^2 var(v_1)}{\sigma_1^2 2^d}$.

The artificial observations that control the prior on the constants in the model are given by:

$$\begin{aligned} Y_{D,3} &= \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \\ X_{D,3} &= \begin{pmatrix} 1/c & 0 & 0 & 0 & 0 \\ 1/c & 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{5.9}$$

As $c \to 0$ the prior is imlemented more tightly. The dummy observations to implement the prior on the error covariance matrix are given by

$$\begin{aligned} Y_{D,4} &= \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \\ X_{D,4} &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{aligned} \tag{5.10}$$

with the magnitude of the diagonal elements of Σ controlled by the scale of the diagonal elements of $Y_{D,4}$ (i.e. larger diagonal elements implement the prior belief that the variance of v_1 and v_2 is larger).

The prior is implemented by adding all these dummy observations to the actual data. That is

$$Y^* = [Y; Y_{D,1}; Y_{D,2}; Y_{D,3}; Y_{D,4}], X^* = [X; X_{D,1}; X_{D,2}; X_{D,3}; X_{D,4}]$$

With this appended data in hand, the conditional distributions in equation 5.3 can be used to implement the Gibbs sampling algorithm. Note that, as discussed in Banbura *et al.* (2007), these dummy observations for a general N variable VAR with P lags are given as

$$Y_D = \begin{pmatrix} \frac{diag(\chi_1\sigma_1 ... \chi_N\sigma_N)}{\tau} \\ 0_{N\times(P-1)\times N} \\ .... \\ diag\left(\sigma_1 ... \sigma_N\right) \\ .... \\ 0_{1\times N} \end{pmatrix}, X_D = \begin{pmatrix} \frac{J_P \otimes diag(\sigma_1 ... \sigma_N)}{\tau} & 0_{NP\times 1} \\ 0_{N\times NP} & 0_{N\times 1} \\ \multicolumn{2}{c}{..............} \\ 0_{1\times NP} & c \end{pmatrix} \tag{5.11}$$

where χ_i are the prior means for the coefficients on the first lags of the dependent variables (these can be different from 1) and $J_P = diag(1..P)$

5.1.2. *Creating dummy variables for the sum of coefficients prior.* If the variables in the VAR have a unit root, this information can be reflected via a prior that incorporates the belief that coefficients on lags of the dependent variable sum to 1 (see Robertson and Tallman (1999)). This prior can be implemented in our example VAR via the following dummy observations

$$Y_{D,5} = \begin{pmatrix} \gamma\mu_1 & 0 \\ 0 & \gamma\mu_2 \end{pmatrix}, X_{D,5} = \begin{pmatrix} 0 & \gamma\mu_1 & 0 & \gamma\mu_1 & 0 \\ 0 & 0 & \gamma\mu_2 & 0 & \gamma\mu_2 \end{pmatrix} \tag{5.12}$$

where μ_1 is the sample mean of y_t and μ_2 is the sample mean of x_t possibly calculated using an intial sample of data. Note that these dummy observations imply prior means of the form $b_{ii} + d_{ii} = 1$ where $i = 1, 2$ and γ controls the tightness of the prior. As $\gamma \to \infty$ the prior is implemented more tightly. Banbura *et al.* (2007) show that these dummy observations for a N variable VAR with P lags are given as

$$Y_D = \frac{diag\left(\chi_1\mu_1 ... \chi_N\mu_N\right)}{\lambda}, X_D = \begin{pmatrix} \frac{(1,2..P)\otimes diag(\chi_1\mu_1 ... \chi_N\mu_N)}{\lambda} & 0_{N\times 1} \end{pmatrix} \tag{5.13}$$

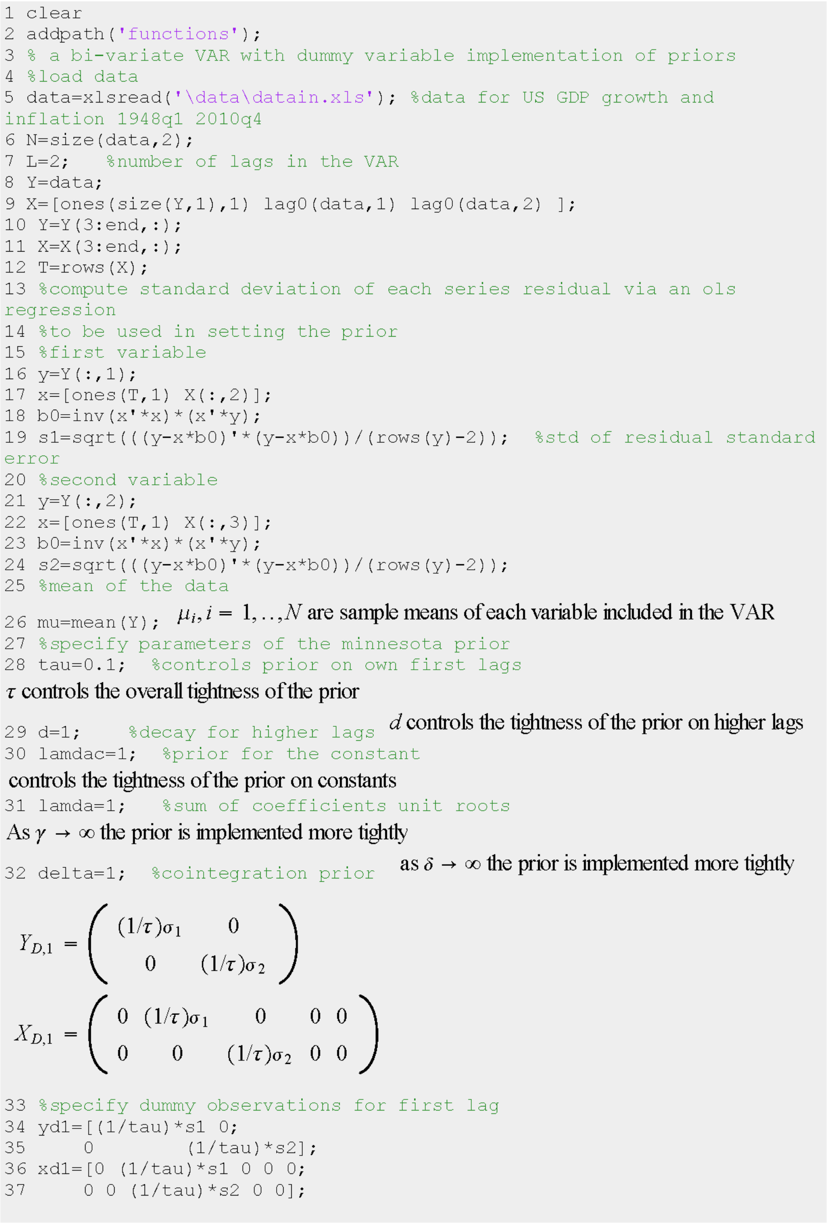
where $\lambda = 1/\gamma$ and $\mu_i, i = 1, .., N$ are sample means of each variable included in the VAR.

5.1.3. *Creating dummy variables for the common stochastic trends prior.* One can express the prior belief that the variables in the VAR have a common stochastic trend via the following dummy observations

$$Y_{D,6} = \begin{pmatrix} \delta\mu_1 & \delta\mu_2 \end{pmatrix}, X_{D,6} = \begin{pmatrix} \delta & \delta\mu_1 & \delta\mu_2 & \delta\mu_1 & \delta\mu_2 \end{pmatrix} \tag{5.14}$$

These dummy observations imply, for example, that $\mu_1 = c_1 + \mu_1 b_{11} + \mu_2 b_{12} + \mu_1 d_{11} + \mu_2 d_{12}$ i.e. the mean of the first variable is a combination of μ_1 and μ_2. Note as $\delta \to \infty$ the prior is implemented more tightly and the series in the VAR share a common stochastic trend.

5.1.4. *Matlab code for implementing priors using dummy observations.* Figures 14, 15 and 16 show the matlab code for the bi-variate VAR(2) model using quarterly data on annual GDP growth and CPI inflation for the US from 1948Q2 to 2010Q4 (example4.m). Line 26 of the code calculates the sample means of the data to be used in setting the dummy observations. Some researchers use a pre-sample to calculate these means and the standard deviations σ_i. Lines 28 to 32 specify the parameters that control the prior. Lines 33 to 37 set the dummy observations for the VAR coefficients on the first lags. Lines 38 to 42 set the dummy observations for the VAR coefficients on the second lag. Lines 43 to 47 specify the dummy observations for the prior on the constant. Lines 48 to 53 specify



```
1 clear
2 addpath('functions');
3 % a bi-variate VAR with dummy variable implementation of priors
4 %load data
5 data=xlsread('\data\datain.xls'); %data for US GDP growth and
inflation 1948q1 2010q4
6 N=size(data,2);
7 L=2;   %number of lags in the VAR
8 Y=data;
9 X=[ones(size(Y,1),1) lag0(data,1) lag0(data,2) ];
10 Y=Y(3:end,:);
11 X=X(3:end,:);
12 T=rows(X);
13 %compute standard deviation of each series residual via an ols
regression
14 %to be used in setting the prior
15 %first variable
16 y=Y(:,1);
17 x=[ones(T,1) X(:,2)];
18 b0=inv(x'*x)*(x'*y);
19 s1=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));  %std of residual standard
error
20 %second variable
21 y=Y(:,2);
22 x=[ones(T,1) X(:,3)];
23 b0=inv(x'*x)*(x'*y);
24 s2=sqrt(((y-x*b0)'*(y-x*b0))/(rows(y)-2));
25 %mean of the data
26 mu=mean(Y);
```

$\mu_i, i = 1, .., N$ are sample means of each variable included in the VAR

```
27 %specify parameters of the minnesota prior
28 tau=0.1;  %controls prior on own first lags
```

τ controls the overall tightness of the prior

```
29 d=1;    %decay for higher lags
```

d controls the tightness of the prior on higher lags

```
30 lamdac=1;  %prior for the constant
```

controls the tightness of the prior on constants

```
31 lamda=1;   %sum of coefficients unit roots
```

As $\gamma \to \infty$ the prior is implemented more tightly

```
32 delta=1;  %cointegration prior
```

as $\delta \to \infty$ the prior is implemented more tightly

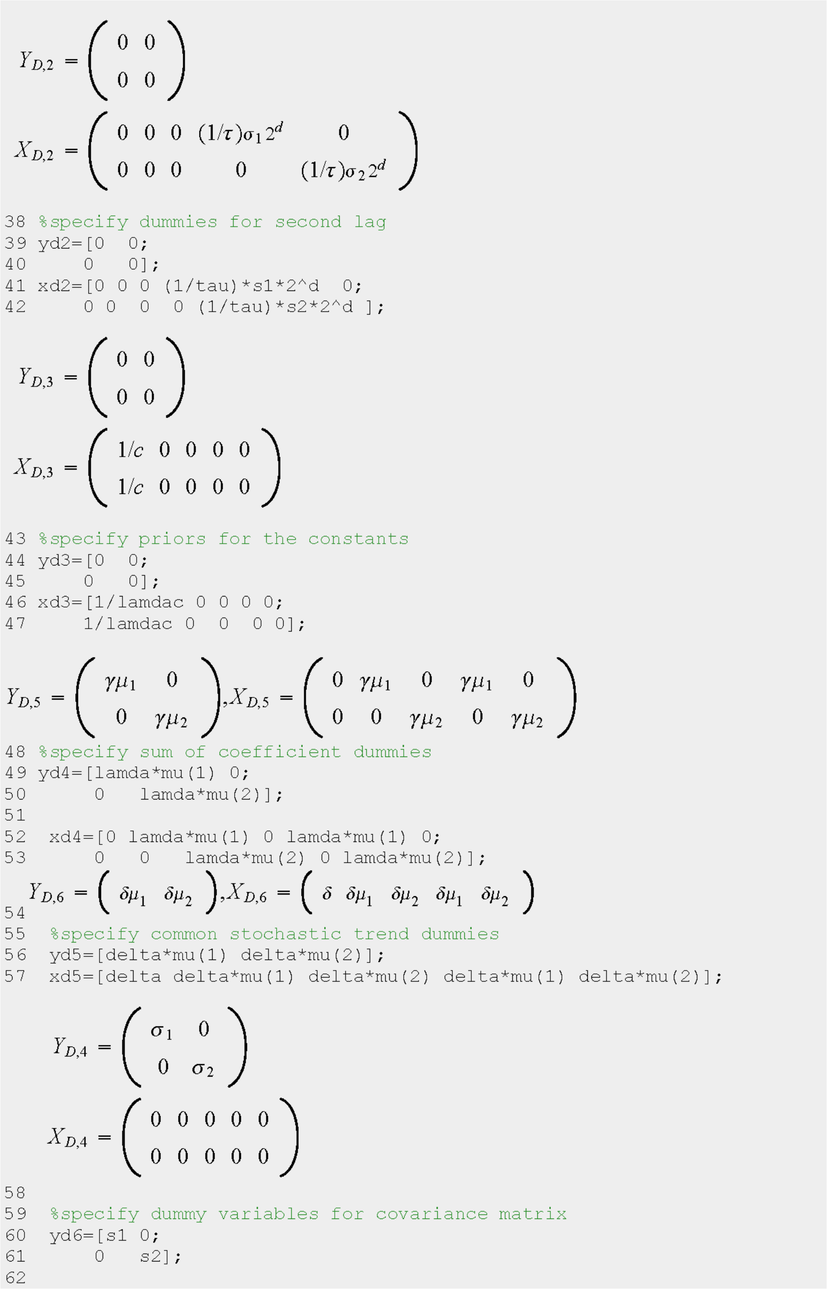
$$Y_{D,1} = \begin{pmatrix} (1/\tau)\sigma_1 & 0 \\ 0 & (1/\tau)\sigma_2 \end{pmatrix}$$

$$X_{D,1} = \begin{pmatrix} 0 & (1/\tau)\sigma_1 & 0 & 0 & 0 \\ 0 & 0 & (1/\tau)\sigma_2 & 0 & 0 \end{pmatrix}$$

```
33 %specify dummy observations for first lag
34 yd1=[(1/tau)*s1 0;
35     0        (1/tau)*s2];
36 xd1=[0 (1/tau)*s1 0 0 0;
37     0 0 (1/tau)*s2 0 0];
```

FIGURE 14. Normal Wishart prior using dummy observations

the dummy observations for the unit root prior. Lines 56 to 57 set out the dummy observations for the common stochastic trends prior. Lines 59 to 64 specify the dummy observations for the prior on the covariance matrix. Lines 68 and 69 mix the actual observations with the dummy data creating $Y^* = [Y; Y_D], X^* = [X; X_D]$. Line 72 computes the mean of the conditional posterior distribution of the VAR coefficients $B^* = (X^{*\prime}X^*)^{-1}(X^{*\prime}Y^*)$. Line 83 calculates the variance of this posterior distribution $\Sigma \otimes (X^{*\prime}X^*)^{-1}$ and line 85 draws the VAR coefficients from the normal distribution with this mean and variance. Line 88 calculates the scale matrix for the inverse Wishart density $S^* = (Y^* - X^*B^*)'(Y^* - X^*B^*)$ and line 89 draws the covariance matrix from the inverse Wishart distribution. Once past the burn-in stage the code forecasts the two variables in the VAR and builds up the predictive density.



$$Y_{D,2}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$$

$$X_{D,2}=\begin{pmatrix}0 & 0 & 0 & (1/\tau)\sigma_1 2^d & 0\\ 0 & 0 & 0 & 0 & (1/\tau)\sigma_2 2^d\end{pmatrix}$$

```
%specify dummies for second lag
yd2=[0  0;
    0   0];
xd2=[0 0 0 (1/tau)*s1*2^d  0;
    0 0  0  0 (1/tau)*s2*2^d ];
```

$$Y_{D,3}=\begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix}$$

$$X_{D,3}=\begin{pmatrix}1/c & 0 & 0 & 0 & 0\\ 1/c & 0 & 0 & 0 & 0\end{pmatrix}$$

```
%specify priors for the constants
yd3=[0  0;
    0   0];
xd3=[1/lamdac 0 0 0 0;
    1/lamdac 0  0  0 0];
```

$$Y_{D,5}=\begin{pmatrix}\gamma\mu_1 & 0\\ 0 & \gamma\mu_2\end{pmatrix}, X_{D,5}=\begin{pmatrix}0 & \gamma\mu_1 & 0 & \gamma\mu_1 & 0\\ 0 & 0 & \gamma\mu_2 & 0 & \gamma\mu_2\end{pmatrix}$$

```
%specify sum of coefficient dummies
yd4=[lamda*mu(1) 0;
     0   lamda*mu(2)];

 xd4=[0 lamda*mu(1) 0 lamda*mu(1) 0;
     0   0   lamda*mu(2) 0 lamda*mu(2)];
```

$$Y_{D,6}=\begin{pmatrix}\delta\mu_1 & \delta\mu_2\end{pmatrix}, X_{D,6}=\begin{pmatrix}\delta & \delta\mu_1 & \delta\mu_2 & \delta\mu_1 & \delta\mu_2\end{pmatrix}$$

```
 %specify common stochastic trend dummies
 yd5=[delta*mu(1) delta*mu(2)];
 xd5=[delta delta*mu(1) delta*mu(2) delta*mu(1) delta*mu(2)];
```

$$Y_{D,4}=\begin{pmatrix}\sigma_1 & 0\\ 0 & \sigma_2\end{pmatrix}$$

$$X_{D,4}=\begin{pmatrix}0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\end{pmatrix}$$

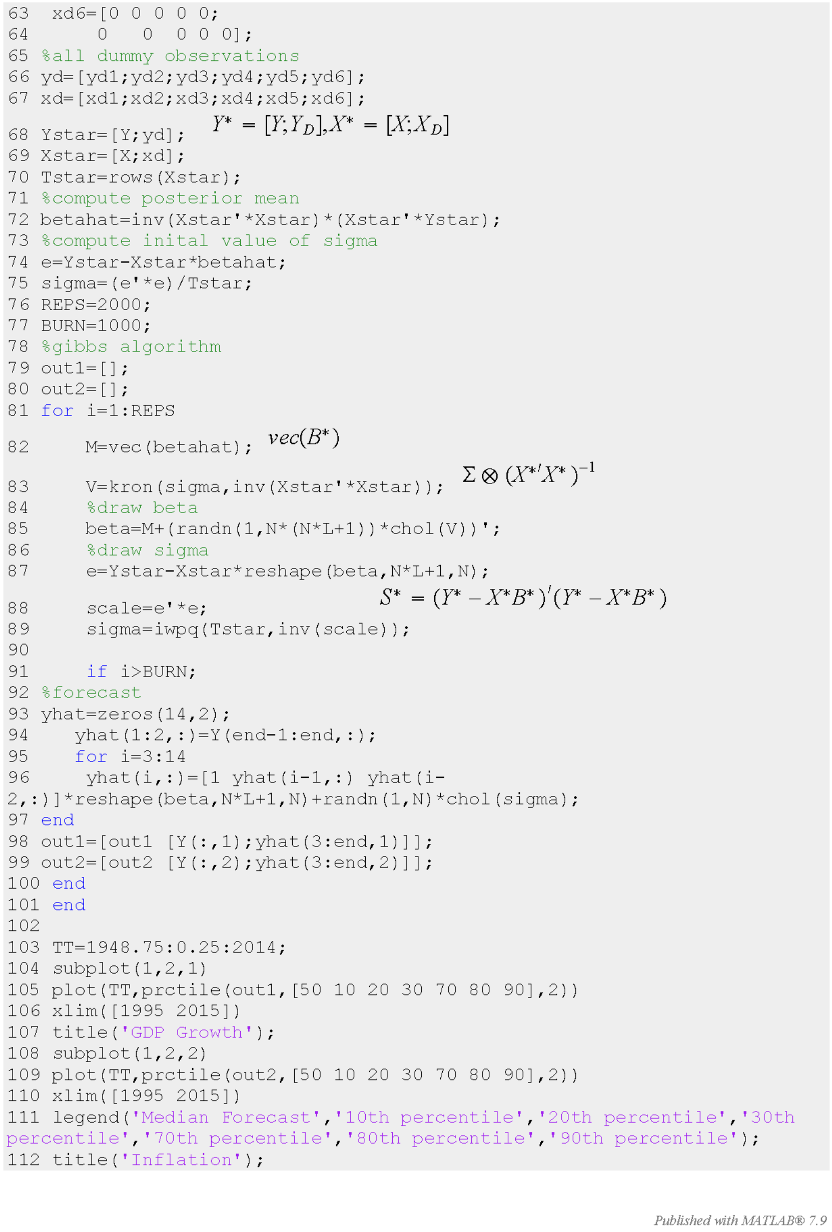
```

 %specify dummy variables for covariance matrix
 yd6=[s1 0;
     0   s2];

```

FIGURE 15. Normal Wishart prior using dummy observations continued

6. Application1: Structural VARs and sign restrictions

Structural VAR models offer a simple and flexible framework for analysing several questions of interest. Once structural shocks are identified using an appropriate identification scheme, impulse response analysis, variance decomposition and historical decomposition offer powerful tools. For a detailed explanation of structural VARs see Hamilton (1994) or Canova (2007). In this section we focus on how structural analysis fits in the Gibbs sampling framework established in the chapter.

As shown in the matlab example in section 3.2.1, one can estimate structural VARs by calculating the structural impact matrix A_0 (where $\Sigma = A_0'A_0$) for each retained Gibbs draw and use this to compute impulse response



```
xd6=[0 0 0 0 0;
     0   0  0 0 0];
%all dummy observations
yd=[yd1;yd2;yd3;yd4;yd5;yd6];
xd=[xd1;xd2;xd3;xd4;xd5;xd6];
Ystar=[Y;yd];
Xstar=[X;xd];
Tstar=rows(Xstar);
%compute posterior mean
betahat=inv(Xstar'*Xstar)*(Xstar'*Ystar);
%compute inital value of sigma
e=Ystar-Xstar*betahat;
sigma=(e'*e)/Tstar;
REPS=2000;
BURN=1000;
%gibbs algorithm
out1=[];
out2=[];
for i=1:REPS
    M=vec(betahat);
    V=kron(sigma,inv(Xstar'*Xstar));
    %draw beta
    beta=M+(randn(1,N*(N*L+1))*chol(V))';
    %draw sigma
    e=Ystar-Xstar*reshape(beta,N*L+1,N);
    scale=e'*e;
    sigma=iwpq(Tstar,inv(scale));

    if i>BURN;
%forecast
yhat=zeros(14,2);
   yhat(1:2,:)=Y(end-1:end,:);
   for i=3:14
    yhat(i,:)=[1 yhat(i-1,:) yhat(i-
2,:)]*reshape(beta,N*L+1,N)+randn(1,N)*chol(sigma);
end
out1=[out1 [Y(:,1);yhat(3:end,1)]];
out2=[out2 [Y(:,2);yhat(3:end,2)]];
end
end

TT=1948.75:0.25:2014;
subplot(1,2,1)
plot(TT,prctile(out1,[50 10 20 30 70 80 90],2))
xlim([1995 2015])
title('GDP Growth');
subplot(1,2,2)
plot(TT,prctile(out2,[50 10 20 30 70 80 90],2))
xlim([1995 2015])
legend('Median Forecast','10th percentile','20th percentile','30th
percentile','70th percentile','80th percentile','90th percentile');
title('Inflation');
```

$Y^* = [Y;Y_D], X^* = [X;X_D]$

$vec(B^*)$

$\Sigma \otimes (X^{*\prime}X^*)^{-1}$

$S^* = (Y^* - X^*B^*)'(Y^* - X^*B^*)$

Figure 16. Normal Wishart prior using dummy observations continued

functions, variance decompositions and historical decompositions. The Gibbs sampling framework is convenient because it allows one to build a distribution for these objects (i.e. impulse response functions, variance decompositions and historical decompositions) and thus characterise uncertainty about these estimates.

Strictly speaking, this indirect method of estimating structural VARs–i.e. calculating A_0 using a Gibbs draw of Σ (and not sampling A_0 directly) provides the posterior distribution of A_0 only if the structural VAR is exactly identified (for e.g. when A_0 is calculated using a Cholesky decomposition as in section 3.2.1) . In the case of over identification one needs to estimate the posterior of A_0 directly (see Sims and Zha (1998)). We will consider such an example in Chapter 4.

Recent applications of structural VARs have used sign restrictions to identify structural shocks (for a critical survey see Fry and Pagan (2007)). Despite the issues raised in Sims and Zha (1998), sign restrictions are implemented using an indirect algorithm. In other words for each retained draw of Σ one calculates an A_0 matrix which results in impulse responses to a shock of interest with signs that are consistent with theory. For example to identify a monetary policy shock one may want an A_0 matrix that leads to a response of output and inflation that is negative and a response of the policy interest rate that is positive for a few periods after the shock.

Ramirez *et al.* (2010) provide an efficient algorithm to find an A_0 matrix consistent with impulse responses of a certain sign consistent with theory. We review this algorithm by considering the following VAR(1) model as an example

$$\begin{pmatrix} Y_t \\ P_t \\ R_t \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} \begin{pmatrix} Y_{t-1} \\ P_{t-1} \\ R_{t-1} \end{pmatrix} + \begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \end{pmatrix} \tag{6.1}$$

where $var\begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \end{pmatrix} = \Sigma, Y_t$ is output growth, P_t is inflation and R_t is the interest rate. The aim is to calculate the impulse response to a monetary policy shock. The monetary policy shock is assumed to be one that decreases Y_t and P_t and increases R_t in the period of the shock. As described above, the Gibbs sampling algorithm to estimate the parameters of the VAR model cycles through two steps, sampling successively from $H(b|\Sigma, Y_t)$ and $H(\Sigma|b, Y_t)$. Once past the burn-in stage the following steps are used calculate the required A_0 matrix:

Step 1 Draw a $N \times N$ matrix K from the standard normal distribution

Step 2 Calculate the matrix Q from the QR decomposition of K. Note that Q is orthonormal i.e. $Q'Q = I$.

Step 3 Calculate the Cholesky decomposition of the current draw of $\Sigma = \tilde{A}_0'\tilde{A}_0$

Step 4 Calculate the candidate A_0 matrix as $A_0 = Q\tilde{A}_0$. Note that because $Q'Q = I$ this implies that $A_0'A_0$ will still equal Σ. By calculating the product $Q\tilde{A}_0$ we alter the elements of $\tilde{A}_0$ but not the property that $\Sigma = \tilde{A}_0'\tilde{A}_0$. The candidate A_0 matrix in our 3 variable VAR example will have the following form

$$A_0 = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

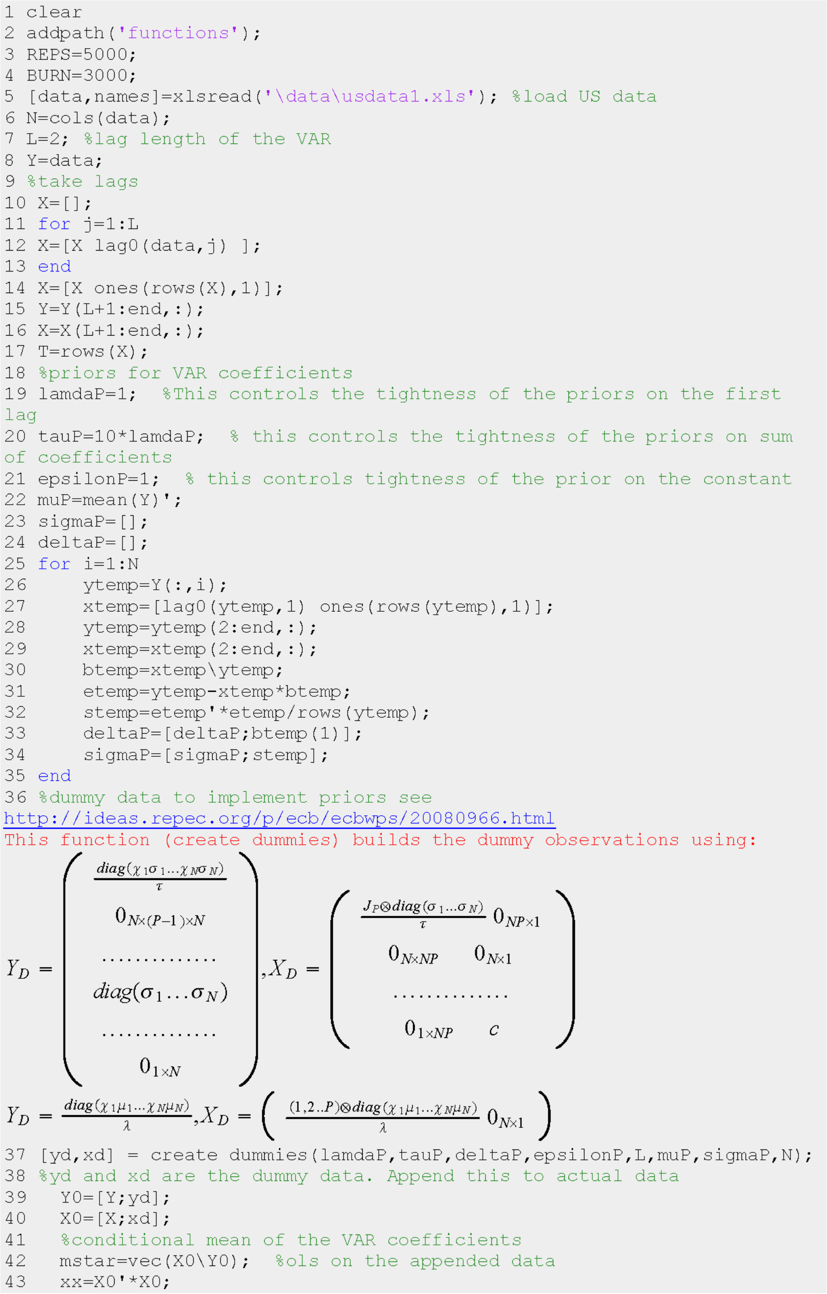
The third row of this matrix corresponds with the interest rate shock. We need to check if $a_{31} < 0$ and $a_{32} < 0$ and $a_{33} > 0$. If this is the case a contemporaneous increase in R_t will lead to a fall in Y_t and P_t as the elements$\begin{pmatrix} a_{31} & a_{32} & a_{33} \end{pmatrix}$ correspond to the current period impulse response of Y_t, P_t and R_t respectively. If $a_{31} < 0$ and $a_{32} < 0$ and $a_{33} > 0$ we stop and use this A_0 matrix to compute impulse responses and other objects of interest. If the restriction is not satisfied we go to step 1 and try with an new K matrix.

Step 5 Repeat steps 1 to 4 for every retained Gibbs draw.

6.1. A Structural VAR with sign restrictions in matlab. We estimate a large scale VAR model for the US using quarterly data from 1971Q1 to 2010Q4 (example5.m). The VAR model includes the following variables (in this order): (1) Federal Funds Rate (2) Annual GDP growth (3) Annual CPI Inflation (4) Annual real consumption growth (5) Unemployment rate (6) change in private investment (7) net exports (8) annual growth in M2 (9) 10 year government bond yield (10) annual growth in stock prices (11) annual growth in the yen dollar exchange rate. We identify a monetary policy shock by assuming that a monetary contraction has the following contemporaneous effects

Variable	Sign restriction
Federal Funds Rate	+
Annual GDP growth	-
Annual CPI Inflation	-
Annual Real Consumption Growth	-
Unemployment rate	+
Annual Investment Growth	-
Annual Money Growth	-

The Matlab code for this example is shown in figures 17, 18 and 19. Line 37 of the code builds the dummy observations for the normal wishart prior for the VAR using equations 5.11 and the sum of coefficients prior using equation 5.13 via the function create_dummies.m in function folder. Lines 49 to 55 sample from the conditional posterior distributions as in the previous example. Once past the burn-in stage, on line 61 we draw a $N \times N$ matrix from the standard normal distribution. Line 62 takes the QR decomposition of K and obtaines the matrix Q. Line 63 calculates the Cholesky decomposition of Σ while line 64 calculates the candidate A_0 matrix as $A_0 = Q\tilde{A}_0$. Lines 66 to 72 check if the sign restrictions are satisfied by checking the elements of the first row of the A_0 matrix (the row that corresponds the interest rate). Lines 77 to 83 check if the sign restrictions are satisfied with the sign reversed. If they are, we multiply the entire first row of the A_0 matrix by -1. The code keeps on drawing K and calculating candidate $A_0 = Q\tilde{A}_0$ matrices until an A_0 matrix is found that satisfies the sign restrictions. Once an A_0 matrix is



```matlab
clear
addpath('functions');
REPS=5000;
BURN=3000;
[data,names]=xlsread('\data\usdata1.xls'); %load US data
N=cols(data);
L=2; %lag length of the VAR
Y=data;
%take lags
X=[];
for j=1:L
X=[X lag0(data,j) ];
end
X=[X ones(rows(X),1)];
Y=Y(L+1:end,:);
X=X(L+1:end,:);
T=rows(X);
%priors for VAR coefficients
lamdaP=1;  %This controls the tightness of the priors on the first lag
tauP=10*lamdaP;  % this controls the tightness of the priors on sum of coefficients
epsilonP=1;  % this controls tightness of the prior on the constant
muP=mean(Y)';
sigmaP=[];
deltaP=[];
for i=1:N
    ytemp=Y(:,i);
    xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
    ytemp=ytemp(2:end,:);
    xtemp=xtemp(2:end,:);
    btemp=xtemp\ytemp;
    etemp=ytemp-xtemp*btemp;
    stemp=etemp'*etemp/rows(ytemp);
    deltaP=[deltaP;btemp(1)];
    sigmaP=[sigmaP;stemp];
end
%dummy data to implement priors see
http://ideas.repec.org/p/ecb/ecbwps/20080966.html
This function (create dummies) builds the dummy observations using:
```

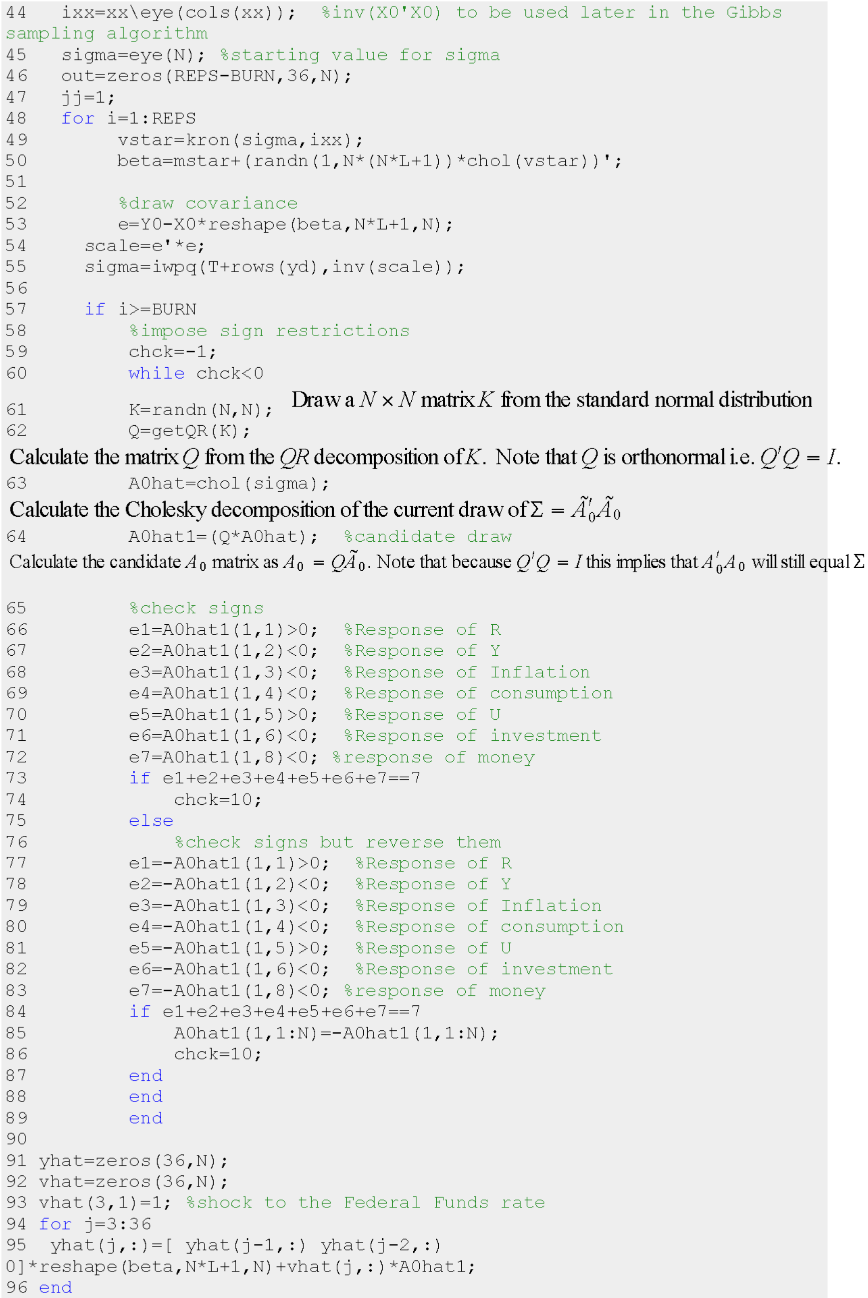
$$Y_D = \begin{pmatrix} \frac{diag(\chi_1\sigma_1 \ldots \chi_N\sigma_N)}{\tau} \\ 0_{N\times(P-1)\times N} \\ \ldots\ldots\ldots\ldots\ldots \\ diag(\sigma_1 \ldots \sigma_N) \\ \ldots\ldots\ldots\ldots\ldots \\ 0_{1\times N} \end{pmatrix}, X_D = \begin{pmatrix} \frac{J_P \otimes diag(\sigma_1 \ldots \sigma_N)}{\tau} & 0_{NP\times 1} \\ 0_{N\times NP} & 0_{N\times 1} \\ \ldots\ldots & \ldots\ldots \\ 0_{1\times NP} & c \end{pmatrix}$$

$$Y_D = \frac{diag(\chi_1\mu_1 \ldots \chi_N\mu_N)}{\lambda}, X_D = \begin{pmatrix} \frac{(1,2..P)\otimes diag(\chi_1\mu_1 \ldots \chi_N\mu_N)}{\lambda} & 0_{N\times 1} \end{pmatrix}$$

```matlab
[yd,xd] = create dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
%yd and xd are the dummy data. Append this to actual data
  Y0=[Y;yd];
  X0=[X;xd];
  %conditional mean of the VAR coefficients
  mstar=vec(X0\Y0);  %ols on the appended data
  xx=X0'*X0;
```

Figure 17. A Structural VAR with sign restrictions: Matlab Code

found that satisfies the sign restrictions, this is used to calculate the impulse response to a monetary policy shock and the impulse response functions for each retained Gibbs draw are saved. The file example6.m has exactly the same code but makes the algorithm to find the A_0 matrix more efficient by searching all rows of candidate $A_0 = Q\tilde{A}_0$ matrix for the one consistent with the policy shock–i.e. with the signs as in the table above. Once this is found we insert this row into the first row of the candidate A_0 matrix. Note that that this re-shuffling of rows does not alter the property that $\Sigma = A_0' A_0$. Note also that the A_0 matrix is not unique. That is, one could find A_0 matrices that satisfy the sign restrictions but have elements of different magnitude. Some researchers deal with this issue by generating M A_0 matrices that satisfy the sign restrictions for each Gibbs draw and then retaining the A_0 matrix that is closest to



```
  ixx=xx\eye(cols(xx));  %inv(X0'X0) to be used later in the Gibbs sampling algorithm
  sigma=eye(N); %starting value for sigma
  out=zeros(REPS-BURN,36,N);
  jj=1;
  for i=1:REPS
      vstar=kron(sigma,ixx);
      beta=mstar+(randn(1,N*(N*L+1))*chol(vstar))';

      %draw covariance
      e=Y0-X0*reshape(beta,N*L+1,N);
    scale=e'*e;
    sigma=iwpq(T+rows(yd),inv(scale));

    if i>=BURN
        %impose sign restrictions
        chck=-1;
        while chck<0
        K=randn(N,N);
```

Draw a $N \times N$ matrix K from the standard normal distribution

```
        Q=getQR(K);
```

Calculate the matrix Q from the QR decomposition of K. Note that Q is orthonormal i.e. $Q'Q = I$.

```
        A0hat=chol(sigma);
```

Calculate the Cholesky decomposition of the current draw of $\Sigma = \tilde{A}_0'\tilde{A}_0$

```
        A0hat1=(Q*A0hat);  %candidate draw
```

Calculate the candidate A_0 matrix as $A_0 = Q\tilde{A}_0$. Note that because $Q'Q = I$ this implies that $A_0'A_0$ will still equal Σ

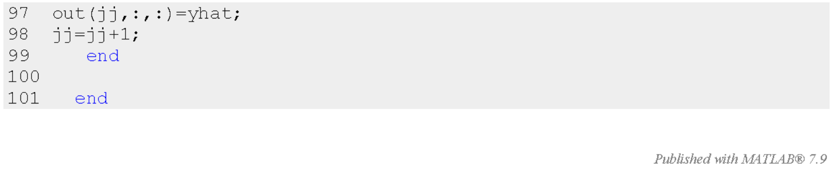
```
        %check signs
        e1=A0hat1(1,1)>0;  %Response of R
        e2=A0hat1(1,2)<0;  %Response of Y
        e3=A0hat1(1,3)<0;  %Response of Inflation
        e4=A0hat1(1,4)<0;  %Response of consumption
        e5=A0hat1(1,5)>0;  %Response of U
        e6=A0hat1(1,6)<0;  %Response of investment
        e7=A0hat1(1,8)<0; %response of money
        if e1+e2+e3+e4+e5+e6+e7==7
            chck=10;
        else
            %check signs but reverse them
        e1=-A0hat1(1,1)>0;  %Response of R
        e2=-A0hat1(1,2)<0;  %Response of Y
        e3=-A0hat1(1,3)<0;  %Response of Inflation
        e4=-A0hat1(1,4)<0;  %Response of consumption
        e5=-A0hat1(1,5)>0;  %Response of U
        e6=-A0hat1(1,6)<0;  %Response of investment
        e7=-A0hat1(1,8)<0; %response of money
        if e1+e2+e3+e4+e5+e6+e7==7
            A0hat1(1,1:N)=-A0hat1(1,1:N);
            chck=10;
        end
        end
        end

yhat=zeros(36,N);
vhat=zeros(36,N);
vhat(3,1)=1; %shock to the Federal Funds rate
for j=3:36
 yhat(j,:)=[ yhat(j-1,:) yhat(j-2,:) 0]*reshape(beta,N*L+1,N)+vhat(j,:)*A0hat1;
end
```

FIGURE 18. A structural VAR with sign restrictions (continued)

the mean or median of these M matrices. This imples that one restricts the distribution of the selected A_0 matrices via a (arbitrary) rule. The file example7.m does this for our example by generating 100 A_0 matrices for each retained Gibbs draw and using the A_0 matrix closest to the median to compute the impulse response functions. Figure 20 shows the estimated impulse response functions computed using example7.m

7. Application 2: Conditional forecasting using VARs and Gibbs sampling

In many cases (relevant to central bank applications) forecasts of macroeconomic variables that are conditioned on fixed paths for other variables is required. For example, one may wish to forecast credit and asset prices assuming



```
out(jj,:,:)=yhat;
jj=jj+1;
   end

 end
```

Published with MATLAB® 7.9

Figure 19. A structural VAR with sign restrictions: Matlab code (continued)

that inflation and GDP growth follow future paths fixed at the official central bank forecast. Waggoner and Zha (1999) provide a convenient framework to calculate not only the conditional forecasts but also the forecast distribution using a Gibbs sampling algorithm.

To see their approach consider a simple VAR(1) model

$$Y_t = c + BY_{t-1} + A_0 \varepsilon_t \tag{7.1}$$

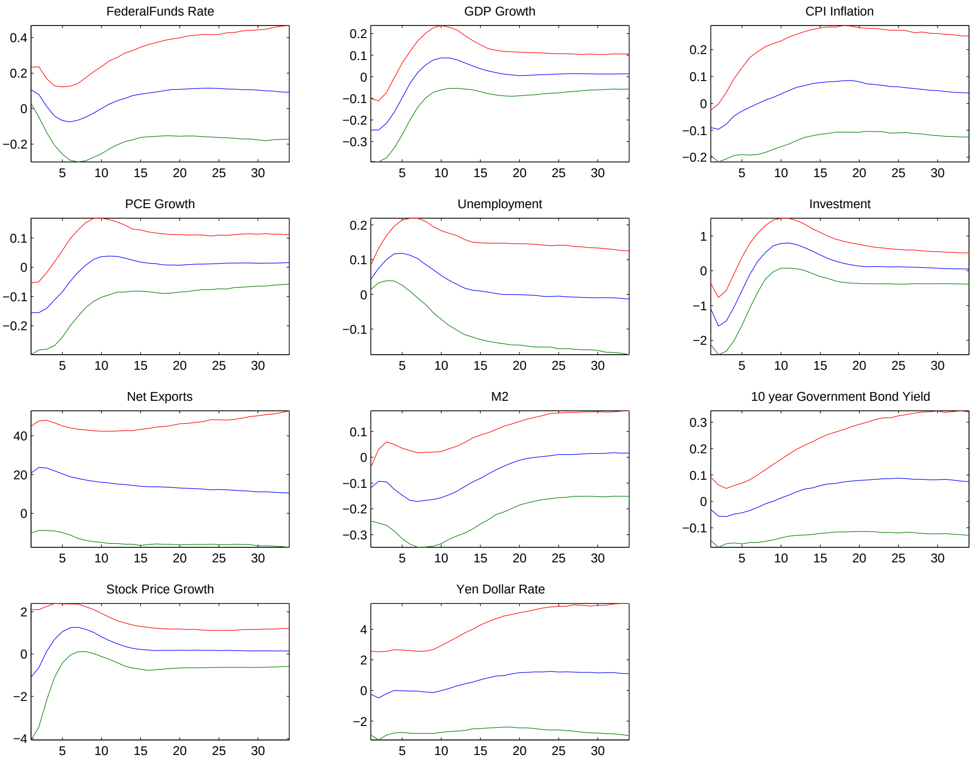


Figure 20. Impulse response to a monetary policy shock using sign restrictions

where Y_t denotes a $T \times N$ matrix of endogenoeus variables, ε_t are the uncorrelated structural shocks and $A_0 A_0' = \Sigma$ where Σ denotes the variance of the reduced form VAR esiduals. Iterating equation 7.1 forward K times we obtain

$$Y_{t+K} = c\sum_{j=0}^{K} B^j + B^j Y_{t-1} + A_0 \sum_{j=0}^{K} B^j \varepsilon_{t+K-j} \tag{7.2}$$

Equation 7.2 shows that the K period ahead forecast Y_{t+K} can be decomposed into components with and without structural shocks. The key point to note is that if a restriction is placed on the future path of the J^{th} variable in Y_t, this implies restrictions on the future shocks to the other variables in the system. This can easily be seen by re-arranging equation 7.2

$$Y_{t+K} - c\sum_{j=0}^{K} B^j - B^j Y_{t-1} = A_0 \sum_{j=0}^{K} B^j \varepsilon_{t+K-j} \tag{7.3}$$

If some of the variables in Y_{t+K} are constrained to follow a fixed path, this implies restrictions on the future innovations on the RHS of equation 7.3. Waggoner and Zha (1999) express these constraints on future innovations as

$$R\varepsilon = r \tag{7.4}$$

where r is a $(M \times k) \times 1$ vector where M are the number of constrained variables and k denotes the number of periods the constraint is applied. The elements of the vector r are the path for the constrained variables minus the unconditional forecast of the constrained variables. R is a matrix with dimensions $(M \times k) \times (N \times k)$. The elements of this matrix are the impulse responses of the constrained variables to the structural shocks ε at horizon $1, 2...k$. The $(N \times k) \times 1$ vector ε contains the constrained future shocks. We give a detailed example showing the structure of these matrices below.

Doan *et al.* (1983) show that a least square solution for the constrained innovations in equation 7.4 is given as

$$\hat{\varepsilon} = R'(R'R)^{-1} r \tag{7.5}$$

With these constrained shocks $\hat{\varepsilon}$ in hand, the conditional forecasts can be calculated by substituting these in equation7.2.

7.1. Calculating conditional forecasts. To see the details of this calculation, consider the following VAR model with two endogenous variables

$$\begin{pmatrix} Y_t \\ X_t \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \begin{pmatrix} Y_{t-1} \\ X_{t-1} \end{pmatrix} + \begin{pmatrix} A_{11} & \\ A_{12} & A_{22} \end{pmatrix} \begin{pmatrix} \varepsilon_{1t} \\ \varepsilon_{2t} \end{pmatrix} \tag{7.6}$$

In addition denote $z_{i,j}^{k}$ as the impulse response of the j^{th} variable at horizon i to the k^{th} structural shock where $k = 1, 2$. Consider forecasting Y_t three periods in the future using the estimated VAR in equation 7.6. However we impose the condition that $\begin{pmatrix} \hat{X}_{t+1} \\ \hat{X}_{t+2} \\ \hat{X}_{t+3} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$, i.e. variable X is fixed at 1 over the forecast horizon. In order to calculate the forecast for Y_t under this condition, the first step involves using equation 7.5 to calculate the restricted structural shocks. Using equation 7.5 requires building the matrices R and r. We now describe the structure of these matrices for our example. First note that the restricted structural shocks (to be calculated) are stacked as

$$\hat{\varepsilon} = \begin{pmatrix} \hat{\varepsilon}_{1t+1} \\ \hat{\varepsilon}_{2t+1} \\ \hat{\varepsilon}_{1t+2} \\ \hat{\varepsilon}_{2t+2} \\ \hat{\varepsilon}_{1t+3} \\ \hat{\varepsilon}_{2t+3} \end{pmatrix} \tag{7.7}$$

The matrix of impulse responses R is built to be compatible with $\hat{\varepsilon}$ (see equation 7.4). In this example, it has the following structure

$$R = \begin{pmatrix} z_{1,2}^{1} & z_{1,2}^{2} & 0 & 0 & 0 & 0 \\ z_{2,2}^{1} & z_{2,2}^{2} & z_{1,2}^{1} & z_{1,2}^{2} & 0 & 0 \\ z_{3,2}^{1} & z_{3,2}^{2} & z_{2,2}^{1} & z_{2,2}^{2} & z_{1,2}^{1} & z_{1,2}^{2} \end{pmatrix} \tag{7.8}$$

The matrix R is made of the response of the constrained variable 2 (i.e. X) to the two structural shocks. The first row of the matrix has the response of X to ε_1 and ε_2 at horizon 1. Note that this row corresponds to the first two elements in $\hat{\varepsilon}$– it links the constrained shocks 1 period ahead to their responses. The second row of R has this impulse response at horizon 2 (first two elements) and then at horizon 1 (third and fourth element). This row corresponds to the forecast two periods ahead and links the structural shocks at horizon 1 and 2 to their respective impulse responses. A similar interpretation applies to the subsequent rows of this matrix.

The matrix r is given as

$$r = \begin{pmatrix} 1 - \tilde{X}_{t+1} \\ 1 - \tilde{X}_{t+2} \\ 1 - \tilde{X}_{t+3} \end{pmatrix} \tag{7.9}$$

where $\tilde{X}_{t+j}$ denotes the unconditional forecast of X. Once these matrices are constructed, the restricted structural shocks are calculated as $\hat{\varepsilon} = R'(R'R)^{-1}r$. These are then used calculate the conditional forecast by substituting them in equation 7.6 and iterating forward.

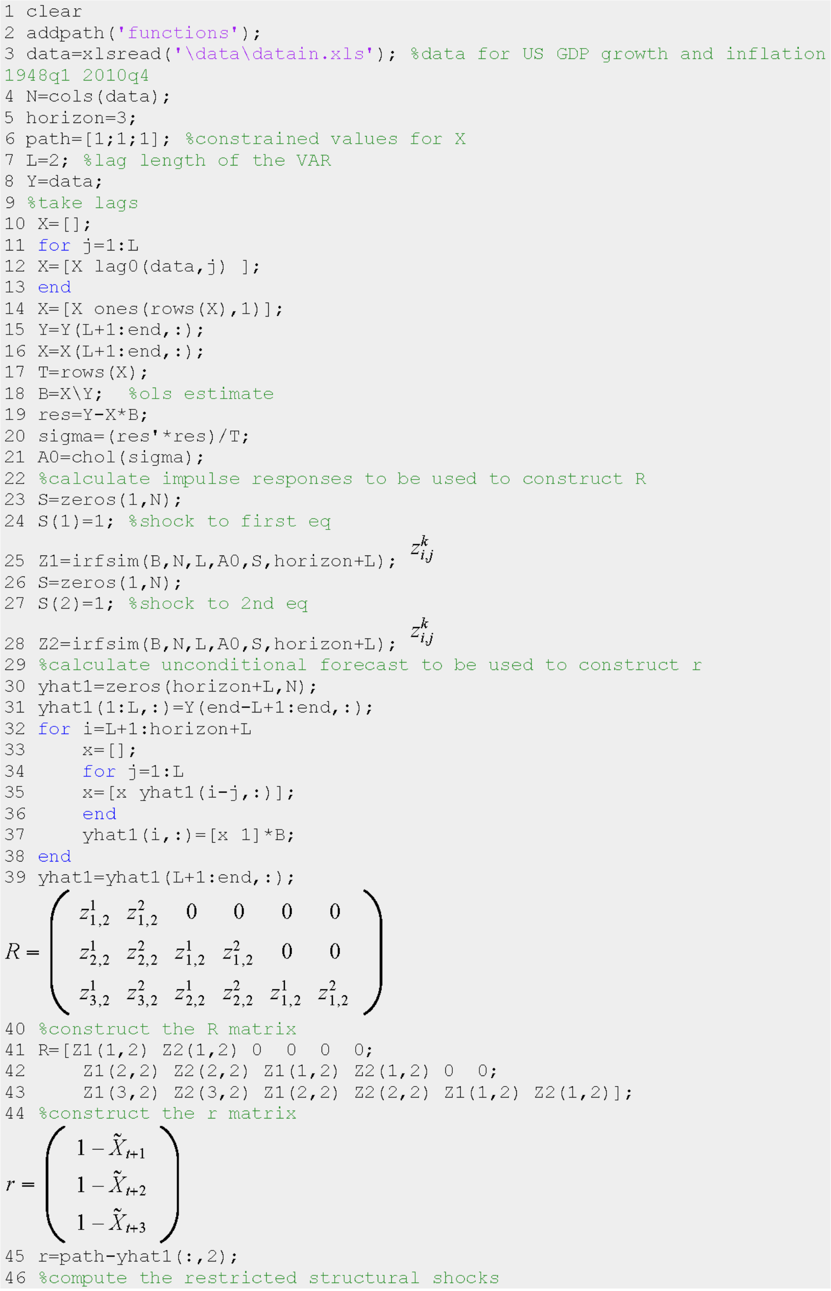
In figures 21 and 22 we show the matlab code for this simple example of calculating a conditional forecast (the matlab file is example8.m). We estimate a VAR(2) model for US GDP growth and inflation and use the estimated VAR to forecast GDP growth 3 periods ahead assuming inflation remains fixed at 1% over the forecast horizon.Lines 18 to 21 of the code estimate the VAR coefficients and error covariance via OLS and calculate A_0 as the Choleski decomposition of the error covariance matrix. As shown in Waggoner and Zha (1999) the choice of identifying restrictions (i.e. the structure of A_0) does not affect the conditional forecast which depends on the reduced form VAR. Therefore it is convenient to use the Choleski decomposition to calculate A_0 for this application. Lines 25 to 28 estimate the impulse response functions $z_{i,j}^{k}$. Line 33 to 39 constructs the unconditional forecast by simulating the estimated VAR model for three periods. Lines 41 to 43 construct the R matrix as specified in equation 7.8. Line 45 constructs the r matrix. With these in hand, the restricted future shocks are calculated on line 50. The conditional forecast is calculated by simulating the VAR using these restricted shocks 50 to 59 of the code with the matlab variable yhat2 holding the conditional forecast.

7.2. Calculating the distribution of the conditional forecast. The main contribution of Waggoner and Zha (1999) is to provide a Gibbs sampling algorithm to construct the distribution of the conditional forecast and thus allow a straigth forward construction of fan charts whn some forecasts are subject to constraints. In particular, Waggoner and Zha (1999) show that the distribution of the restricted future shocks ε is normal with mean $\bar{M}$ and variance $\bar{V}$ where

$$\begin{aligned} \bar{M} &= R'(R'R)^{-1}r \\ \bar{V} &= I - R'(RR')^{-1}R \end{aligned} \tag{7.10}$$

The Gibbs sampling algorithm to generate the forecast distribution proceeds in the following steps

(1) Initialise the VAR coefficients and the A_0 matrix.



```
clear
addpath('functions');
data=xlsread('\data\datain.xls'); %data for US GDP growth and inflation 1948q1 2010q4
N=cols(data);
horizon=3;
path=[1;1;1]; %constrained values for X
L=2; %lag length of the VAR
Y=data;
%take lags
X=[];
for j=1:L
X=[X lag0(data,j) ];
end
X=[X ones(rows(X),1)];
Y=Y(L+1:end,:);
X=X(L+1:end,:);
T=rows(X);
B=X\Y;  %ols estimate
res=Y-X*B;
sigma=(res'*res)/T;
A0=chol(sigma);
%calculate impulse responses to be used to construct R
S=zeros(1,N);
S(1)=1; %shock to first eq
Z1=irfsim(B,N,L,A0,S,horizon+L);
```

$z_{i,j}^{k}$

```
S=zeros(1,N);
S(2)=1; %shock to 2nd eq
Z2=irfsim(B,N,L,A0,S,horizon+L);
```

$z_{i,j}^{k}$

```
%calculate unconditional forecast to be used to construct r
yhat1=zeros(horizon+L,N);
yhat1(1:L,:)=Y(end-L+1:end,:);
for i=L+1:horizon+L
    x=[];
    for j=1:L
    x=[x yhat1(i-j,:)];
    end
    yhat1(i,:)=[x 1]*B;
end
yhat1=yhat1(L+1:end,:);
```

$$R=\begin{pmatrix} z_{1,2}^{1} & z_{1,2}^{2} & 0 & 0 & 0 & 0 \\ z_{2,2}^{1} & z_{2,2}^{2} & z_{1,2}^{1} & z_{1,2}^{2} & 0 & 0 \\ z_{3,2}^{1} & z_{3,2}^{2} & z_{2,2}^{1} & z_{2,2}^{2} & z_{1,2}^{1} & z_{1,2}^{2} \end{pmatrix}$$

```
%construct the R matrix
R=[Z1(1,2) Z2(1,2) 0  0  0  0;
    Z1(2,2) Z2(2,2) Z1(1,2) Z2(1,2) 0  0;
    Z1(3,2) Z2(3,2) Z1(2,2) Z2(2,2) Z1(1,2) Z2(1,2)];
%construct the r matrix
```

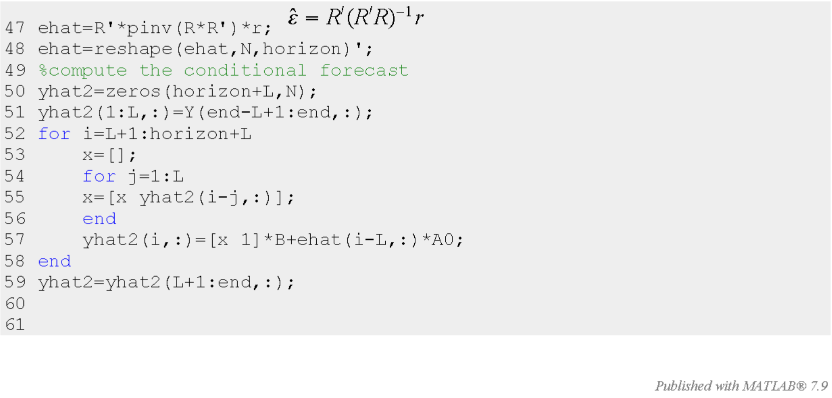
$$r=\begin{pmatrix} 1-\tilde{X}_{t+1} \\ 1-\tilde{X}_{t+2} \\ 1-\tilde{X}_{t+3} \end{pmatrix}$$

```
r=path-yhat1(:,2);
%compute the restricted structural shocks
```

FIGURE 21. Matlab code for computing the conditional forecast

(2) Form the matrices R and r. Draw the restricted structural shocks from the $N(\bar{M},\bar{V})$ distribution where $\bar{M}$ and $\bar{V}$ are calculated as in equation 7.10. This draw of structural shocks is used to calculate the conditional forecast $\hat{Y}_{t+k}$.

(3) Construct the appended dataset $Y_t^* = [Y_t; \hat{Y}_{t+k}]$. This the actual data for the VAR model with the forecasts added to it. The conditional posterior of the VAR coefficients and covariance matrix is constructed using Y_t^* and new values of the coefficients and covariance matrix are drawn. The A_0 matrix can be updated as the Cholesky decomposition of the new draw of the covariance matrix. Note that by using Y_t^* we ensure



```
47 ehat=R'*pinv(R*R')*r;
48 ehat=reshape(ehat,N,horizon)';
49 %compute the conditional forecast
50 yhat2=zeros(horizon+L,N);
51 yhat2(1:L,:)=Y(end-L+1:end,:);
52 for i=L+1:horizon+L
53     x=[];
54     for j=1:L
55     x=[x yhat2(i-j,:)];
56     end
57     yhat2(i,:)=[x 1]*B+ehat(i-L,:)*A0;
58 end
59 yhat2=yhat2(L+1:end,:);
60
61
```

$\hat{\varepsilon} = R'(R'R)^{-1}r$

Published with MATLAB® 7.9

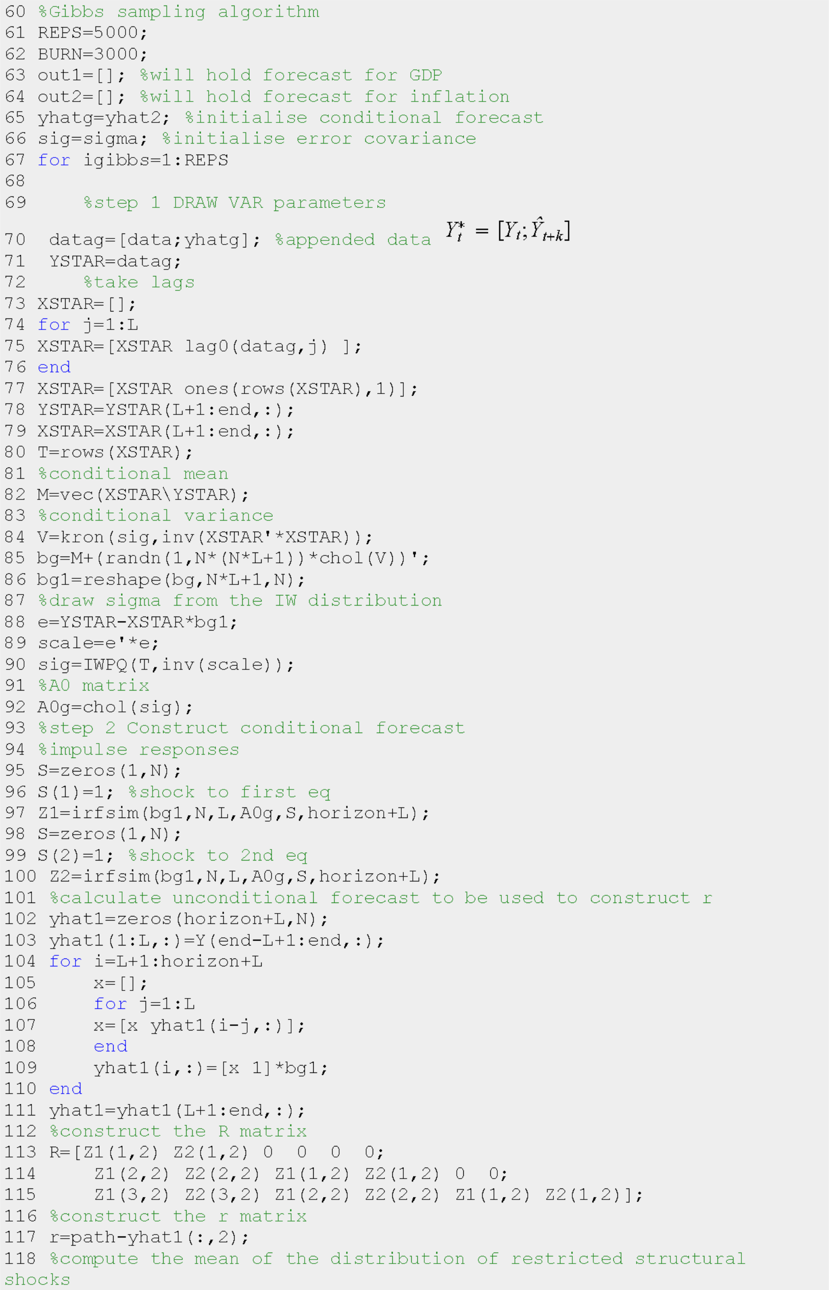
Figure 22. Matlab code for computing the conditional forecast (continued)

that the draws of the VAR parameters take into account the restrictions $R\varepsilon = r$. This procedure therefore accounts for parameter uncertainty and the restrictions imposed on the forecasts by the researcher.

(4) Goto step 2 and repeat M times. The last R draws of $\hat{Y}_{t+k}$ can be used to construct the distribution of the forecast.

In order to demonstrate this algorithm we continue our Matlab example above and calculate the distribution of the GDP growth forecast, leaving the inflation forecast restricted at 1%.

The Matlab code is shown in figures 23 and 24. Note that this is a continuation of the code in the previous example from line 60. We use 5000 Gibbs iterations and discard the first 3000 as burn in. Line 70 of the code constructs the

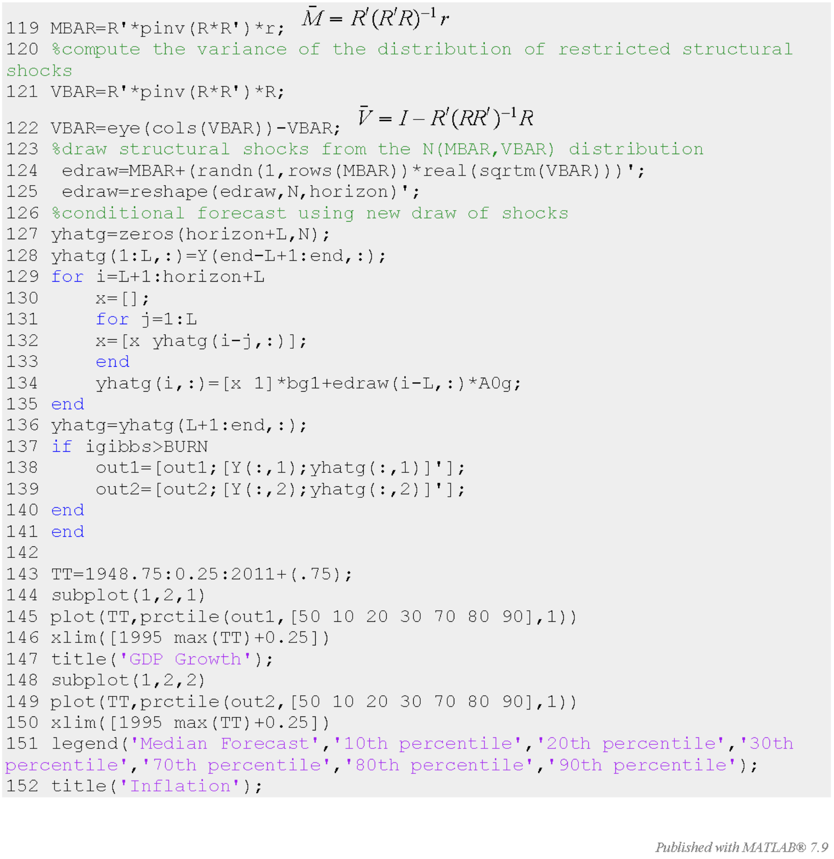


```
60 %Gibbs sampling algorithm
61 REPS=5000;
62 BURN=3000;
63 out1=[]; %will hold forecast for GDP
64 out2=[]; %will hold forecast for inflation
65 yhatg=yhat2; %initialise conditional forecast
66 sig=sigma; %initialise error covariance
67 for igibbs=1:REPS
68
69     %step 1 DRAW VAR parameters
70  datag=[data;yhatg]; %appended data $Y_t^* = [Y_t;\hat{Y}_{t+k}]$
71  YSTAR=datag;
72     %take lags
73 XSTAR=[];
74 for j=1:L
75 XSTAR=[XSTAR lag0(datag,j) ];
76 end
77 XSTAR=[XSTAR ones(rows(XSTAR),1)];
78 YSTAR=YSTAR(L+1:end,:);
79 XSTAR=XSTAR(L+1:end,:);
80 T=rows(XSTAR);
81 %conditional mean
82 M=vec(XSTAR\YSTAR);
83 %conditional variance
84 V=kron(sig,inv(XSTAR'*XSTAR));
85 bg=M+(randn(1,N*(N*L+1))*chol(V))';
86 bg1=reshape(bg,N*L+1,N);
87 %draw sigma from the IW distribution
88 e=YSTAR-XSTAR*bg1;
89 scale=e'*e;
90 sig=IWPQ(T,inv(scale));
91 %A0 matrix
92 A0g=chol(sig);
93 %step 2 Construct conditional forecast
94 %impulse responses
95 S=zeros(1,N);
96 S(1)=1; %shock to first eq
97 Z1=irfsim(bg1,N,L,A0g,S,horizon+L);
98 S=zeros(1,N);
99 S(2)=1; %shock to 2nd eq
100 Z2=irfsim(bg1,N,L,A0g,S,horizon+L);
101 %calculate unconditional forecast to be used to construct r
102 yhat1=zeros(horizon+L,N);
103 yhat1(1:L,:)=Y(end-L+1:end,:);
104 for i=L+1:horizon+L
105     x=[];
106     for j=1:L
107     x=[x yhat1(i-j,:)];
108     end
109     yhat1(i,:)=[x 1]*bg1;
110 end
111 yhat1=yhat1(L+1:end,:);
112 %construct the R matrix
113 R=[Z1(1,2) Z2(1,2) 0  0  0  0;
114     Z1(2,2) Z2(2,2) Z1(1,2) Z2(1,2) 0  0;
115     Z1(3,2) Z2(3,2) Z1(2,2) Z2(2,2) Z1(1,2) Z2(1,2)];
116 %construct the r matrix
117 r=path-yhat1(:,2);
118 %compute the mean of the distribution of restricted structural
shocks
```

Figure 23. Calculating the distribution of the conditional forecast via Gibbs sampling

appended dataset and lines 82 to 90 use this appended data to draw the VAR coefficients and covariance matrix from their conditional distributions. The impulse responses and unconditional forecasts based on this draw of the VAR coefficients and the new A_0 matrix are used to construct the R matrix and the r vector on lines 113 to 117. Lines 119 to 122 construct the mean and variance of the restricted structural shocks $\bar{M}$ and $\bar{V}$. On line 124 we draw the structural shocks from the $N(\bar{M}, \bar{V})$ distrbution and lines 126 to 136 use these to construct the conditional forecast. Once past the burn-in stage the conditional forecasts are saved in the matrices out1 and out2.

Running the code produces figure 25. The left panel of the figure shows the forecast distribution for GDP growth. The right panel shows the forecast for inflation which is restricted at 1% over the forecast horizon.



```
MBAR=R'*pinv(R*R')*r;
%compute the variance of the distribution of restricted structural shocks
VBAR=R'*pinv(R*R')*R;
VBAR=eye(cols(VBAR))-VBAR;
%draw structural shocks from the N(MBAR,VBAR) distribution
 edraw=MBAR+(randn(1,rows(MBAR))*real(sqrtm(VBAR)))';
 edraw=reshape(edraw,N,horizon)';
%conditional forecast using new draw of shocks
yhatg=zeros(horizon+L,N);
yhatg(1:L,:)=Y(end-L+1:end,:);
for i=L+1:horizon+L
    x=[];
    for j=1:L
    x=[x yhatg(i-j,:)];
    end
    yhatg(i,:)=[x 1]*bg1+edraw(i-L,:)*A0g;
end
yhatg=yhatg(L+1:end,:);
if igibbs>BURN
    out1=[out1;[Y(:,1);yhatg(:,1)]'];
    out2=[out2;[Y(:,2);yhatg(:,2)]'];
end
end

TT=1948.75:0.25:2011+(.75);
subplot(1,2,1)
plot(TT,prctile(out1,[50 10 20 30 70 80 90],1))
xlim([1995 max(TT)+0.25])
title('GDP Growth');
subplot(1,2,2)
plot(TT,prctile(out2,[50 10 20 30 70 80 90],1))
xlim([1995 max(TT)+0.25])
legend('Median Forecast','10th percentile','20th percentile','30th percentile','70th percentile','80th percentile','90th percentile');
title('Inflation');
```

Published with MATLAB® 7.9

FIGURE 24. Calculating the distribution of the conditional forecast via Gibbs sampling (continued)

7.3. Extensions and other issues. The example above places restrictions on both structural shocks ε_1 and ε_2 to produce the conditional forecast. In some applications it may be preferable to produce the conditional forecast by placing restrictions only on a subset of shocks. For instances one may wish to restrict ε_1 only in our application. This can be done easily by modifying the R matrix as follows:

$$R = \begin{pmatrix} z^1_{1,2} & 0 & 0 & 0 & 0 & 0 \\ z^1_{2,2} & 0 & z^1_{1,2} & 0 & 0 & 0 \\ z^1_{3,2} & 0 & z^1_{2,2} & 0 & z^1_{1,2} & 0 \end{pmatrix} \tag{7.11}$$

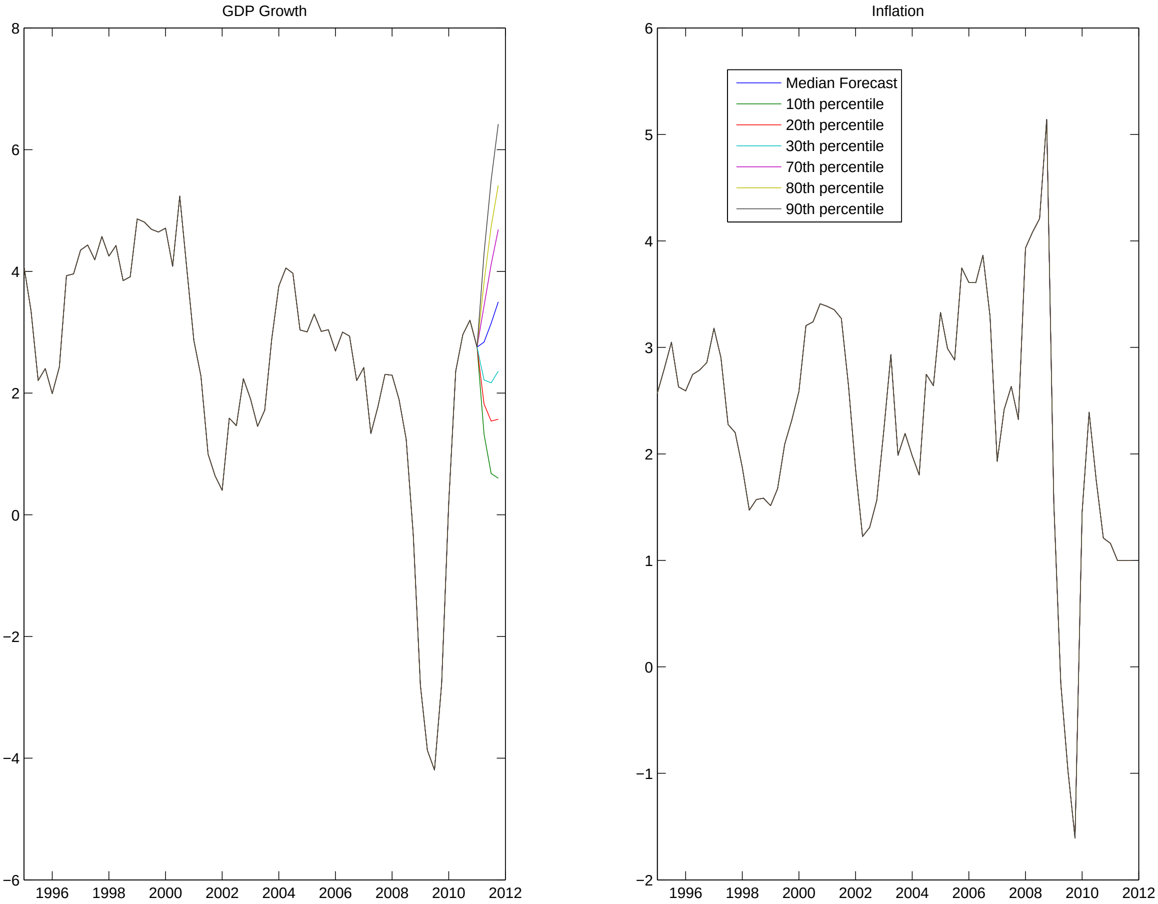


Figure 25. Conditional forecast for US GDP growth

Waggoner and Zha (1999) also discuss a simple method for imposing 'soft conditions' on forecasts– i.e. restricting the forecasts for some variables to lie within a range rather than the 'hard condition' we examine in the example above. Robertson *et al.* (2005) introduce an alternative method to impose 'soft conditions'.

8. Further Reading

- A comprehensive general treatment of Bayesian VARs can be found in Canova (2007) Chapter 10.
- An excellent intuitive explanation of priors and conditional forecasting can be found in Robertson and Tallman (1999).
- A heavily cited article discussing different prior distributions for VARs and methods for calculating posterior distributions is Kadiyala and Karlsson (1997).
- Banbura *et al.* (2007) is an illuminating example of implementing the natural conjugate prior via dummy observations.
- The appendix of Zellner (1971) provides an excellent description of the Inverse Wishart density.

9. Appendix: The marginal likelihood for a VAR model via Gibbs sampling

We can easily apply the method in Chib (1995) to calculate the marginal likelihood for a VAR model. This can then be used to select prior tightness (see for example Carriero *et al.* (2010)) or to choose the lag length and compare different models.

Consider the following VAR model

$$Y_t = c + \sum_{j=1}^{J} b_j Y_{t-j} + v_t, VAR(v_t) = \Sigma \tag{9.1}$$

The prior for the VAR coefficients $B = \{c, b_j\}$ is $P(B)\tilde{}N(\tilde{b}, H)$ and for the covariance matrix $P(\Sigma)\tilde{}IW(\bar{S}, \alpha)$. The posterior distribution of the model parameters $\Phi = B, \Sigma$ is defined via the Bayes rule

$$H(\Phi|Y) = \frac{F(Y|\Phi) \times P(\Phi)}{F(Y)} \tag{9.2}$$

where $\ln F(Y|\Phi) = \frac{-TN}{2}\ln 2\pi + \frac{T}{2}\ln\left|\Sigma^{-1}\right| - 0.5\sum_{i=1}^{T}\left(v_i\Sigma^{-1}v_i'\right)$ is the likelihood function with N representing the number of endogenous variables, $P(\Phi)$ is the joint prior distribution while $F(Y)$ is the marginal likelihood that we want to compute. Chib (1995) suggests computing the marginal likelihood by re-arranging equation 9.2. Note that in logs we can re-write equation 9.2 as

$$\ln F(Y) = \ln F(Y|\Phi) + \ln P(\Phi) - \ln H(\Phi|Y) \tag{9.3}$$

Note that equation 9.3 can be evaluated at any value of the parameters Φ to calculate $\ln F(Y)$. In practice a high density point Φ^* such as the posterior mean or posterior mode is used.

The likelihood function is easy to evaluate. In order to evaluate the priors, the pdf of the normal density and the inverse Wishart is needed. The latter is given in definition 3 above.

Following Chib (1995) the posterior density $H(\Phi^*|Y) = H(B^*, \Sigma^*|Y)$can be factored as

$$H(B^*, \Sigma^*|Y) = H(B^*|\Sigma^*, Y) \times H(\Sigma^*|Y) \tag{9.4}$$

The first term on the RHS of equation 9.4 can be evaluated easily as this is simply the conditional posterior distribution of the VAR coefficients–i.e. a normal distribution with a known mean and covariance matrix.

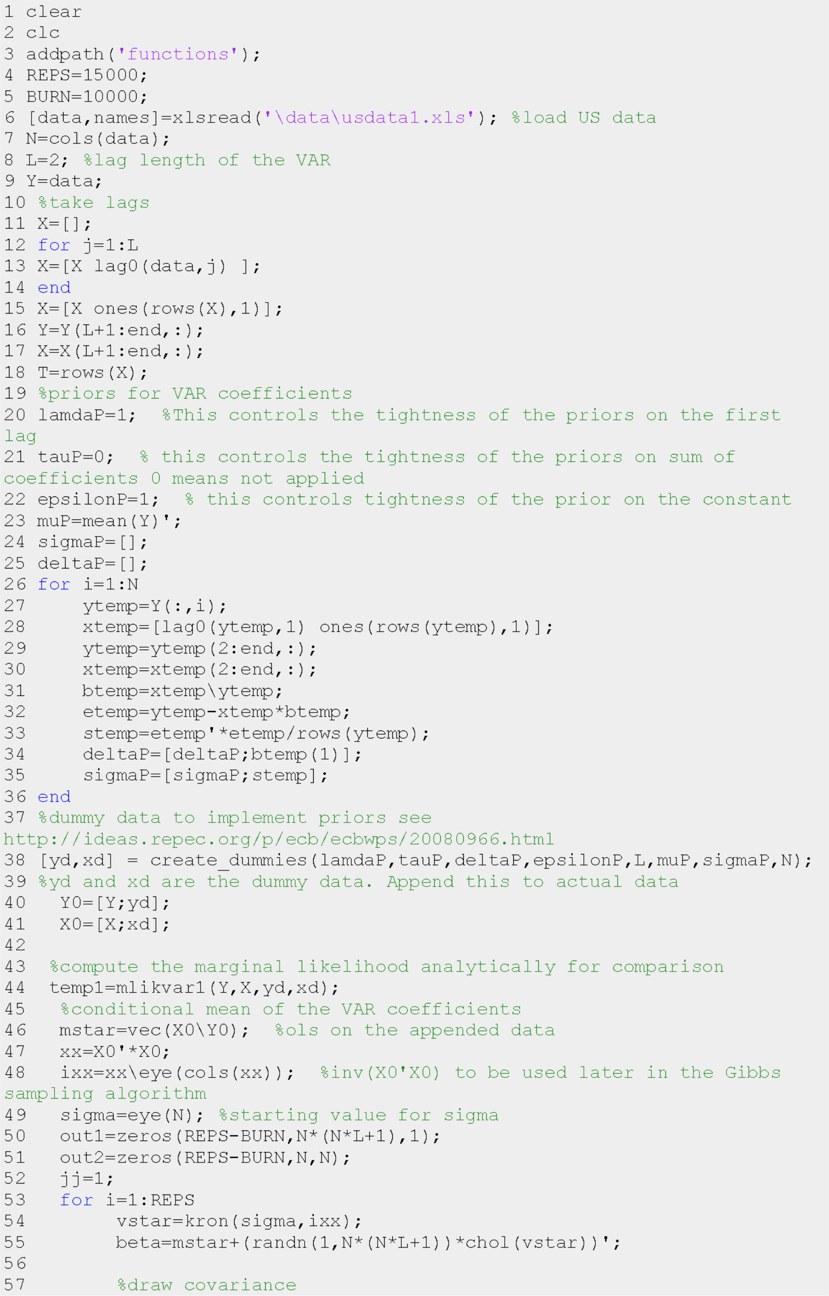
$$\begin{aligned} & H(B^*|\Sigma^*, Y)\tilde{}N(M, V) \\ M &= \left(H^{-1} + \Sigma^{*-1} \otimes X_t'X_t\right)^{-1}\left(H^{-1}\tilde{b}_0 + \Sigma^{*-1} \otimes X_t'X_t\hat{b}\right) \\ V &= \left(H^{-1} + \Sigma^{*-1} \otimes X_t'X_t\right)^{-1} \end{aligned}$$

The second term on the on the RHS of equation 9.4 can be evaluated by noting that

$$H(\Sigma^*|Y) \approx \frac{1}{J}\sum_{j=1}^{J} H(\Sigma^*|B_j, Y) \tag{9.5}$$

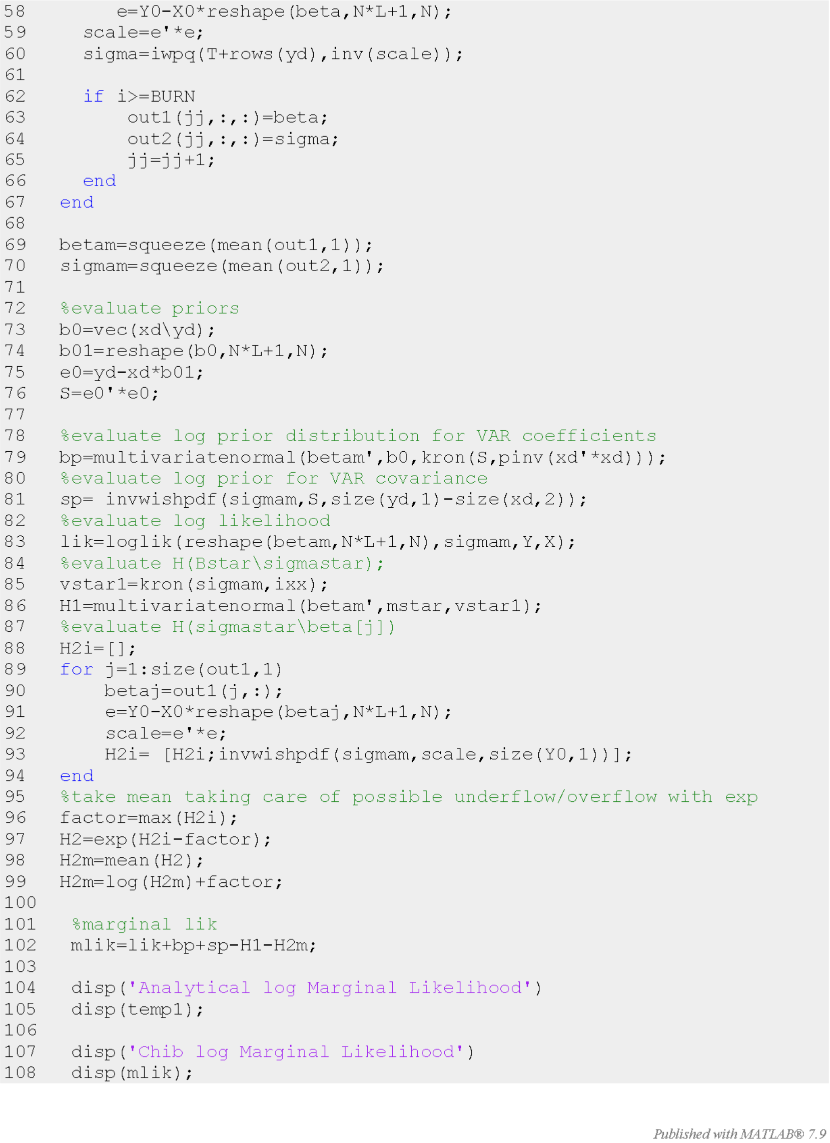
where B_j represent $j = 1...J$ draws of the VAR coefficients from the Gibbs sampler used to estimate the VAR model. Note that $H(\Sigma^*|B_j, Y)$ is the inverse Wishart distribution with scale matrix $\bar{\Sigma} = \bar{S} + v_t'v_t$ and degrees of freedom $T + \alpha$ where the residuals v_t are calculated using the draws B_j.

Figures 26 and 27 show the matlab code for estimating the marginal likelihood in a simple BVAR with a natural conjugate prior implemented via dummy observations. On line 44 we calculate the marginal likelihood analytically for comparison with Chib's estimate. Analytical computation is possible with the natural conjugate prior (see Bauwens *et al.* (1999)), while Chib's estimator can be used more generally. Lines 46 to 67 estimate the VAR model using Gibbs sampling with the posterior means calculated on lines 69 and 70. Lines 73 to 76 calculate the prior moments which are used to evaluate the prior densities on lines 79 and 81. Line 83 evaluates the log likelihood function for the VAR model. Line 86 evaluates the term $H(B^*|\Sigma^*, Y)$. Lines 88 to 99 evaluate the term $H(\Sigma^*|Y) \approx \frac{1}{J}\sum_{j=1}^{J} H(\Sigma^*|B_j, Y)$. These components are used to calculate the marginal likelihood on line 102.



```
clear
clc
addpath('functions');
REPS=15000;
BURN=10000;
[data,names]=xlsread('\data\usdata1.xls'); %load US data
N=cols(data);
L=2; %lag length of the VAR
Y=data;
%take lags
X=[];
for j=1:L
X=[X lag0(data,j) ];
end
X=[X ones(rows(X),1)];
Y=Y(L+1:end,:);
X=X(L+1:end,:);
T=rows(X);
%priors for VAR coefficients
lamdaP=1;  %This controls the tightness of the priors on the first lag
tauP=0;  % this controls the tightness of the priors on sum of coefficients 0 means not applied
epsilonP=1;  % this controls tightness of the prior on the constant
muP=mean(Y)';
sigmaP=[];
deltaP=[];
for i=1:N
    ytemp=Y(:,i);
    xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
    ytemp=ytemp(2:end,:);
    xtemp=xtemp(2:end,:);
    btemp=xtemp\ytemp;
    etemp=ytemp-xtemp*btemp;
    stemp=etemp'*etemp/rows(ytemp);
    deltaP=[deltaP;btemp(1)];
    sigmaP=[sigmaP;stemp];
end
%dummy data to implement priors see http://ideas.repec.org/p/ecb/ecbwps/20080966.html
[yd,xd] = create_dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
%yd and xd are the dummy data. Append this to actual data
  Y0=[Y;yd];
  X0=[X;xd];

 %compute the marginal likelihood analytically for comparison
 temp1=mlikvar1(Y,X,yd,xd);
  %conditional mean of the VAR coefficients
  mstar=vec(X0\Y0);  %ols on the appended data
  xx=X0'*X0;
  ixx=xx\eye(cols(xx));  %inv(X0'X0) to be used later in the Gibbs sampling algorithm
  sigma=eye(N); %starting value for sigma
  out1=zeros(REPS-BURN,N*(N*L+1),1);
  out2=zeros(REPS-BURN,N,N);
  jj=1;
  for i=1:REPS
       vstar=kron(sigma,ixx);
       beta=mstar+(randn(1,N*(N*L+1))*chol(vstar))';

       %draw covariance
```

Figure 26. Marginal Likelihood for a VAR model



```
     e=Y0-X0*reshape(beta,N*L+1,N);
  scale=e'*e;
  sigma=iwpq(T+rows(yd),inv(scale));

  if i>=BURN
      out1(jj,:,:)=beta;
      out2(jj,:,:)=sigma;
      jj=jj+1;
  end
end

betam=squeeze(mean(out1,1));
sigmam=squeeze(mean(out2,1));

%evaluate priors
b0=vec(xd\yd);
b01=reshape(b0,N*L+1,N);
e0=yd-xd*b01;
S=e0'*e0;

%evaluate log prior distribution for VAR coefficients
bp=multivariatenormal(betam',b0,kron(S,pinv(xd'*xd)));
%evaluate log prior for VAR covariance
sp= invwishpdf(sigmam,S,size(yd,1)-size(xd,2));
%evaluate log likelihood
lik=loglik(reshape(betam,N*L+1,N),sigmam,Y,X);
%evaluate H(Bstar\sigmastar);
vstar1=kron(sigmam,ixx);
H1=multivariatenormal(betam',mstar,vstar1);
%evaluate H(sigmastar\beta[j])
H2i=[];
for j=1:size(out1,1)
    betaj=out1(j,:);
    e=Y0-X0*reshape(betaj,N*L+1,N);
    scale=e'*e;
    H2i=  [H2i;invwishpdf(sigmam,scale,size(Y0,1))];
end
%take mean taking care of possible underflow/overflow with exp
factor=max(H2i);
H2=exp(H2i-factor);
H2m=mean(H2);
H2m=log(H2m)+factor;

 %marginal lik
 mlik=lik+bp+sp-H1-H2m;

 disp('Analytical log Marginal Likelihood')
 disp(temp1);

 disp('Chib log Marginal Likelihood')
 disp(mlik);
```

Published with MATLAB® 7.9

Figure 27. Marginal Likelihood for a VAR model (continued)

CHAPTER 3

Gibbs Sampling for state space models

1. Introduction

State space models have become a key tool for research and analysis in central banks. In particular, they can be used to detect structural changes in time series relationships and to extract unobserved components from data (such as the trend in a time series). The state space formulation is also used when calculating the likelihood function for DSGE models.

The classic approach to state space modelling can be computationally inefficient in large scale models as it is based on maximising the likelihood function with respect to all parameters. In contrast, Gibbs sampling proceeds by drawing from conditional distributions which implies dealing with smaller components of the model. In addition, Gibbs sampling provides an approximation to the marginal posterior distribution of the state variable and therefore directly provides a measure of uncertainty associated with the estimate of the state variable. The use of prior information also helps along the dimensions of the model where the data is less informative.

This chapter discusses the Gibbs sampling algorithm for state space models and provides examples of implementing the algorithm in Matlab.

2. Examples of state space models

In general, a state space model consists of the following two equations

$$Y_t = H\beta_t + Az_t + e_t \text{ Observation Equation} \tag{2.1}$$

$$\beta_t = \mu + F\beta_{t-1} + v_t \text{ Transition Equation} \tag{2.2}$$

Consider first the components of the observation equation 2.1. Here Y_t is observed data, H denotes either the right hand side variables or a coefficient matrix depending on the model as discussed below. β_t is the unobserved component or the state variable. z_t denotes exogenous variables with coefficient A. The observation equation, therefore, connects observed data to the unobserved state variable.

Consider the transition equation 2.2. This equation describes the dynamics of the state variable. Note that the order of the AR process in equation 2.2 is restricted to be 1. This condition is not restrictive in a practical sense as any AR(p) process can always be re-written in first order companion form. This is described in the examples below.

Finally, note that we make the following assumptions about the error terms e_t and v_t :

$$VAR(e_t) = R, VAR(v_t) = Q, COV(e_t, v_t) = 0 \tag{2.3}$$

As an example of a state space model consider a time-varying parameter regression: $Y_t = c_t + B_tX_t + e_t$, where the coefficients c_t and B_t are assumed to evolve as random walks. In state-space form this model can be expressed as:

$$Y_t = \overset{H}{\begin{pmatrix} 1 & X_t \end{pmatrix}} \overset{\beta_t}{\begin{pmatrix} c_t \\ B_t \end{pmatrix}} + e_t \text{ Observation Equation} \tag{2.4}$$

$$\overset{\beta_t}{\begin{pmatrix} c_t \\ B_t \end{pmatrix}} = \overset{\beta_{t-1}}{\begin{pmatrix} c_{t-1} \\ B_{t-1} \end{pmatrix}} + \overset{v_t}{\begin{pmatrix} v_{1t} \\ v_{2t} \end{pmatrix}} \text{ Transition Equation} \tag{2.5}$$

where $VAR(e_t) = R, VAR(v_t) = Q, COV(e_t, v_t) = 0$. Note that: (a) In this model $\mu = 0$ and $F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ by assumption and (b) the matrix H in the observation equation represents the right hand side variables of the time-varying regression.

As a second example of a state space model, consider decomposing a series Y_t into two unobserved components, i.e. $Y_t = C_t + \tau_t$. We assume that: (1) the trend component τ_t follows random walk: $\tau_t = \tau_{t-1} + v_{2t}$ and (2) the cyclical component C_t follows an AR(2) process with a constant: i.e. $C_t = c + \rho_1 C_{t-1} + \rho_2 C_{t-2} + v_{1t}$. In state space form this model can be expressed as:

$$Y_t = \overset{H}{\begin{pmatrix} 1 & 1 & 0 \end{pmatrix}} \overset{\beta_t}{\begin{pmatrix} C_t \\ \tau_t \\ C_{t-1} \end{pmatrix}} \text{ Observation Equation} \tag{2.6}$$

$$\overset{\beta_t}{\begin{pmatrix} C_t \\ \tau_t \\ C_{t-1} \end{pmatrix}} = \overset{\mu}{\begin{pmatrix} c \\ 0 \\ 0 \end{pmatrix}} + \overset{F}{\begin{pmatrix} \rho_1 & 0 & \rho_2 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}} \overset{\beta_{t-1}}{\begin{pmatrix} C_{t-1} \\ \tau_{t-1} \\ C_{t-2} \end{pmatrix}} + \overset{v_t}{\begin{pmatrix} v_{1t} \\ v_{2t} \\ 0 \end{pmatrix}} \text{ Transition Equation} \tag{2.7}$$

where $var\begin{pmatrix} v_{1t} \\ v_{2t} \\ 0 \end{pmatrix} = Q = \begin{pmatrix} Q_{1,1} & Q_{1,2} & 0 \\ Q_{1,2} & Q_{2,2} & 0 \\ 0 & 0 & 0 \end{pmatrix}$. Consider the observation equation for this model. Here the matrix H is a coefficient matrix which links the state variables $\begin{pmatrix} C_t \\ \tau_t \\ C_{t-1} \end{pmatrix}$ to Y_t. Note that the observation equation has no error term as we assume that Y_t decomposes exactly into the two components.

The left hand side of the transition equation has the state vector at time t i.e. β_t. The right hand side contains the state vector lagged one period i.e. $\beta_{t-1} = \begin{pmatrix} C_{t-1} \\ \tau_{t-1} \\ C_{t-2} \end{pmatrix}$. The fact that the state vector contains C_{t-1} implies that β_{t-1} contains C_{t-2}. This gives us a way to incorporate the AR(2) process for C_t into the transition equation. In general, if the state variable follows an AR(p) process, this implies adding $p-1$ lags of that state-variable into the state vector β_t.

The first row of the matrix F contains the AR coefficients for C_t with the constant in the corresponding row of μ. The second row forms the random walk process for τ_t. Note that the last row of F contains a 1 (element (1,1)) to link C_{t-1} on the left hand side and C_{t-1} on the right hand side and represents an identity. As a consequence, the last row of v_t equals zero with corresponding zeros in the Q matrix.

As a final example of a state space model, consider a dynamic factor model for a panel of series Y_{it} where $t = 1, 2..T$ represents time and $i = 1, 2...N$ represents the cross-section. Each series in the panel is assumed to depend on a common component F_t i.e. $Y_{it} = B_i F_{t-1} + e_{it}$.We assume that the common unobserved component F_t follows an $AR(2)$ process: $F_t = c + \rho_1 F_{t-1} + \rho_2 F_{t-2} + v_t$. This model has the following state-space representation:

$$\begin{pmatrix} Y_{1t} \\ Y_{2t} \\ . \\ Y_{Nt} \end{pmatrix} = \overset{H}{\begin{pmatrix} B_1 & 0 \\ B_1 & 0 \\ . & 0 \\ B_N & 0 \end{pmatrix}} \overset{\beta_t}{\begin{pmatrix} F_t \\ F_{t-1} \end{pmatrix}} + \overset{e_t}{\begin{pmatrix} e_{1t} \\ e_{2t} \\ . \\ e_{Nt} \end{pmatrix}} \text{ Observation Equation} \tag{2.8}$$

$$\overset{\beta_t}{\begin{pmatrix} F_t \\ F_{t-1} \end{pmatrix}} = \overset{\mu}{\begin{pmatrix} c \\ 0 \end{pmatrix}} + \overset{F}{\begin{pmatrix} \rho_1 & \rho_2 \\ 1 & 0 \end{pmatrix}} \overset{\beta_{t-1}}{\begin{pmatrix} F_{t-1} \\ F_{t-2} \end{pmatrix}} + \overset{v_t}{\begin{pmatrix} v_{1t} \\ 0 \end{pmatrix}} \text{ Transition Equation} \tag{2.9}$$

where $VAR(e_t) = R = \begin{pmatrix} R_{1,1} & 0 & 0 & 0 \\ 0 & R_{2,2} & 0 & 0 \\ 0 & 0 & . & 0 \\ 0 & 0 & 0 & R_{N,N} \end{pmatrix}$ and $VAR(v_t) = Q = \begin{pmatrix} Q_{1,1} & 0 \\ 0 & 0 \end{pmatrix}$. As in the unobserved component model, the matrix H contains the coefficients linking the data Y_{it} to the state variables β_t. The first lag of F_t appears in the state vector because of our assumption that F_t follows an AR(2) process. The transition equation of the system incorporates the AR(2) dynamics for the state variable in companion form with appropriate structures for the μ, F and Q matrices.

See Kim and Nelson (1999) Chapter 2 for further examples of state space models.

3. The Gibbs sampling algorithm for state space models

It is instructive to consider the unknown parameters of our state space system:

$$y_t = H\beta_t + Az_t + e_t,\ VAR(e_t) = R \tag{3.1}$$

$$\beta_t = \mu + F\beta_{t-1} + v_t, VAR(v_t) = Q \tag{3.2}$$

In the observation equation the unknown parameters consist of the elements of H that are not fixed or given as data (for e.g. the coefficients B_i in equation 2.8), the elements of A and the non-zero elements of the covariance matrix R. In the transition equation, the parameters to be estimated are the non-zero and free elements of μ, F and Q.In addition, the state variable β_t is unknown and needs to be estimated.

A Gibbs sampling algorithm for this problem can be discerned by considering the hypothetical case where the state variable β_t is known and observed. If this is the case, then the observation and the transition equations collapse to linear regressions with the conditional posterior distribution of coefficients and variances exactly as in Chapter 1. For example if the common factor F_t in equations 2.8 and 2.9 is known, these equations become a series of linear regressions. Equation 2.8 is then simply N linear regressions while equation 2.9 is an AR(2) model. The conditional distributions of the parameters in this case are known from Chapter 1. This observation indicates the following general Gibbs algorithm for the state space model in equations 3.1 and 3.2.

Step 1 Conditional on β_t, sample H and R from their posterior distributions.
Step 2 Conditional on β_t, sample μ, F and Q from their posterior distributions.
Step 3 Conditional on the parameters of the state space: H, R, μ, F and Q sample the state variable β_t from its conditional posterior distribution.
Step 4 Repeat steps 1 to 3 until convergence is detected.

As emphasised above, steps 1 to 3 are standard and involve linear regressions and/or VARs with known conditional posteriors. *The new step required for the state space model is step 3 where we sample β_t from its conditional posterior distribution.* We turn to a description of the conditional posterior distribution for β_t next.

3.1. The conditional distribution of the state variable. We follow Kim and Nelson (1999) chapter 8 closely in this description. Let $\tilde{\beta}_T = [\beta_1, \beta_2, ...\beta_T]$ i.e. the time series of β from time $1, 2..T$. Similarly, let $\tilde{Y}_T = [Y_1, ...Y_T]$. Recall that we are interested in deriving the conditional posterior distribution $H\left(\tilde{\beta}_T | H, Q, R, F, \mu, \tilde{Y}_T\right)$ i.e. the joint posterior for $\beta_1, \beta_2, ...\beta_T$. As shown by Carter and Kohn (1994), it is convenient to consider a factorisation of the joint density $H\left(\tilde{\beta}_T | \tilde{Y}_T\right)$. Note, we drop the conditioning arguments for simplicity in what follows below.

We can factor $H\left(\tilde{\beta}_T | \tilde{Y}_T\right)$ into the following conditional distributions

$$H\left(\tilde{\beta}_T | \tilde{Y}_T\right) = H\left(\beta_T | \tilde{Y}_T\right) \times H\left(\tilde{\beta}_{T-1} | \beta_T, \tilde{Y}_T\right) \tag{3.3}$$

Note that the right hand side of 3.3 splits $H\left(\tilde{\beta}_T | \tilde{Y}_T\right)$ into the product of the marginal distribution of the state variable at time T and the distribution of the vector $\tilde{\beta}_{T-1} = [\beta_1, \beta_2, ...\beta_{T-1}]$ conditioned on β_T. We can expand the term $H\left(\tilde{\beta}_{T-1} | \beta_T, \tilde{Y}_T\right)$ as $H\left(\tilde{\beta}_{T-1} | \beta_T, \tilde{Y}_T\right) = H\left(\beta_{T-1} |, B_T, \tilde{Y}_T\right) \times H\left(\tilde{\beta}_{T-2} | \beta_T, \beta_{T-1}, \tilde{Y}_T\right)$ where $\tilde{\beta}_{T-2} = [\beta_1, \beta_2, ...\beta_{T-2}]$. Thus:

$$H\left(\tilde{\beta}_T | \tilde{Y}_T\right) = H\left(\beta_T | \tilde{Y}_T\right) \times H\left(\beta_{T-1} | B_T, \tilde{Y}_T\right) \times H\left(\tilde{\beta}_{T-2} | \beta_T, \beta_{T-1}, \tilde{Y}_T\right) \tag{3.4}$$

Continuing in this vein and expanding $H\left(\tilde{\beta}_{T-2} | \beta_T, \beta_{T-1}, \tilde{Y}_T\right) = H\left(\beta_{T-2} |, \beta_T, \beta_{T-1}, \tilde{Y}_T\right) \times$ $H\left(\tilde{\beta}_{T-3} | \beta_T, \beta_{T-1}, \beta_{T-2}, \tilde{Y}_T\right)$

$$H\left(\tilde{\beta}_T | \tilde{Y}_T\right) = H\left(\beta_T | \tilde{Y}_T\right) \times H\left(\beta_{T-1} | B_T, \tilde{Y}_T\right) \times H\left(\beta_{T-2} | \beta_T, \beta_{T-1}, \tilde{Y}_T\right) \times H\left(\tilde{\beta}_{T-3} | \beta_T, \beta_{T-1}, \beta_{T-2}, \tilde{Y}_T\right)$$

Expanding further →

$$\begin{aligned} H\left(\tilde{\beta}_T | \tilde{Y}_T\right) &= H\left(\beta_T | \tilde{Y}_T\right) \times H\left(\beta_{T-1} | B_T, \tilde{Y}_T\right) \times H\left(\beta_{T-2} | \beta_T, \beta_{T-1}, \tilde{Y}_T\right) \\ \times H\left(\beta_{T-3} | \beta_T, \beta_{T-1}, \beta_{T-2}, \tilde{Y}_T\right) & \quad \times ....H\left(\beta_1 | \beta_T, \beta_{T-1}, \beta_{T-2}, ...\beta_2, \tilde{Y}_T\right) \end{aligned} \tag{3.5}$$

As shown in Kim and Nelson (1999) (page 191) expression 3.5 can be simplified by considering the fact that β_T follows a first order AR or Markov process. Because of this Markov property, given $\tilde{Y}_T$ and β_{T-1}, in the term $H\left(\beta_{T-2} |, \beta_T, \beta_{T-1}, \tilde{Y}_T\right)$, β_T contains no additional information for β_{T-2}. This term can therefore be re-written as $H\left(\beta_{T-2} | \beta_{T-1}, \tilde{Y}_T\right)$. Similarly $H\left(\beta_{T-3} | \beta_T, \beta_{T-1}, \beta_{T-2}, \tilde{Y}_T\right)$ can be re-written as $H\left(\beta_{T-3} | \beta_{T-2}, \tilde{Y}_T\right)$.

A similar argument applies to the data vector $\tilde{Y}_T$. For example, in the term $H\left(\beta_{T-2} | \beta_{T-1}, \tilde{Y}_T\right)$, $\tilde{Y}_{T-2} = [Y_1, ...Y_{T-2}]$ contains all the required information for β_{T-2} (given β_{T-1}). Therefore, this term can be re-written as $H\left(\beta_{T-2} | \beta_{T-1}, \tilde{Y}_{T-2}\right)$. Similarly, the term $H\left(\beta_{T-3} | \beta_{T-2}, \tilde{Y}_T\right)$ can be re-written as $H\left(\beta_{T-3} | \beta_{T-2}, \tilde{Y}_{T-3}\right)$.

Given these simplifications we can re-write expression 3.5 as

$$\begin{aligned} H\left(\tilde{\beta}_T | \tilde{Y}_T\right) &= H\left(\beta_T | \tilde{Y}_T\right) \times H\left(\beta_{T-1} | B_T, \tilde{Y}_{T-1}\right) \times H\left(\beta_{T-2} | \beta_{T-1}, \tilde{Y}_{T-2}\right) \times H\left(\beta_{T-3} | \beta_{T-2}, \tilde{Y}_{T-3}\right) \\ & \quad \times ....H\left(\beta_1 | \beta_2, \tilde{Y}_1\right) \end{aligned} \tag{3.6}$$

or more compactly

$$H\left(\tilde{\beta}_T | \tilde{Y}_T\right) = H\left(\beta_T | \tilde{Y}_T\right) \prod_{t=1}^{T-1} H\left(\beta_t | B_{t+1}, \tilde{Y}_t\right) \tag{3.7}$$

The conditional distribution of the state variable is given by expression 3.7.

Assuming that the disturbances of the state space model e_t and v_t are normally distributed:

$$\begin{aligned} H\left(\beta_T|\tilde{Y}_T\right) \tilde{} N(\beta_{T|T}, P_{T|T}) \\ H\left(\beta_t|B_{t+1}, \tilde{Y}_t\right) \tilde{} N(\beta_{t|t,\beta_{t+1}}, P_{t|t,\beta_{t+1}}) \end{aligned} \tag{3.8}$$

where the notation $\beta_{i|j}$ denotes an estimate of β at time i given information upto time j. The two components on the right hand side of expression 3.7 are normal distributions. However, to draw from these distributions, we need to calculate their respective means and variances. To see this calculation we consider each component in turn.

3.1.1. *The mean and variance of* $H\left(\beta_T|\tilde{Y}_T\right)$. We can compute the mean $\beta_{T|T}$ and the variance $P_{T|T}$ using the Kalman filter. The Kalman filter is a recursive algorithm which provides with an estimate of the state variable at each time period, given information up to that time period–i.e. it provides an estimate of $\beta_{t|t}$ and its variance $P_{t|t}$. To estimate the state variable, the Kalman filter requires knowledge of the parameters of the state space H, R, μ, F and Q. These are available in our Gibbs sampling framework from the previous draw of the Gibbs sampler.

The Kalman filter consists of the following equations which are evaluated recursively through time starting from an initial value $\beta_{0|0}$ and $P_{0|0}$.

$$\begin{aligned} \beta_{t|t-1} &= \mu + F\beta_{t-1|t-1} \\ P_{t|t-1} &= FP_{t-1|t-1}F' + Q \\ \eta_{t|t-1} &= Y_t - H\beta_{t|t-1} - Az_t \\ f_{t|t-1} &= HP_{t|t-1}H' + R \\ \beta_{t|t} &= \beta_{t|t-1} + K\eta_{t|t-1} \\ P_{t|t} &= P_{t|t-1} - KHP_{t|t-1} \end{aligned} \tag{3.9}$$

where $K = P_{t|t-1}H'f_{t|t-1}^{-1}$. Running these equations from $t = 1, 2...T$ delivers $\beta_{T|T}$ and $P_{T|T}$ at the end of the recursion.

Consider the intuition behind each equation of the Kalman filter. The first and the second equation are referred to as the prediction equations. The first equation $\beta_{t|t-1} = \mu + F\beta_{t-1|t-1}$ simply predicts the value of the state variable one period ahead using the transition equation of the model. This equation can be easily derived by taking the expected value of the transition equation i.e. $E\left(\mu + F\beta_{t-1} + v_t|\bar{Y}_{t-1}\right) = \mu + F\beta_{t-1|t-1}$ where $\bar{Y}_t = \{Y_t, z_t\}$. This follows by noting that $E(v_t|\bar{Y}_{t-1}) = 0$ and $E\left(\beta_{t-1}|\bar{Y}_{t-1}\right) = \beta_{t-1|t-1}$. The second equation is simply the estimated variance of the state variable given information at time $t-1$ and can be derived by taking the variance of β_t (i.e. calculating $E\left[\beta_t - E\left(\beta_{t-1}|\bar{Y}_{t-1}\right)\right]$). The prediction equations of the Kalman filter threfore produce an estimate of the state variable simply based on the parameters of the transition equation. Note that the observed data $\bar{Y}_t$ is not involved upto this point. The third equation of the Kalman filter calculates the prediction error $\eta_{t|t-1} = Y_t - H\beta_{t|t-1} - Az_t$. The fourth equation calculates the variance of the prediction error $f_{t|t-1} = HP_{t|t-1}H' + R$. This equation can be derived by calculating $E\left(Y_t - H\beta_{t|t-1} - Az_t\right)^2$.

The final two equations of the Kalman filter are referred to as the updating equations. These equations update the initial estimates $\beta_{t|t-1}$ and $P_{t|t-1}$ using the information contained in the prediction error $\eta_{t|t-1}$. Note that $K = P_{t|t-1}H'f_{t|t-1}^{-1}$ (referred to as the Kalman gain) can be thought of as the weight attached to prediction error. The updating equations can be derived by considering the formula for updating a linear projection.

As shown in Hamilton (1994) page 99 this formula is given as

$$\hat{P}(Y_3|Y_2, Y_1) = \hat{P}(Y_3|Y_1) + H_{32}H_{22}^{-1}\left[Y_2 - \hat{P}(Y_2|Y_1)\right] \tag{3.10}$$

In equation 3.10 we consider the hypothetical case where we have three variables $Y_{1,}Y_2$ and Y_3. Originally we were forecasting Y_3 based on Y_1 i.e. the term $\hat{P}(Y_3|Y_1)$ and we want to update this projection using the variable Y_2. According to equation 3.10 the updated projection is the sum of $\hat{P}(Y_3|Y_1)$ and the error in predicting Y_2 where the projection of Y_2 is based on Y_1. The weight attached to this prediction error is $H_{32}H_{22}^{-1}$ where H_{ij} is the covariance between $Y_{i,}Y_j$. Consider first the intuition behind the prediction error $Y_2 - \hat{P}(Y_2|Y_1)$. If the information contained in Y_1 and Y_2 is very similar, it is likely that $\hat{P}(Y_2|Y_1)$ and Y_2 will be similar and hence the extra unanticipated information contained in Y_2 will be limited. The weight attached to this extra information $H_{32}H_{22}^{-1}$ can be interpreted as the regression coefficient between Y_3 and Y_2. A larger value of this coefficient implies that the information contained in Y_2 receives a larger weight when updating the forecast $\hat{P}(Y_3|Y_1)$.

In our application, if we let $Y_3 = \beta_t$, $Y_2 = Y_t$ and $Y_1 = z_t, Y_{t-1} \rightarrow$

$$\begin{aligned} \beta_{t|t} &= \beta_{t|t-1} + E\left[\left(\beta_t - \beta_{t|t-1}\right)\left(Y_t - Y_{t|t-1}\right)'\right] \times \\ & E\left[\left(Y_t - Y_{t|t-1}\right)\left(Y_t - Y_{t|t-1}\right)'\right]^{-1} \times \eta_{t|t-1} \end{aligned} \tag{3.11}$$

where $Y_{t|t-1} = H\beta_{t|t-1} + Az_t$. Note that the term $E\left[\left(Y_t - Y_{t|t-1}\right)\left(Y_t - Y_{t|t-1}\right)'\right]$ is simply the forecast error variance $f_{t|t-1}$. Also note that $Y_t - Y_{t|t-1} = (H\beta_t + Az_t + e_t) - \left(H\beta_{t|t-1} + Az_t\right) = H\left(\beta_t - \beta_{t|t-1}\right) + e_t$. Thus

$$E\left[\left(\beta_t - \beta_{t|t-1}\right)\left(Y_t - Y_{t|t-1}\right)'\right] = E\left[\left(\beta_t - \beta_{t|t-1}\right)\left(H\left(\beta_t - \beta_{t|t-1}\right) + e_t\right)'\right]$$

$\rightarrow$

$$E\left[\left(\beta_t - \beta_{t|t-1}\right)\left(H\left(\beta_t - \beta_{t|t-1}\right)\right)'\right] = P_{t|t-1}H'$$

Substituting these in equation 3.11 produces the updating equation $\beta_{t|t} = \beta_{t|t-1} + K\eta_{t|t-1}$. A similar derivation can be used to obtain the final updating equation $P_{t|t}$ as shown in Hamilton (1994) page 380. Finally note that the likelihood function is given as a by product of the Kalman filter recursions as $-\frac{1}{2}\sum_{t=1}^{T}\ln 2\pi^n\left|f_{t|t-1}\right| - \frac{1}{2}\sum_{t=1}^{T}\eta'_{t|t-1}f_{t|t-1}^{-1}\eta_{t|t-1}$.

For a stationary transition equation, the initial values for the Kalman filter recursions $\beta_{0|0}$ and $P_{0|0}$ are given as the unconditional mean and variance. That is $\beta_{0|0} = (I_k - F)^{-1}\mu$ and $vec(P_{0|0}) = (I - F\otimes F)^{-1}vec(Q)$. If the transition equation of the system is non-stationary (for e.g. if the state variable follows a random walk) the unconditional moments do not exist. In this case $\beta_{0|0}$ can be set arbitrarily. $P_{0|0}$ is then set as a diagonal matrix with large diagonal entries reflecting uncertainty around this initial guess.

To recap, we evaluate the equations of the Kalman filter given in 3.9 for periods $t = 1...T$. The final recursion delivers $\beta_{T|T}$ and $P_{T|T}$ the mean and variance of $H\left(\beta_T|\tilde{Y}_T\right)$.

3.1.2. *The mean and variance of* $H\left(\beta_t|\beta_{t+1}, \tilde{Y}_t\right)$. The mean and variance of the conditional distribution $H\left(\beta_t|\beta_{t+1}, \tilde{Y}_t\right)$ can also be derived using the Kalman filter updating equations. As discussed in Kim and Nelson (1999) page 192, deriving the mean $\beta_{t|t,\beta_{t+1}}$ can be thought of as updating $\beta_{t|t}$ (the kalman filter estimate of the state variable) for information contained in β_{t+1}which we treat as observed (for e.g. at time $T-1$, β_{t+1} is given using a draw from $H\left(\beta_T|\tilde{Y}_T\right)$ which we discussed above) Note that this task fits into the framework of the updating equations discussed in the previous section as we are updating an estimate using new information. In other words, the updating equations of the Kalman filter apply with parameters and the prediction error chosen to match our problem.

For the purpose of this derivation we can consider a state space system with the observation equation:

$$\beta_{t+1} = \mu + F\beta_t + v_{t+1} \tag{3.12}$$

This implies that the prediction error is given by $\eta^*_{t+1|t} = \beta_{t+1} - \mu + F\beta_{t|t}$. The forecast error variance is given by $f^*_{t+1|t} = FP_{t|t}F' + Q$. Note also that for this observation equation, the matrix that relates the state variable β_t to the observed data β_{t+1} is $H^* = F$. With these definitions in hand we can simply use the updating equations of the Kalman filter. That is

$$\beta_{t|t,\beta_{t+1}} = \beta_{t|t} + K^*\left(\beta_{t+1} - \mu + F\beta_{t|t}\right) \tag{3.13}$$

$$P_{t|t,\beta_{t+1}} = P_{t|t} - K^*H^*P_{t|t} \tag{3.14}$$

where the gain matrix is $K^* = P_{t|t}H^{*\prime}f^{*-1}_{t+1|t}$.

Equations 3.13 and 3.14 are evaluated backwards in time starting from period $T-1$ and iterating backwards to period 1. This recursion consists of the following steps:

Step 1 Run the Kalman filter from $t = 1...T$ to obtain the mean $\beta_{T|T}$ and the variance $P_{T|T}$ of the distribution $H\left(\beta_T|\tilde{Y}_T\right)$. Also save $\beta_{t|t}$ and $P_{t|t}$ for $t = 1...T$. Draw β_T from the normal distribution with mean $\beta_{T|T}$ and the variance $P_{T|T}$. Denote this draw by $\hat{\beta}_T$

Step 2 At time $T-1$, use 3.13 to calculate $\beta_{T-1|T-1,\beta_T} = \beta_{T-1|T-1} + K^*\left(\hat{\beta}_T - \mu + F\beta_{T-1|T-1}\right)$ where $\beta_{T-1|T-1}$ is the Kalman filter estimate of the state variable (from step 1) at time $T-1$. Use equation 3.14 to calculate $P_{t|t,\beta_{t+1}}$. Draw $\hat{\beta}_{T-1}$ from the normal distribution with mean $\beta_{T-1|T-1,\beta_T}$ and variance $P_{t|t,\beta_{t+1}}$.

Step 3 Repeat step 2 for $t = T-2, T-3, ...1$.

This backward recursion (The Carter and Kohn algorithm) delivers a draw of $\tilde{\beta}_T = [\beta_1, \beta_2, ...\beta_T]$ from its conditional posterior distribution.

A minor modification to this algorithm is required if the matrix Q is singular (see the example of the state space model given in equation 2.6). In this case we evaluate equations 3.13 and 3.14 using $\bar{F}$ instead of F, $\bar{Q}$ instead of Q and $\bar{\mu}$ instead of μ where $\bar{F}, \bar{Q}, \bar{\mu}$ correspond to the non-singular block of Q. In the example given in equation 2.6 above $\bar{\mu} = \begin{pmatrix} c \\ 0 \end{pmatrix}$, $\bar{F} = \begin{pmatrix} \rho_1 & 0 & \rho_2 \\ 0 & 1 & 0 \end{pmatrix}$ and $\bar{Q} = \begin{pmatrix} Q_{1,1} & Q_{1,2} \\ Q_{1,2} & Q_{2,2} \end{pmatrix}$.

3.2. The Gibbs sampling algorithm. We can now re-state the Gibbs alogrithm for the state space model in equations 3.1 and 3.2.

Step 1 Conditional on β_t, sample H and R from their posterior distributions.

Step 2 Conditional on β_t, sample μ, F and Q from their posterior distributions.

Step 3 Conditional on the parameters of the state space: H, R, μ, F and Q sample the state variable β_t from its conditional posterior distribution. That is, run the Kalman filter to obtain $\beta_{t|t}$ and $P_{t|t}$ for $t = 1...T$ and draw β_T. Use equations 3.13 and 3.14 to draw $\beta_1, \beta_2, ...\beta_{T-1}$.

Step 4 Repeat steps 1 to 3 until convergence is detected.

Implementing this Gibbs sampling algorithm therefore requires programming the Kalman filter and equations 3.13 and 3.14 in matlab. The remainder of this chapter describes this task with the help of several examples.

4. The Kalman filter in Matlab

To discuss the implementation of the Kalman filter in Matlab we will consider the following time varying parameter model as an example

$$\begin{aligned} Y_t &= X_t\beta_t + v_t \\ \beta_t &= \mu + F\beta_{t-1} + e_t \\ VAR(v_t) &= R \\ VAR(e_t) &= Q \end{aligned} \tag{4.1}$$

where Y_t is a $T \times 1$ matrix containing the dependent variable, X_t is a $T \times 1$ matrix containg the regressor with time-varying coefficient β_t. For the moment we assume that the parameters of this state space model μ, F, R and Q are known and we are interested in estimating the time-varying coefficient β_t the state variable.

Figures 1 and 2 show the matlab code for the Kalman filter equations (Example1.m). Lines 7 to 20 of the file generate artificial data for Y_t (see equation 4.1) assuming that $\mu = 0, F = 1, Q = 0.001, R = 0.01$. Line 21 starts the Kalman filter by setting up the initial conditions for the state variable β_t. Line 22 assumes that $\beta_{0|0} = 0$ and line 23 sets $p_{0|0}$ the variance of the initial state equal to 1. The Kalman filter starts with $\beta_{t-1|t-1} = \beta_{0|0}$ and $P_{t-1|t-1} = P_{0|0}$ (lines 27 and 28) and then iterates through the sample (loop starts on line 29). Line 32 is the first equation of the prediction step of the Kalman filter $\beta_{t|t-1} = \mu + F\beta_{t-1|t-1}$. Line 33 calculates the variance of $\beta_{t|t-1}$ using the equation $P_{t|t-1} = FP_{t-1|t-1}F' + Q$. Line 34 calculates the fitted value of Y_t for that time period as $X_t\beta_{t|t-1}$ and the next line calculates the prediction error for that time period $\eta_{t|t-1} = Y_t - X_t\beta_{t|t-1}$. Line 36 calculates the variance of the prediction error $f_{t|t-1} = X_tP_{t|t-1}X_t' + R$. Line 38 starts the updating step of the Kalman filter by calculating the Kalman gain $K_t = P_{t|t-1}X_t'f_{t|t-1}^{-1}$. Line 39 updates the the estimate of the state variable based on information contained in the prediction error $\beta_{t|t} = \beta_{t|t-1} + K_t\eta_{t|t-1}$ where this information is weighted by the Kalman gain. The final equation of the Kalman filter (line 40) updates the variance of the state variable $P_{t|t} = P_{t|t-1} - K_tX_tp_{t|t-1}$

Figure 3 shows the estimates of β_t obtained using the Kalman filter. These closely match the assumed true value of β_t.

5. The Carter and Kohn algorithm in Matlab

Recall that that the conditional distribution of the state variable $\tilde{\beta}_T = [\beta_1, \beta_2, ...\beta_T]$ is

$$H\left(\tilde{\beta}_T\right) = H\left(\beta_T, \tilde{Y}_T\right) \prod_{t=1}^{T-1} H\left(\beta_t | \beta_{t+1}, \tilde{Y}_t\right) \tag{5.1}$$

As discussed above, this implies that

$$\begin{aligned} &\beta_T \tilde{\ } N(\beta_{T|T}, p_{T|T}) \\ &\beta_t|\beta_{t+1} \tilde{\ } N(\beta_{t|t,\beta_{t+1}}, P_{t|t,\beta_{t+1}}) \end{aligned} \tag{5.2}$$

As described above, the mean and variance in $\beta_T \tilde{\ } N(\beta_{T|T}, P_{T|T})$ is delivered by the Kalman filter at time $t = T$. The computation of the mean and variance in $N(\beta_{t|t,\beta_{t+1}}, P_{t|t,\beta_{t+1}})$ requires the updating equations 3.13 and 3.14. Written in full these are:

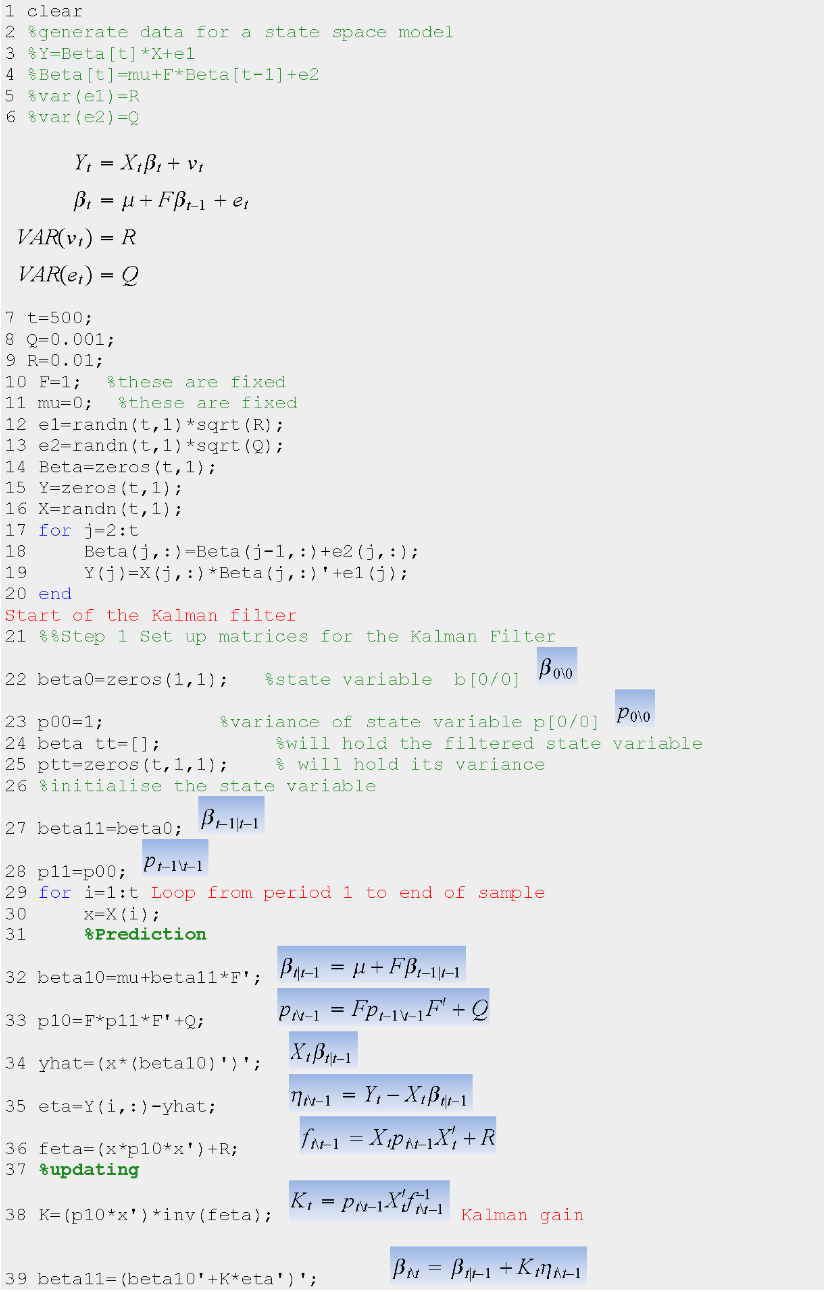
$$\beta_{t|t,\beta_{t+1}} = \beta_{t|t} + P_{t|t}F'\left(Fp_{t|t}F' + Q\right)^{-1}\left(\beta_{t+1} - \mu - F\beta_{t|t}\right) \tag{5.3}$$

$$P_{t|t,\beta_{t+1}} = P_{t|t} - P_{t|t}F'\left(FP_{t|t}F' + Q\right)^{-1}FP_{t|t} \tag{5.4}$$

These are computed going backwards in time from period $t - 1$ to 1. We now turn to the implementation of the algorithm in Matlab

Figures 4 and 5 show the matlab code for the Carter and Kohn algorithm for artificial data on the state space model shown in equation 4.1) assuming that $\mu = 0, F = 1, Q = 0.001, R = 0.01$ (See example2.m). As alluded to above, the algorithm works in two steps. As a first step we run the Kalman filter to obtain $\beta_{T|T}, p_{T|T}$. Lines 21 to 44 of the code are the Kalman filter equations and are identical to the example above. Note that the matrix ptt saves $p_{t|t}$ for each time period.[1]The matrix beta_tt saves $\beta_{t|t}$ for each time period. Line 47 specifies an empty matrix to hold the draw of β_t. Line 48 specifies a $T \times 1$ vector from the $N(0, 1)$ distribution to be used below. Line 51 draws

[1] This is set up as a three dimensional matrix where the first dimension is time and the second two dimensions are the rows and columns of the covariance matrix $p_{t\backslash t}$. In this example this matrix has dimension $500 \times 1 \times 1$.



```
1 clear
2 %generate data for a state space model
3 %Y=Beta[t]*X+e1
4 %Beta[t]=mu+F*Beta[t-1]+e2
5 %var(e1)=R
6 %var(e2)=Q
```

$$Y_t = X_t\beta_t + v_t$$
$$\beta_t = \mu + F\beta_{t-1} + e_t$$
$$VAR(v_t) = R$$
$$VAR(e_t) = Q$$

```
7 t=500;
8 Q=0.001;
9 R=0.01;
10 F=1;  %these are fixed
11 mu=0;  %these are fixed
12 e1=randn(t,1)*sqrt(R);
13 e2=randn(t,1)*sqrt(Q);
14 Beta=zeros(t,1);
15 Y=zeros(t,1);
16 X=randn(t,1);
17 for j=2:t
18     Beta(j,:)=Beta(j-1,:)+e2(j,:);
19     Y(j)=X(j,:)*Beta(j,:)'+e1(j);
20 end
Start of the Kalman filter
21 %%Step 1 Set up matrices for the Kalman Filter

22 beta0=zeros(1,1);   %state variable  b[0/0]
```

$$\beta_{0\backslash 0}$$

```
23 p00=1;          %variance of state variable p[0/0]
```

$$p_{0\backslash 0}$$

```
24 beta tt=[];          %will hold the filtered state variable
25 ptt=zeros(t,1,1);    % will hold its variance
26 %initialise the state variable

27 beta11=beta0;
```

$$\beta_{t-1|t-1}$$

```
28 p11=p00;
```

$$p_{t-1\backslash t-1}$$

```
29 for i=1:t Loop from period 1 to end of sample
30     x=X(i);
31     %Prediction

32 beta10=mu+beta11*F';
```

$$\beta_{t|t-1} = \mu + F\beta_{t-1|t-1}$$

```
33 p10=F*p11*F'+Q;
```

$$p_{t\backslash t-1} = Fp_{t-1\backslash t-1}F' + Q$$

```
34 yhat=(x*(beta10)')';
```

$$X_t\beta_{t|t-1}$$

```
35 eta=Y(i,:)-yhat;
```

$$\eta_{t\backslash t-1} = Y_t - X_t\beta_{t|t-1}$$

```
36 feta=(x*p10*x')+R;
```

$$f_{t\backslash t-1} = X_t p_{t\backslash t-1} X_t' + R$$

```
37 %updating

38 K=(p10*x')*inv(feta);
```

$$K_t = p_{t\backslash t-1} X_t' f_{t\backslash t-1}^{-1}$$

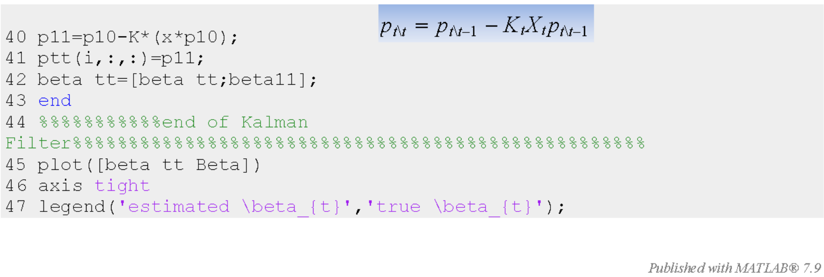
Kalman gain

```
39 beta11=(beta10'+K*eta')';
```

$$\beta_{t\backslash t} = \beta_{t|t-1} + K_t\eta_{t\backslash t-1}$$

FIGURE 1. The Kalman filter in Matlab

from $H\left(\beta_T, \tilde{Y}_T\right)$ where the mean of this distribution is $\beta_{T|T}$ and the variance is $P_{T|T}$ where both these quantities are delivered by the kalman filter and saved as the last row of beta_tt and ptt respectively. Line 53 starts the second step of the Carter and Kohn algorithm and begins a loop going backwards from period T-1 to1. Line 55 computes the mean of $H\left(\beta_t|\beta_{t+1}, \tilde{Y}_t\right)$ using the expression $\beta_{t|t,\beta_{t+1}} = \beta_{t|t} + P_{t|t}F'\left(FP_{t|t}F' + Q\right)^{-1}\left(\beta_{t+1} - \mu - F\beta_{t|t}\right)$. Note that the term β_{t+1} is the draw of β_t one period in the future. Line 56 calculates the variance of $H\left(\beta_t|\beta_{t+1}, \tilde{Y}_t\right)$ using the expression $P_{t|t,\beta_{t+1}} = P_{t|t} - P_{t|t}F'\left(Fp_{t|t}F' + Q\right)^{-1} FP_{t|t}$.Line 57 draws the state variable from a normal distribution using this mean and variance.



```
p11=p10-K*(x*p10);
ptt(i,:,:)=p11;
beta tt=[beta tt;beta11];
end
%%%%%%%%%%%end of Kalman
Filter%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot([beta tt Beta])
axis tight
legend('estimated \beta_{t}','true \beta_{t}');
```

$p_{t\backslash t} = p_{t\backslash t-1} - K_t X_t p_{t\backslash t-1}$
Published with MATLAB® 7.9

FIGURE 2. The Kalman filter in Matlab continued

Figure 6 plots the result of running this code and shows the draw for β_t.

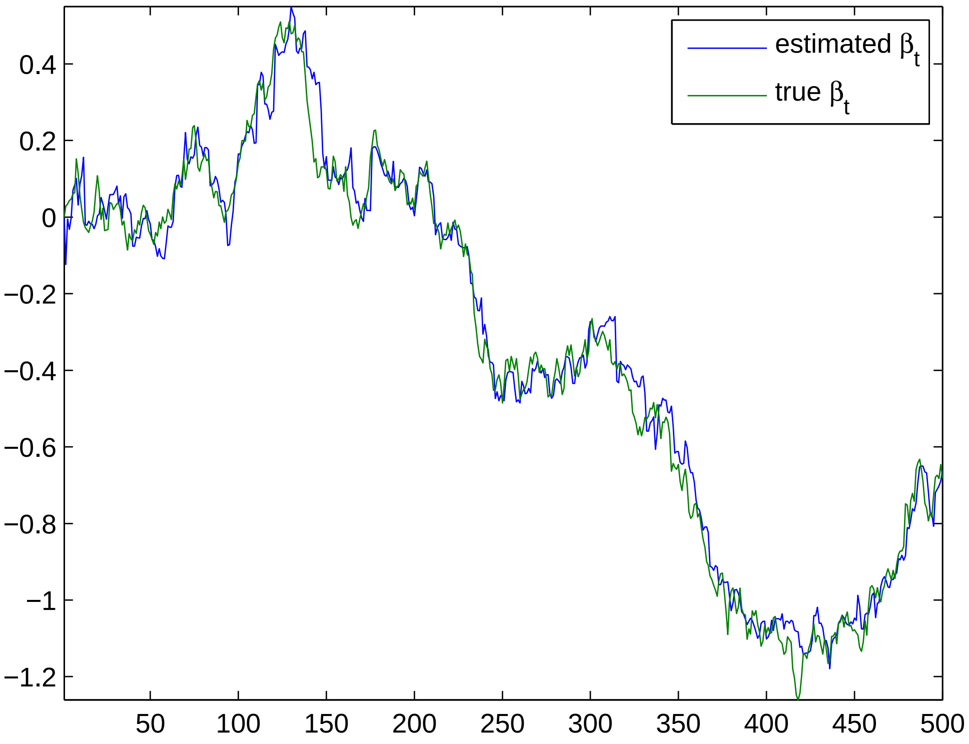


FIGURE 3. Estimates of β_t from the Kalman filter

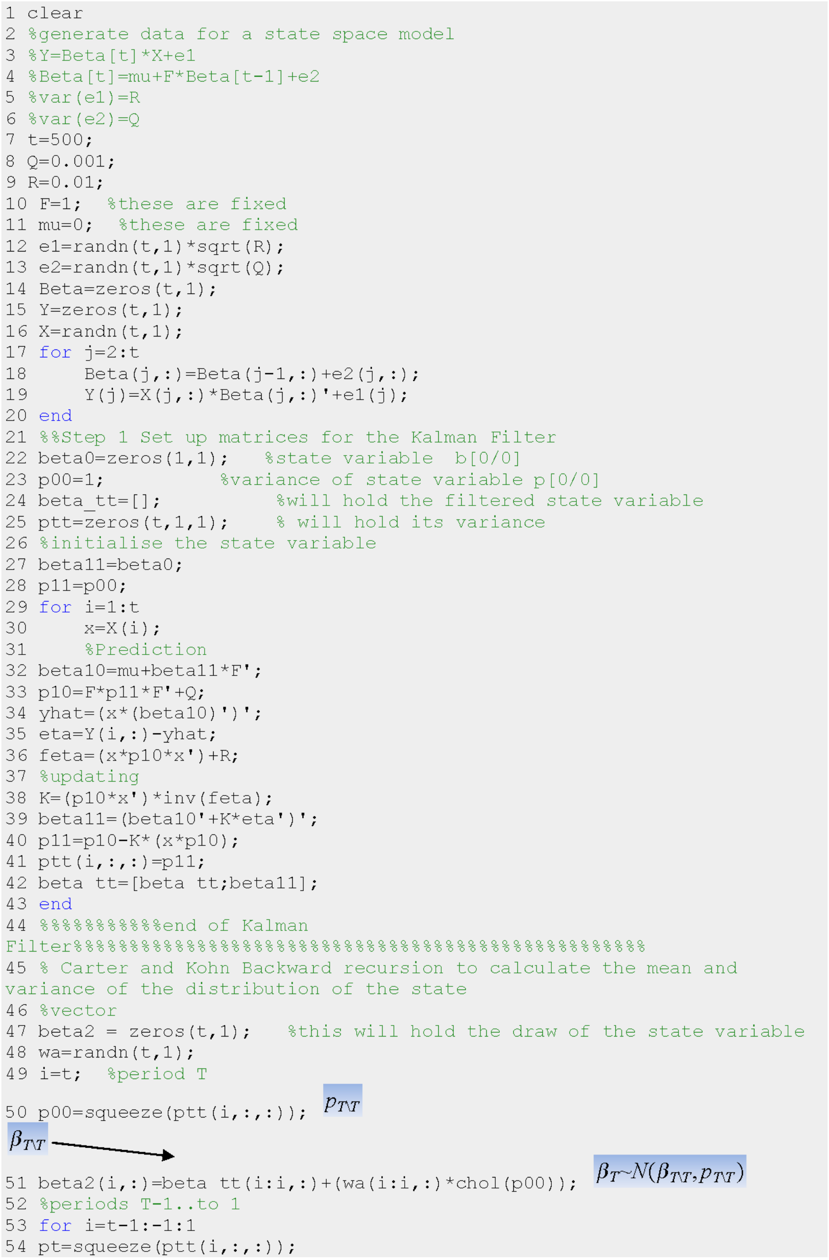
6. The Gibbs sampling algorithm for a VAR with time-varying parameters

We now consider our first example that illustrates the Carter and Kohn algorithm. Following Cogley and Sargent (2002), we consider the following VAR model with time-varying coefficients

$$\begin{aligned} Y_t &= c_t + \sum_{j=1}^{P} B_{j,t} Y_{t-j} + v_t, VAR\left(v_t\right) = R \\ \beta_t &= \{c_t, B_{1,t}....B_{P,t}\} \\ \beta_t &= \mu + F\beta_{t-1} + e_t, VAR(e_t) = Q \end{aligned} \tag{6.1}$$

Note that most empirical applications of this model assume that $\mu = 0$ and $F = 1$ and we are going to implement this assumption in our code below. The Gibbs sampling algorithm for this model can be discerned by noticing that if the time-varying coefficients β_t are known, the conditional posterior distribution of R is inverse Wishart. Similarly, conditional on β_t the distribution of Q is inverse Wishart. Conditional on R and Q and with $\mu = 0$ and $F = 1$ the model in 6.1 is a linear Gaussian state space model. The conditional posterior of β_t is normal and the mean and the variance can be derived via the Carter Kohn algorithm. Therefore the Gibbs sampling algorithm consists of the following steps

Step 1 Set a prior for R and Q and starting values for the Kalman filter. The prior for Q is inverse Wishart $p\left(Q\right) \sim IW(Q_0, T_0)$. Note that this prior is quite crucial as it influences the amount of time-variation allowed for in the VAR model. In other words, a large value for the scale matrix Q_0 would imply more fluctuation in β_t. This prior is typically set using a training sample. The first T_0 observations of the sample are used to estimate a standard fixed coefficient VAR via OLS such that $\beta_0 = \left(X_{0t}'X_{0t}\right)^{-1}\left(X_{0t}'Y_{0t}\right)$ with a coefficient covariance matrix given by $p_{0|0} = \Sigma_0 \otimes \left(X_{0t}'X_{0t}\right)^{-1}$ where $X_{0t} = \{Y_{0t-1}, ...Y_{0t-p}, 1\}$, $\Sigma_0 = \frac{(Y_{0t} - X_{0t}\beta_0)'(Y_{0t} - X_{0t}\beta_0)}{T_0 - K}$ and the subscript 0 denotes the fact that this is the training sample. The scale matrix Q_0 is set equal to $p_{0|0} \times T_0 \times \tau$ where τ is a scaling factor chosen by the researcher. Some studies set $\tau = 3.510^{-4}$ i.e. a small number to reflect the fact that the training sample in typically short and the resulting estimates of $p_{0|0}$ maybe imprecise. Note that one can control the apriori amount of time-variation in the model by varying τ. The prior degrees of freedom are set equal to T_0. The prior for R is inverse Wishart with scale parameter R_0 and degrees of freedom v_{R_0}. The initial state is set equal to $\beta_{0|0} = vec($ $\beta_0)'$ and the intial state covariance is given by $P_{0|0}$. We set a starting value for R and Q.

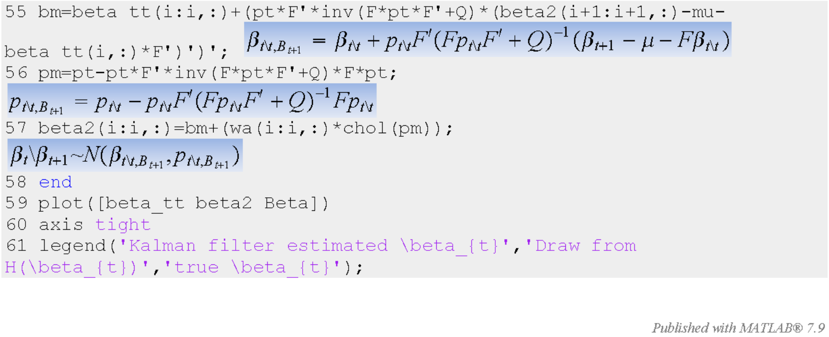


```
clear
%generate data for a state space model
%Y=Beta[t]*X+e1
%Beta[t]=mu+F*Beta[t-1]+e2
%var(e1)=R
%var(e2)=Q
t=500;
Q=0.001;
R=0.01;
F=1;  %these are fixed
mu=0;  %these are fixed
e1=randn(t,1)*sqrt(R);
e2=randn(t,1)*sqrt(Q);
Beta=zeros(t,1);
Y=zeros(t,1);
X=randn(t,1);
for j=2:t
    Beta(j,:)=Beta(j-1,:)+e2(j,:);
    Y(j)=X(j,:)*Beta(j,:)'+e1(j);
end
%%Step 1 Set up matrices for the Kalman Filter
beta0=zeros(1,1);   %state variable  b[0/0]
p00=1;          %variance of state variable p[0/0]
beta_tt=[];          %will hold the filtered state variable
ptt=zeros(t,1,1);    % will hold its variance
%initialise the state variable
beta11=beta0;
p11=p00;
for i=1:t
    x=X(i);
    %Prediction
beta10=mu+beta11*F';
p10=F*p11*F'+Q;
yhat=(x*(beta10)')';
eta=Y(i,:)-yhat;
feta=(x*p10*x')+R;
%updating
K=(p10*x')*inv(feta);
beta11=(beta10'+K*eta')';
p11=p10-K*(x*p10);
ptt(i,:,:)=p11;
beta tt=[beta tt;beta11];
end
%%%%%%%%%%%end of Kalman Filter%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Carter and Kohn Backward recursion to calculate the mean and variance of the distribution of the state
%vector
beta2 = zeros(t,1);   %this will hold the draw of the state variable
wa=randn(t,1);
i=t;  %period T
p00=squeeze(ptt(i,:,:));
beta2(i,:)=beta tt(i:i,:)+(wa(i:i,:)*chol(p00));
%periods T-1..to 1
for i=t-1:-1:1
pt=squeeze(ptt(i,:,:));
```

FIGURE 4. The Carter and Kohn algorithm in Matlab

Step 2 Sample $\tilde{\beta}_t$ conditional on R and Q from its conditional posterior distribution $H\left(\tilde{\beta}_T|R,Q,\tilde{Y}_t\right)$ where $\tilde{\beta}_T = [vec(\beta_1)', vec(\beta_2)', ...vec(\beta_T)']$ and $\tilde{Y}_t = [Y_1, ...Y_T]$. This is done via the Carter and Kohn algorithm as described in the example above. We describe the Matlab implementation in the next section.

Step 3 Sample Q from its conditional posterior distribution. Conditional on $\tilde{\beta}_t$ the posterior of Q is inverse Wishart with scale matrix $\left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right)'\left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right) + Q_0$ and degrees of freedom $T + T_0$ where T denotes the length of the estimation sample and $\tilde{\beta}_t^1$ is the previous draw of the state variable $\tilde{\beta}_t$. Notice that once the state variable is drawn from its distribution we treat it like data. It is therefore easy to extend this step to sample



```
bm=beta tt(i:i,:)+(pt*F'*inv(F*pt*F'+Q)*(beta2(i+1:i+1,:)-mu-
beta tt(i,:)*F')')';
```

$$\beta_{t\backslash t,B_{t+1}} = \beta_{t\backslash t} + p_{t\backslash t}F'(Fp_{t\backslash t}F' + Q)^{-1}(\beta_{t+1} - \mu - F\beta_{t\backslash t})$$

```
pm=pt-pt*F'*inv(F*pt*F'+Q)*F*pt;
```

$$p_{t\backslash t,B_{t+1}} = p_{t\backslash t} - p_{t\backslash t}F'(Fp_{t\backslash t}F' + Q)^{-1}Fp_{t\backslash t}$$

```
beta2(i:i,:)=bm+(wa(i:i,:)*chol(pm));
```

$$\beta_t\backslash\beta_{t+1}\sim N(\beta_{t\backslash t,B_{t+1}}, p_{t\backslash t,B_{t+1}})$$

```
end
plot([beta_tt beta2 Beta])
axis tight
legend('Kalman filter estimated \beta_{t}','Draw from
H(\beta_{t})','true \beta_{t}');
```

Published with MATLAB® 7.9

FIGURE 5. The Carter and Kohn algorithm in Matlab (continued)

μ, F which are just the intercept and AR coefficients in an AR regression for each individual coefficient in $\tilde{\beta}_t$ (the conditional distributions for linear regression models are described in chapter 1)

Step 4. Sample R from its conditional posterior distribution. Conditional on the draw $\tilde{\beta}_t^1$ the posterior of R is inverse Wishart with scale matrix $\left(Y_t - \left(c_t^1 + \sum_{j=1}^P B_{j,t}^1 Y_{t-j}\right)\right)' \left(Y_t - \left(c_t^1 + \sum_{j=1}^P B_{j,t}^1 Y_{t-j}\right)\right) + R_0$ and degrees of freedom $T + v_{R_0}$.

Step 5. Repeat steps 2 to 4 M times and use the last L draws for inference. Note that unlike fixed coefficient VAR models, this state space model requires a large number of draws for convergence (e.g. $M \geq 100,000$).

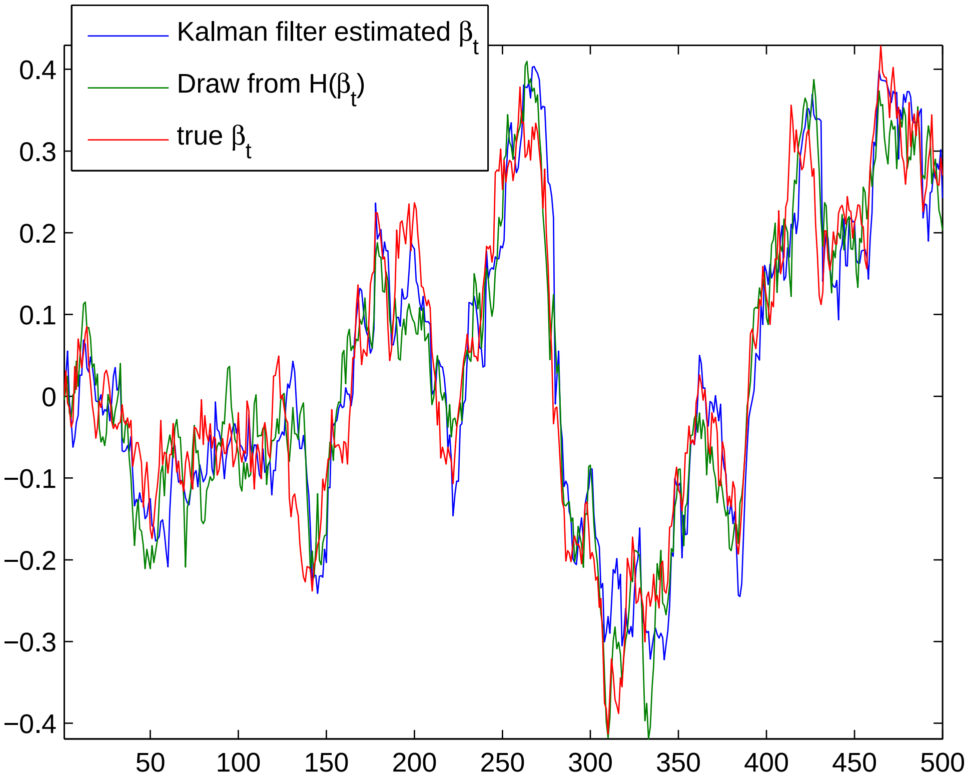
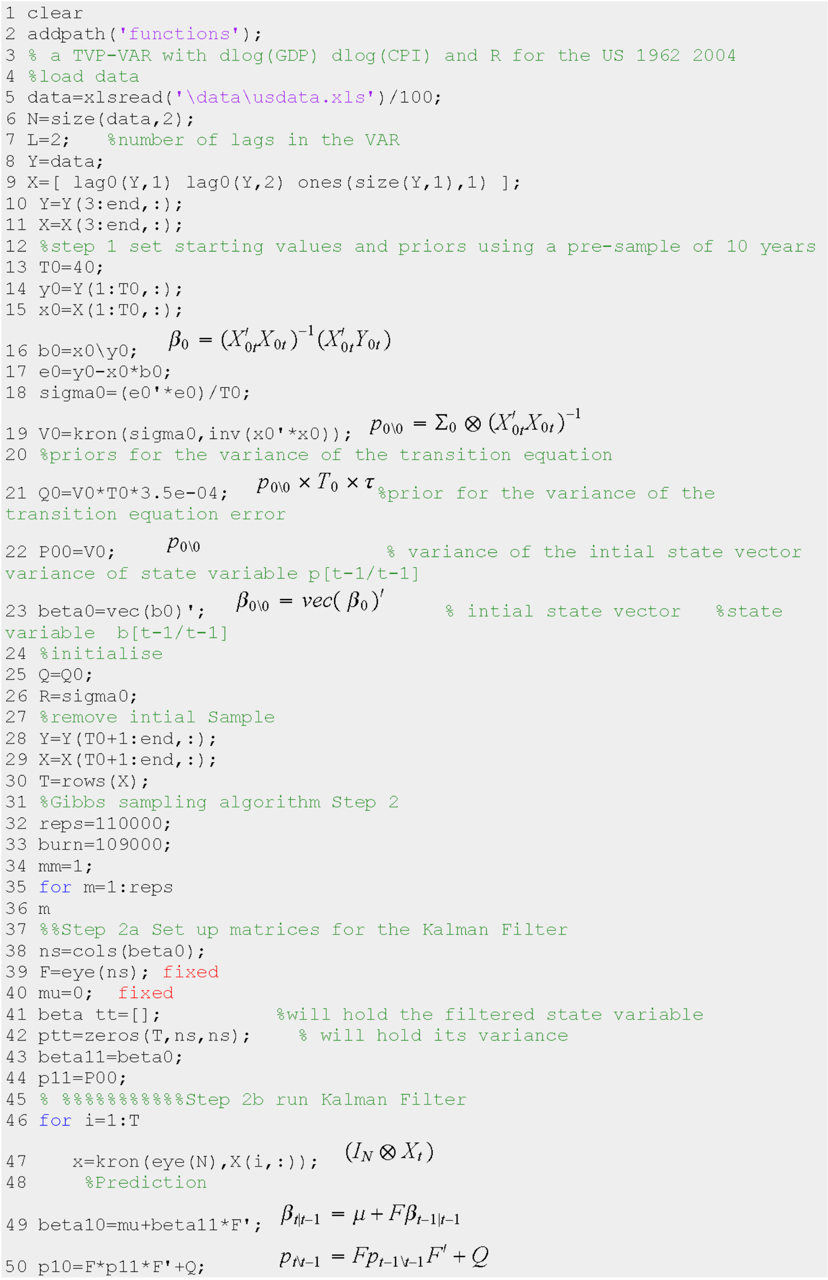


FIGURE 6. A draw from the conditional posterior distribution of β_t using the Carter and Kohn algorithm

6.1. Matlab code for the time-varying parameter VAR. We consider a time-varying VAR model with two lags using US data on GDP growth, CPI inflation and the Federal Funds rate over the period 1954Q3 to 2010Q2 (Example3.m). We use the time-varying VAR model to compute the impulse response to a monetary policy shock at each point in time and see if the response of the US economy to this shock has changed over this period. The code for this model can be seen in figures 7, 8, 9 and 10. Line 13 of the code sets the training sample equal to the first 40 observations of the sample and line 16 calculates a fixed coefficient VAR using this training sample to obtain β_0 and $p_{0|0}$. In calculating Q_0 on line 21 we set $\tau = 3.510^{-4}$. Lines 25 and 26 set a starting value for R and Q. Lines 29 and 30 remove the training sample from the data–the model is estimated on the remaining sample. Lines 38 to 88 sample the time-varying coefficients conditional on R and Q using the Carter and Kohn algorithm. The code for this is exactly the same as in the previous example with some monor differences. First, note that the VAR is expressed as $Y_t = (I_N \otimes X_t)\, vec(\beta_t) + v_t$ for each time period $t = 1....T$. This is convenient as it allows us to write the transition equation in terms of $vec(\beta_t)$ i.e. the VAR coefficients in vectorised form at each point in time. Therefore, on line 47 x is set equal to $(I_N \otimes X_t)$. The second practical differences arises in the backward recursion on lines 64 to 87. In particular (following earlier papers) we draw $\tilde{\beta}_T = [vec(\beta_1)', vec(\beta_2)', ...vec(\beta_T)']$ for $H\left(\tilde{\beta}_T | R, Q, \tilde{Y}_t\right)$ but ensure that the VAR is stable at each point in time. If the stability condition fails for one time period, the entire matrix $\tilde{\beta}_T = [vec(\beta_1)', vec(\beta_2)', ...vec(\beta_T)']$ is discarded and the algorithm tries again. With the draw of $\tilde{\beta}_T$ in hand line 89 calculates the residuals of the transition equation e_t. Line 90 calculates the scale matrix $\left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right)' \left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right) + Q_0$ and line 91 draws Q from the inverse Wishart distribution. Line 94 draws the VAR error covariance R from the inverse Wishart distribution. Note that we use a flat prior for R in this example. One past the burn-in stage we save the draws for $\tilde{\beta}_T$, R and Q. We use the saved draws to compute the impulse response to a monetary policy shock and use sign restrictions to identify a monetary policy shock (lines 106 to 180). We assume that a monetary policy shock is one that increases interest rates, decreases inflation and output growth. The results for the time-varying impulse response are shown in 11. The 3-D surface charts show the impulse response horizon on the Y-axis and the time-series on the X-axis. These results show little evidence of significant variation in the impulse response functions across time for this dataset.



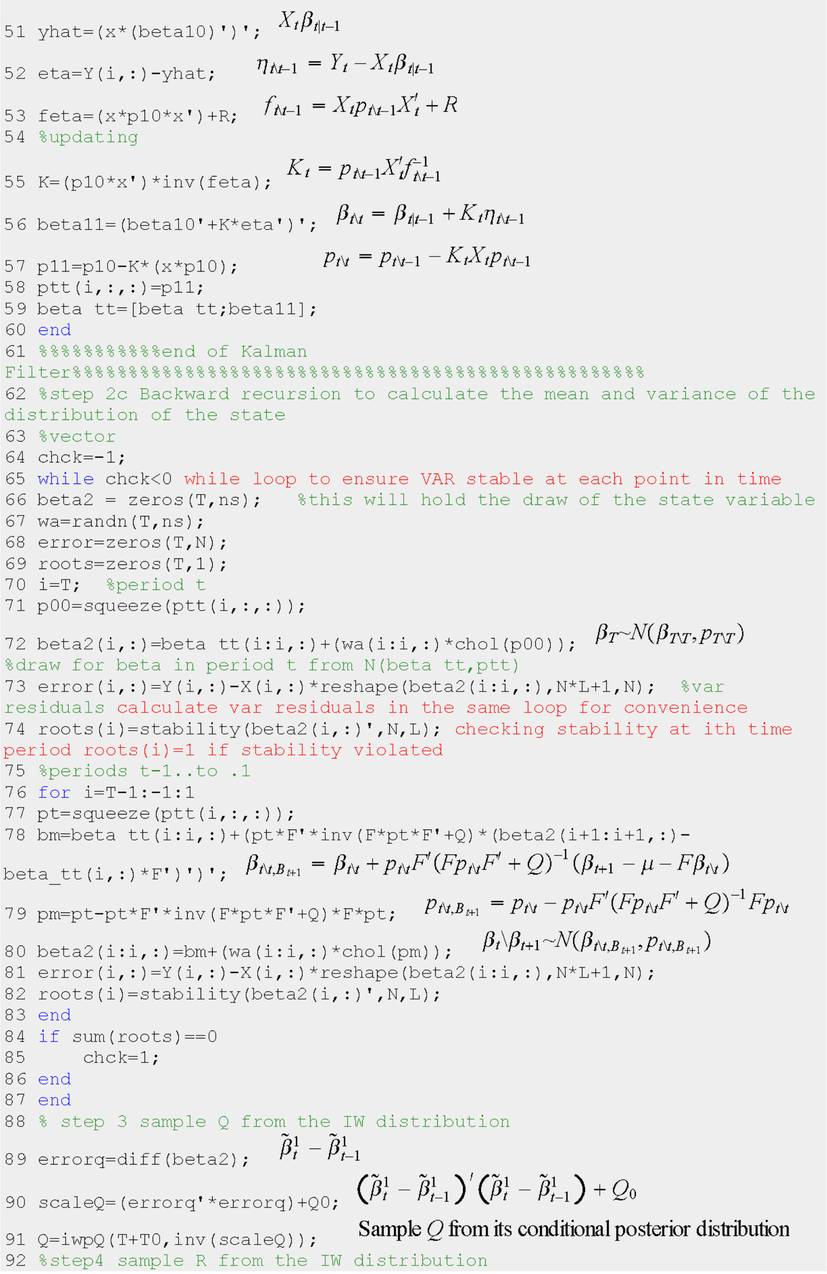
```
clear
addpath('functions');
% a TVP-VAR with dlog(GDP) dlog(CPI) and R for the US 1962 2004
%load data
data=xlsread('\data\usdata.xls')/100;
N=size(data,2);
L=2;   %number of lags in the VAR
Y=data;
X=[ lag0(Y,1) lag0(Y,2) ones(size(Y,1),1) ];
Y=Y(3:end,:);
X=X(3:end,:);
%step 1 set starting values and priors using a pre-sample of 10 years
T0=40;
y0=Y(1:T0,:);
x0=X(1:T0,:);
b0=x0\y0;    $\beta_0 = (X_{0t}'X_{0t})^{-1}(X_{0t}'Y_{0t})$
e0=y0-x0*b0;
sigma0=(e0'*e0)/T0;
V0=kron(sigma0,inv(x0'*x0));  $p_{0\backslash 0} = \Sigma_0 \otimes (X_{0t}'X_{0t})^{-1}$
%priors for the variance of the transition equation
Q0=V0*T0*3.5e-04;  $p_{0\backslash 0} \times T_0 \times \tau$ %prior for the variance of the
transition equation error
P00=V0;     $p_{0\backslash 0}$                 % variance of the intial state vector
variance of state variable p[t-1/t-1]
beta0=vec(b0)';   $\beta_{0\backslash 0} = vec(\beta_0)'$      % intial state vector   %state
variable  b[t-1/t-1]
%initialise
Q=Q0;
R=sigma0;
%remove intial Sample
Y=Y(T0+1:end,:);
X=X(T0+1:end,:);
T=rows(X);
%Gibbs sampling algorithm Step 2
reps=110000;
burn=109000;
mm=1;
for m=1:reps
m
%%Step 2a Set up matrices for the Kalman Filter
ns=cols(beta0);
F=eye(ns); fixed
mu=0;  fixed
beta tt=[];          %will hold the filtered state variable
ptt=zeros(T,ns,ns);     % will hold its variance
beta11=beta0;
p11=P00;
% %%%%%%%%%%%Step 2b run Kalman Filter
for i=1:T
   x=kron(eye(N),X(i,:));  $(I_N \otimes X_t)$
    %Prediction
beta10=mu+beta11*F';  $\beta_{t|t-1} = \mu + F\beta_{t-1|t-1}$
p10=F*p11*F'+Q;       $p_{t|t-1} = Fp_{t-1\backslash t-1}F' + Q$
```

FIGURE 7. Matlab code for a time-varying VAR

7. The Gibbs sampling algorithm for a Factor Augmented VAR

Our second example is based on the Factor augmented VAR model introduced in Bernanke *et al.* (2005). The FAVAR model can be written compactly as

$$\begin{aligned} X_{i,t} &= b_i F_t + \gamma_i FFR_t + v_{i,t} \qquad (7.1) \\ Z_t &= c_t + \sum_{j=1}^{P} B_j Z_{t-j} + e_t \\ Z_t &= \{F_t, FFR_t\} \\ VAR(v_{i,t}) &= R, VAR(e_t) = Q \end{aligned}$$



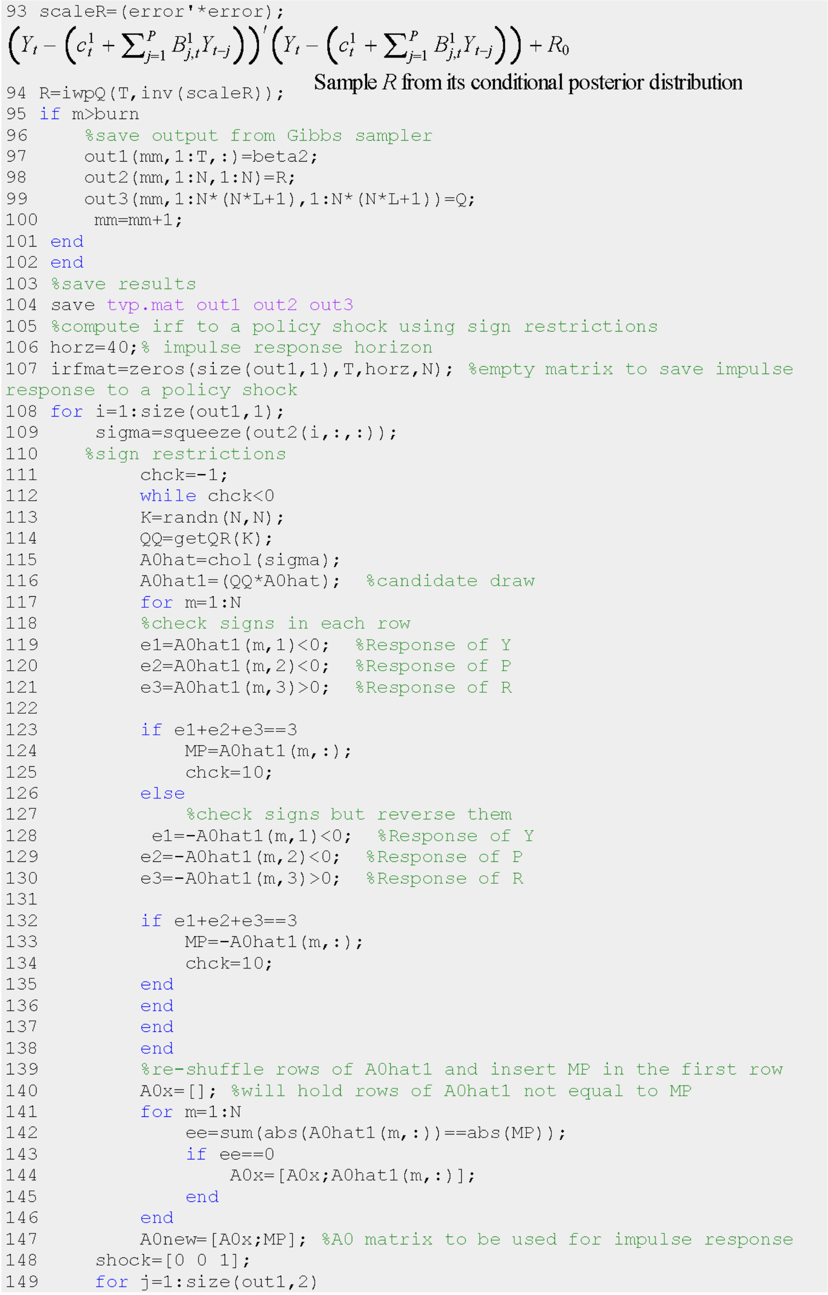
```
yhat=(x*(beta10)')';
eta=Y(i,:)-yhat;
feta=(x*p10*x')+R;
%updating
K=(p10*x')*inv(feta);
beta11=(beta10'+K*eta')';
p11=p10-K*(x*p10);
ptt(i,:,:)=p11;
beta tt=[beta tt;beta11];
end
%%%%%%%%%%%end of Kalman Filter%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%step 2c Backward recursion to calculate the mean and variance of the distribution of the state
%vector
chck=-1;
while chck<0 while loop to ensure VAR stable at each point in time
beta2 = zeros(T,ns);   %this will hold the draw of the state variable
wa=randn(T,ns);
error=zeros(T,N);
roots=zeros(T,1);
i=T;  %period t
p00=squeeze(ptt(i,:,:));
beta2(i,:)=beta tt(i:i,:)+(wa(i:i,:)*chol(p00)); %draw for beta in period t from N(beta tt,ptt)
error(i,:)=Y(i,:)-X(i,:)*reshape(beta2(i:i,:),N*L+1,N);  %var residuals calculate var residuals in the same loop for convenience
roots(i)=stability(beta2(i,:)',N,L); checking stability at ith time period roots(i)=1 if stability violated
%periods t-1..to .1
for i=T-1:-1:1
pt=squeeze(ptt(i,:,:));
bm=beta tt(i:i,:)+(pt*F'*inv(F*pt*F'+Q)*(beta2(i+1:i+1,:)-beta_tt(i,:)*F')')';
pm=pt-pt*F'*inv(F*pt*F'+Q)*F*pt;
beta2(i:i,:)=bm+(wa(i:i,:)*chol(pm));
error(i,:)=Y(i,:)-X(i,:)*reshape(beta2(i:i,:),N*L+1,N);
roots(i)=stability(beta2(i,:)',N,L);
end
if sum(roots)==0
    chck=1;
end
end
% step 3 sample Q from the IW distribution
errorq=diff(beta2);
scaleQ=(errorq'*errorq)+Q0;
Q=iwpQ(T+T0,inv(scaleQ));
%step4 sample R from the IW distribution
```

$X_t\beta_{t|t-1}$
$\eta_{t|t-1} = Y_t - X_t\beta_{t|t-1}$
$f_{t|t-1} = X_t p_{t|t-1} X_t' + R$
$K_t = p_{t|t-1} X_t' f_{t|t-1}^{-1}$
$\beta_{t|t} = \beta_{t|t-1} + K_t \eta_{t|t-1}$
$p_{t|t} = p_{t|t-1} - K_t X_t p_{t|t-1}$
$\beta_T \sim N(\beta_{T|T}, p_{T|T})$
$\beta_{t|t,B_{t+1}} = \beta_{t|t} + p_{t|t}F'(Fp_{t|t}F' + Q)^{-1}(\beta_{t+1} - \mu - F\beta_{t|t})$
$p_{t|t,B_{t+1}} = p_{t|t} - p_{t|t}F'(Fp_{t|t}F' + Q)^{-1}Fp_{t|t}$
$\beta_t \backslash \beta_{t+1} \sim N(\beta_{t|t,B_{t+1}}, p_{t|t,B_{t+1}})$
$\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1$
$\left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right)'\left(\tilde{\beta}_t^1 - \tilde{\beta}_{t-1}^1\right) + Q_0$
Sample Q from its conditional posterior distribution

FIGURE 8. Matlab code for a time-varying VAR (continued)

where $X_{i,t}$ is a $T \times M$ matrix containing a panel of macroeconomic and financial variables. FFR_t denotes the Federal Funds rate and F_t are the unobserved factors which summarise the information in the data $X_{i,t}$. The first equation is the observation equation of the model, while the second equation is the transition equation. Bernanke *et al.* (2005) consider a shock to the interest rate in the transition equation and calculate the impulse response of each variable in $X_{i,t}$.

It is instructive to consider the state-space representation of the FAVAR model in more detail. We assume in this example that the lag length in the transition equation equals 2 and there are 3 unobserved factors $F_t = \{F_{1t}, F_{2t}, F_{3t}\}$.



```
scaleR=(error'*error);
R=iwpQ(T,inv(scaleR));
if m>burn
    %save output from Gibbs sampler
    out1(mm,1:T,:)=beta2;
    out2(mm,1:N,1:N)=R;
    out3(mm,1:N*(N*L+1),1:N*(N*L+1))=Q;
    mm=mm+1;
end
end
%save results
save tvp.mat out1 out2 out3
%compute irf to a policy shock using sign restrictions
horz=40;% impulse response horizon
irfmat=zeros(size(out1,1),T,horz,N); %empty matrix to save impulse response to a policy shock
for i=1:size(out1,1);
    sigma=squeeze(out2(i,:,:));
   %sign restrictions
        chck=-1;
        while chck<0
        K=randn(N,N);
        QQ=getQR(K);
        A0hat=chol(sigma);
        A0hat1=(QQ*A0hat);  %candidate draw
        for m=1:N
        %check signs in each row
        e1=A0hat1(m,1)<0;  %Response of Y
        e2=A0hat1(m,2)<0;  %Response of P
        e3=A0hat1(m,3)>0;  %Response of R

        if e1+e2+e3==3
            MP=A0hat1(m,:);
            chck=10;
        else
            %check signs but reverse them
         e1=-A0hat1(m,1)<0;  %Response of Y
        e2=-A0hat1(m,2)<0;  %Response of P
        e3=-A0hat1(m,3)>0;  %Response of R

        if e1+e2+e3==3
            MP=-A0hat1(m,:);
            chck=10;
        end
        end
        end
        end
        %re-shuffle rows of A0hat1 and insert MP in the first row
        A0x=[]; %will hold rows of A0hat1 not equal to MP
        for m=1:N
            ee=sum(abs(A0hat1(m,:))==abs(MP));
            if ee==0
                A0x=[A0x;A0hat1(m,:)];
            end
        end
        A0new=[A0x;MP]; %A0 matrix to be used for impulse response
    shock=[0 0 1];
    for j=1:size(out1,2)
```

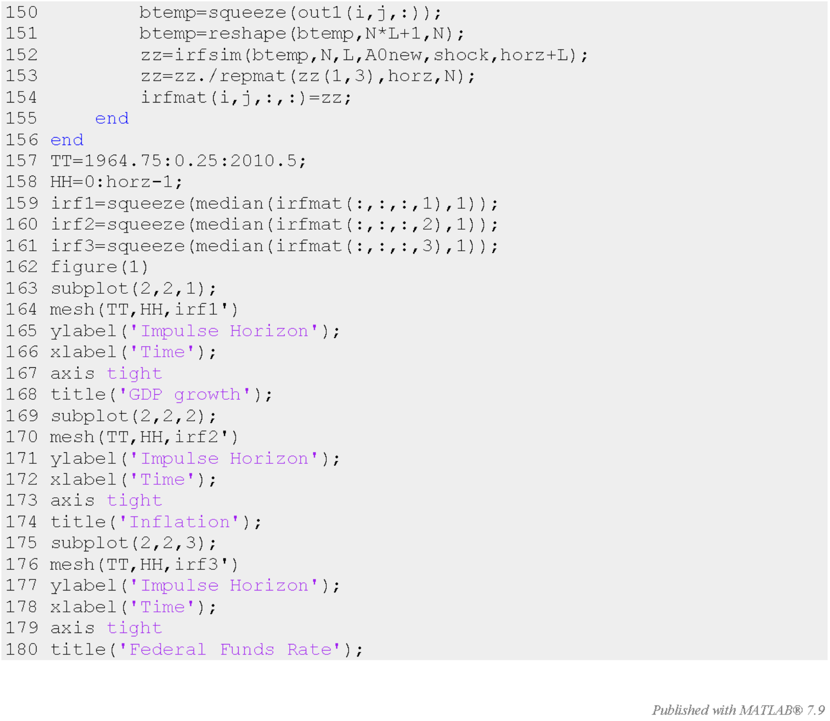
$$\left(Y_t - \left(c_t^1 + \sum_{j=1}^{P} B_{j,t}^1 Y_{t-j}\right)\right)'\left(Y_t - \left(c_t^1 + \sum_{j=1}^{P} B_{j,t}^1 Y_{t-j}\right)\right) + R_0$$

Sample R from its conditional posterior distribution

FIGURE 9. Matlab code for a time-varying VAR (continued)

Consider the observation equation of the model

$$\underbrace{\begin{pmatrix} X_{1t} \\ X_{2t} \\ X_{3t} \\ . \\ . \\ . \\ . \\ X_{Mt} \\ FFR_t \end{pmatrix}}_{\tilde{X}_{it}} = \underbrace{\begin{pmatrix} b_{11} & . & b_{13} & \gamma_1 & 0 & 0 & 0 & 0 \\ b_{21} & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ b_{M1} & & b_{M3} & \gamma_M & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} F_{1t} \\ F_{2t} \\ F_{3t} \\ FFR_t \\ F_{1t-1} \\ F_{2t-1} \\ F_{3t-1} \\ FFR_{t-1} \end{pmatrix}}_{\beta_t} + \underbrace{\begin{pmatrix} v_{1t} \\ v_{2t} \\ v_{3t} \\ . \\ . \\ . \\ . \\ v_{Nt} \\ 0 \end{pmatrix}}_{v_t} \tag{7.2}$$



```
        btemp=squeeze(out1(i,j,:));
        btemp=reshape(btemp,N*L+1,N);
        zz=irfsim(btemp,N,L,A0new,shock,horz+L);
        zz=zz./repmat(zz(1,3),horz,N);
        irfmat(i,j,:,:)=zz;
    end
end
TT=1964.75:0.25:2010.5;
HH=0:horz-1;
irf1=squeeze(median(irfmat(:,:,:,1),1));
irf2=squeeze(median(irfmat(:,:,:,2),1));
irf3=squeeze(median(irfmat(:,:,:,3),1));
figure(1)
subplot(2,2,1);
mesh(TT,HH,irf1')
ylabel('Impulse Horizon');
xlabel('Time');
axis tight
title('GDP growth');
subplot(2,2,2);
mesh(TT,HH,irf2')
ylabel('Impulse Horizon');
xlabel('Time');
axis tight
title('Inflation');
subplot(2,2,3);
mesh(TT,HH,irf3')
ylabel('Impulse Horizon');
xlabel('Time');
axis tight
title('Federal Funds Rate');
```

Published with MATLAB® 7.9

Figure 10. Matlab code for a time-varying VAR (continued)

The left hand side of the observation equation 7.2 contains the dataset $X_{i,t}$ with the Funds rate as the last variable (thus $\tilde{X}_{it} = \{X_{i,t}, FFR_t\}$) . $X_{i,t}$ is related to the three factors via the factor loadings b_{ij} where $i = 1...M$ and $j = 1, 2, 3$. $X_{i,t}$ is related to the Federal Funds rate via the coefficients γ_i. Bernanke *et al.* (2005) assume that γ_i are non-zero only for fast moving financial variables. FFR_t appears in the state vector β_t (even though it is observed) as we want it to be part of the transition equation. Therefore the last row of the coefficient matrix H describes the identity $FFR_t = FFR_t$. The state vector contains the first lag of all state variables as we want two lags in the VAR

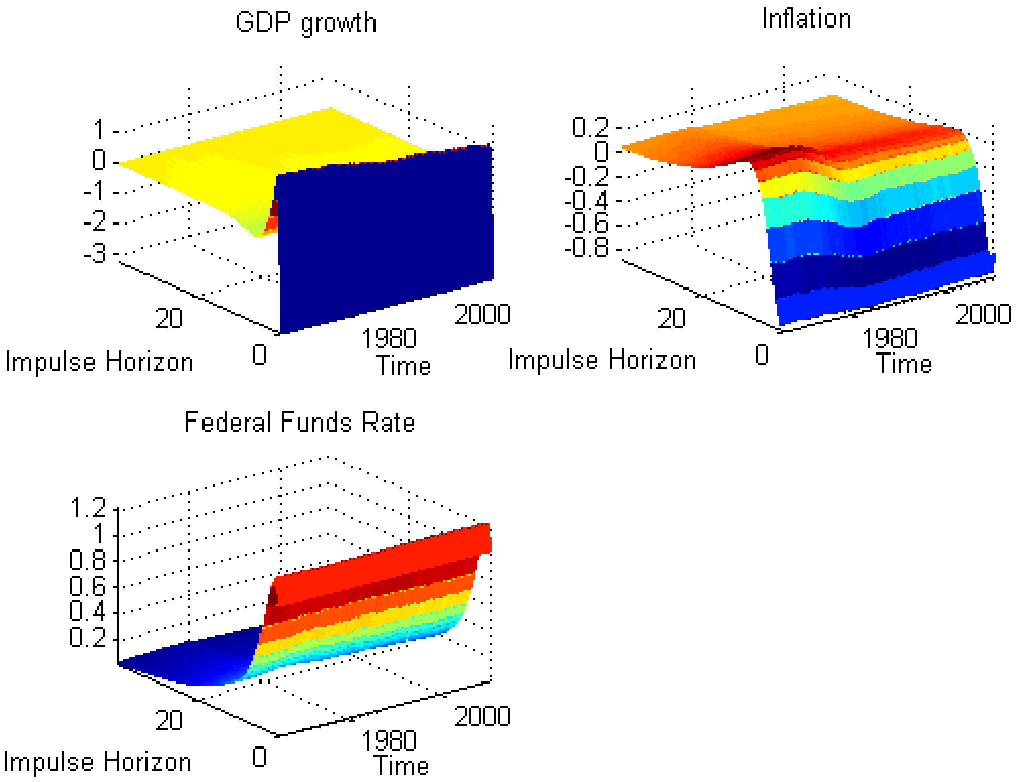


Figure 11. Time-varying impulse responses to a monetary policy shock

that forms the transition equation. Note also that

$$VAR\left(v_t\right) = R = \begin{pmatrix} R_1 & 0 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 & 0 \\ 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & R_M & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.3}$$

The transition equation of the model is

$$\underbrace{\begin{pmatrix} F_{1t} \\ F_{2t} \\ F_{3t} \\ FFR_t \\ F_{1t-1} \\ F_{2t-1} \\ F_{3t-1} \\ FFR_{t-1} \end{pmatrix}}_{\beta_t} = \underbrace{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\mu} + \underbrace{\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} & A_{27} & A_{28} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} & A_{37} & A_{38} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} & A_{47} & A_{48} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{F} \underbrace{\begin{pmatrix} F_{1t-1} \\ F_{2t-1} \\ F_{3t-1} \\ FFR_{t-1} \\ F_{1t-2} \\ F_{2t-2} \\ F_{3t-2} \\ FFR_{t-2} \end{pmatrix}}_{\beta_{t-1}} + \underbrace{\begin{pmatrix} e_{1t} \\ e_{2t} \\ e_{3t} \\ e_{4t} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{e_t} \tag{7.4}$$

Note that this is simply a VAR(2) in F_{1t}, F_{2t}, F_{3t} and FFR_t written in first order companion form to make consistent with the usual form of a transition equation (i.e. the transition equation needs to be in AR(1) form). Note that:

$$VAR(e_t) = Q = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} & 0 & 0 & 0 & 0 \\ Q_{12} & Q_{22} & Q_{23} & Q_{24} & 0 & 0 & 0 & 0 \\ Q_{13} & Q_{23} & Q_{33} & Q_{34} & 0 & 0 & 0 & 0 \\ Q_{14} & Q_{24} & Q_{23} & Q_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{7.5}$$

where the zeros result from the fact that the last 4 equations in the transition equation describe identities. Therefore the matrix Q is singular in this FAVAR model. This implies that the Carter and Kohn recursion has to be generalised slightly to take this singularity into account as discussed above. This modification implies that we use $\mu^*, F^*, Q^*, \beta^*_{t+1}$

in equations 3.13 and 3.14 where $\mu^*, F^*, Q^*, \beta^*_{t+1}$ denote the first jv rows of μ, F, Q, β_{t+1}. In our example $jv = 4$ as the top 4×4 block of Q is non-singular and we draw three factors (the FFR_t equation in the observation equation is an identity).

The Gibbs sampling algorithm can be discerned by imagining the situation where the factors F_t are observed. Give the factors, the observation equation is just M linear regressions of the form $X_{i,t} = b_{ij}F_{jt} + \gamma_i FFR_t + v_{i,t}$ and the conditional distributions studies in Chapter 1 apply immediately to sample b_{ij} and γ_i (i.e. the elements of H) and R. Similarly, given the factors, the transition equation is simply a VAR model. The conditional distributions in Chapter 2 can be used to sample u, F and Q. Finally, given a draw for H, R, u, F and Q the model can be cast into the state-space form shown in equations 7.2 and 7.4. Then the Carter and Kohn algorithm can be used to draw F_t from its conditional distribution. The Gibbs sampling algorithm consists of the following steps

Step 1 Set priors and starting values. The prior for the factor loadings is normal. Let $H_i = \{b_{ij}, \gamma_i\}$. Then $p(H_i) \sim N(H_{i0}, \Sigma_{H_i})$. The prior for the diagonal elements of R is inverse Gamma and given by $p(R_{ii}) \sim IG(R_{ii0}, V_{R0})$. The prior for the VAR parameters μ, F and Q can be set using any of the priors for VARs considered in the previous chapter. For example, one may consider setting an independent Normal inverse Wishart prior. Collecting the VAR coefficients in the matrix B and the non-zero elements of Q in the matrix Ω this prior can be represented as $p(B) \sim N(B_0, \Sigma_B)$ and $p(\Omega) \sim IW(\Omega_0, V_0)$. The Kalman filter requires the initial value of the state vector $\beta_t = \begin{pmatrix} F_{1t} \\ F_{2t} \\ F_{3t} \\ FFR_t \\ F_{1t-1} \\ F_{2t-1} \\ F_{3t-1} \\ FFR_{t-1} \end{pmatrix}$. One can use principal components to get an initial estimate of F_{1t}, F_{2t} and F_{3t} to set $\beta_{0|0}$. The principal component estimate also provides a good starting value for the factors $F_t = F_{1t}, F_{2t}$ and F_{3t}. One can arbitrarily set $R_{ii} = 1$ and Ω to an identity matrix to start the algorithm.

Step 2. Conditional on the factors F_t and R_{ii} sample the factor loadings $H_i = \{b_{ij}, \gamma_i\}$ from their conditional distributions. For each variable in X_{it} the factor loadings have a normal conditional posterior (as described in Chapter 1) $H(H_i|F_t, R_{ii}) \sim N(H_i^*, V_i^*)$

$$H_i^* = \left(\Sigma_{H_i}^{-1} + \frac{1}{R_{ii}} Z_t' Z_t\right)^{-1} \left(\Sigma_{H_i}^{-1} H_{i0} + \frac{1}{R_{ii}} Z_t' X_{it}\right)$$

$$V_i^* = \left(\Sigma_{H_i}^{-1} + \frac{1}{R_{ii}} Z_t' Z_t\right)^{-1}$$

where $Z_t = \{F_{1t}, F_{2t}, F_{3t}, FFR_t\}$ if the ith series X_{it} is a fast moving data series which has a contemporaneous relationship with the Federal Funds rate (e.g. stock prices) and $Z_t = \{F_{1t}, F_{2t}, F_{3t}\}$ if the ith series X_{it} is a slow moving data series which has no contemporaneous relationship with the Federal Funds rate (e.g. GDP). Note that as F_{1t}, F_{2t}, F_{3t} and H_i are both estimated the model is unidentified. Bernanke *et al.* (2005) suggest fixing the top 3×3 block of b_{ij} to an identity matrix and the top 3×1 block of γ_i to zero for identification. See Bernanke *et al.* (2005) for more details on this issue.

Step 3. Conditional on the factors F_t and the factor loadings $H_i = \{b_{ij}, \gamma_i\}$ sample the variance of the error terms of the observation equation R_{ii} from the inverse Gamma distribution with scale parameter $(X_{it} - Z_t H_i)'(X_{it} - Z_t H_i) + R_{ii0}$ with degrees of freedom $T + V_{R0}$ where T is the length of the estimation sample.

Step 4. Conditional on the factors F_t and the error covariance matrix Ω, the posterior for the VAR coefficients B (recall $B = \{\mu, F\}$ the coefficients in the transition equation of the model) is normal (see Chapter 2) and given as $H(B|F_t, \Omega) \sim N(B^*, D^*)$ where B_0, Σ_B

$$\begin{aligned} B^* &= \left(\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t' \bar{X}_t\right)^{-1} \left(\Sigma_B^{-1} vec(B_0) + \Omega^{-1} \otimes \bar{X}_t' \bar{X}_t vec(\hat{B})\right) \\ D^* &= \left(\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t' \bar{X}_t\right)^{-1} \end{aligned}$$

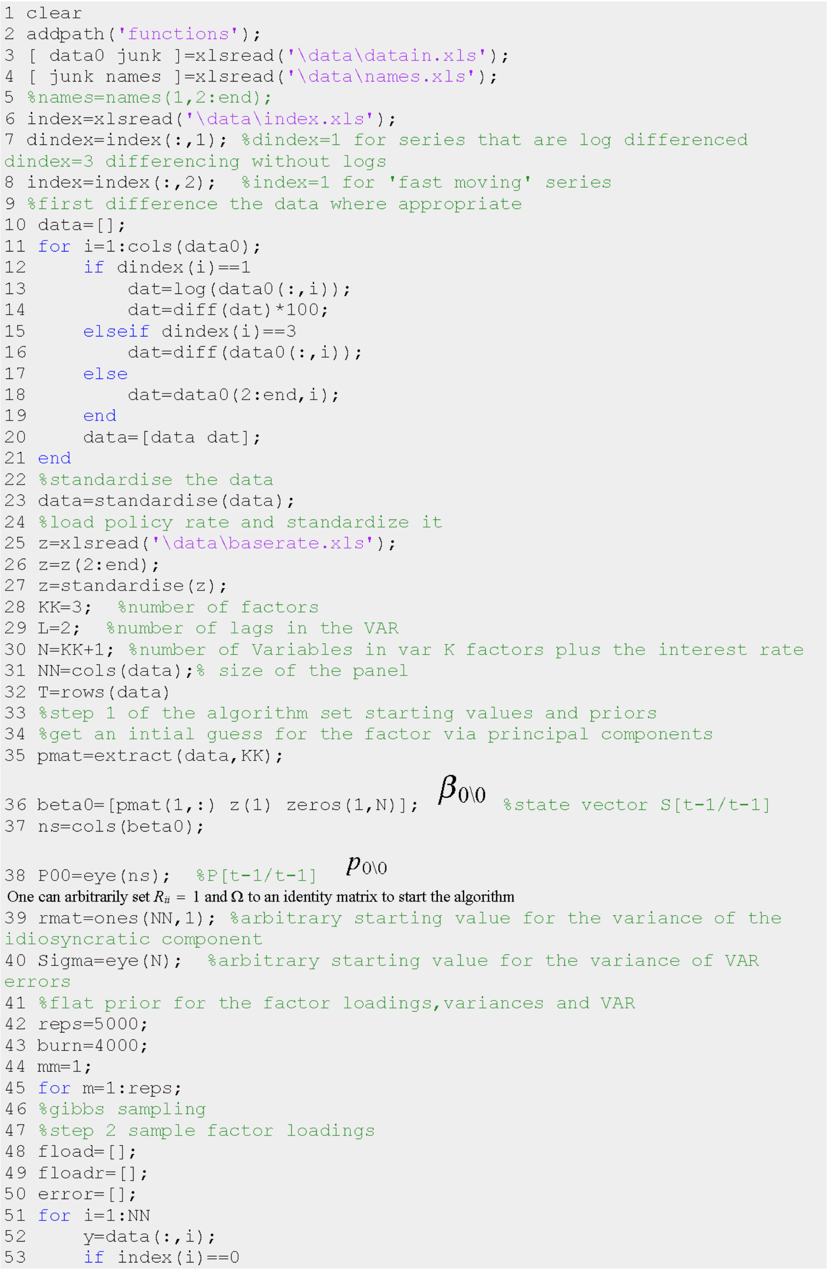
where $\bar{X}_t = \{F_{t-1}, FFR_{t-1}, F_{t-2}, FFR_{t-2}, 1\}$ and $\hat{B}$ is the OLS estimate of B.

Step 5. Conditional on the factors F_t and the VAR coefficients B the error covariance Ω has a inverse Wishart posterior with scale matrix $\left(Y_t - \bar{X}_t B\right)' \left(Y_t - \bar{X}_t B\right) + \Omega_0$ and degrees of freedom $T{+}V_0$.

Step 6. Given H_i, R, B and Ω the model can be cast into state-space form and then the factors F_t are sampled via the Carter and Kohn algorithm.

Step 7 Repeat steps 2 to 6 M times and use the last L values for inference

7.1. Matlab code for the FAVAR model. We estimate a FAVAR model using UK data over the period 1970Q1 to 2006Q1. We use 40 Macroeconomic and financial time series along with the Bank of England policy rate to estimate the model and consider the impact of a monetary policy shock.



```
1 clear
2 addpath('functions');
3 [ data0 junk ]=xlsread('\data\datain.xls');
4 [ junk names ]=xlsread('\data\names.xls');
5 %names=names(1,2:end);
6 index=xlsread('\data\index.xls');
7 dindex=index(:,1); %dindex=1 for series that are log differenced
dindex=3 differencing without logs
8 index=index(:,2);  %index=1 for 'fast moving' series
9 %first difference the data where appropriate
10 data=[];
11 for i=1:cols(data0);
12     if dindex(i)==1
13         dat=log(data0(:,i));
14         dat=diff(dat)*100;
15     elseif dindex(i)==3
16         dat=diff(data0(:,i));
17     else
18         dat=data0(2:end,i);
19     end
20     data=[data dat];
21 end
22 %standardise the data
23 data=standardise(data);
24 %load policy rate and standardize it
25 z=xlsread('\data\baserate.xls');
26 z=z(2:end);
27 z=standardise(z);
28 KK=3;  %number of factors
29 L=2;  %number of lags in the VAR
30 N=KK+1; %number of Variables in var K factors plus the interest rate
31 NN=cols(data);% size of the panel
32 T=rows(data)
33 %step 1 of the algorithm set starting values and priors
34 %get an intial guess for the factor via principal components
35 pmat=extract(data,KK);
```

$\beta_{0\backslash 0}$

```
36 beta0=[pmat(1,:) z(1) zeros(1,N)];  %state vector S[t-1/t-1]
37 ns=cols(beta0);
```

$p_{0\backslash 0}$

```
38 P00=eye(ns);  %P[t-1/t-1]
```

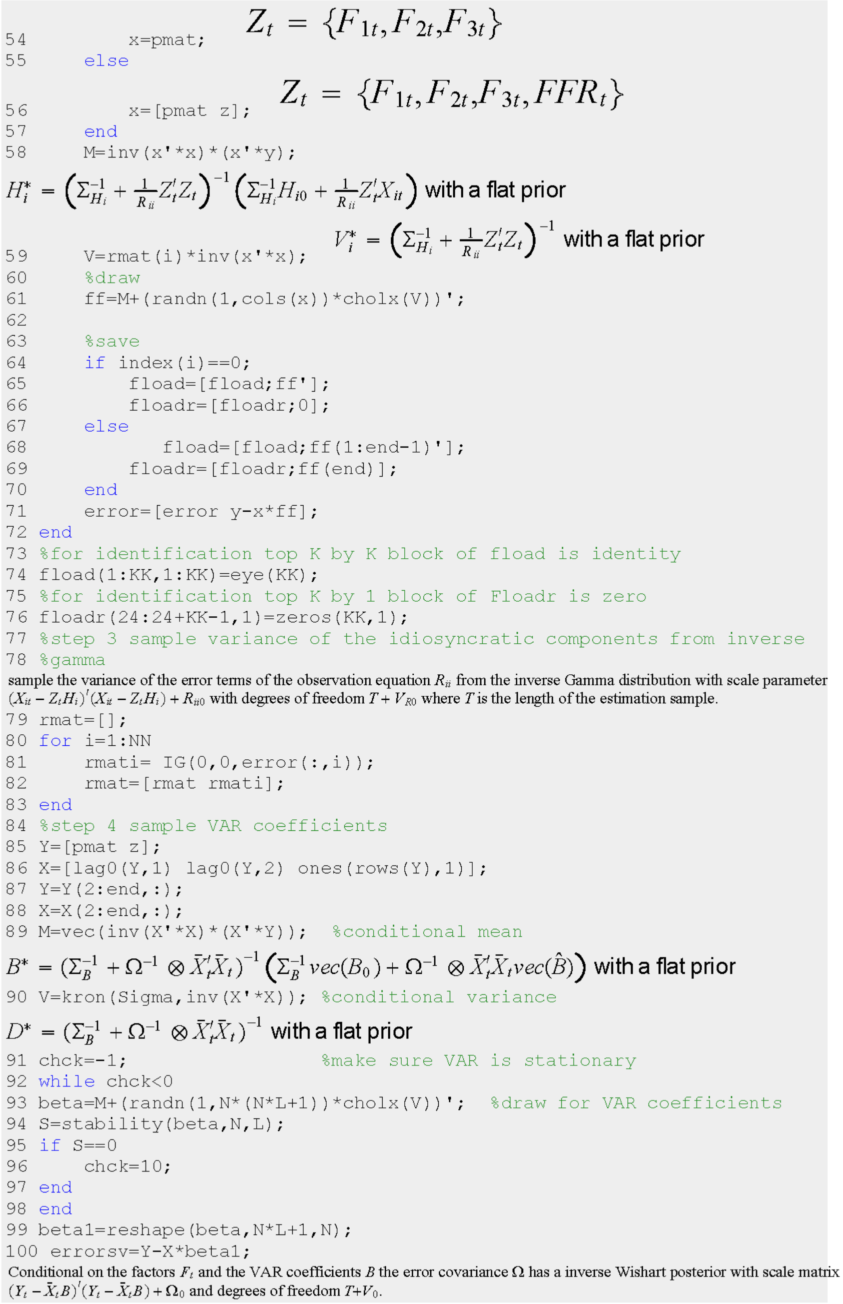
One can arbitrarily set R_{ii} = 1 and Ω to an identity matrix to start the algorithm

```
39 rmat=ones(NN,1); %arbitrary starting value for the variance of the
idiosyncratic component
40 Sigma=eye(N);  %arbitrary starting value for the variance of VAR
errors
41 %flat prior for the factor loadings,variances and VAR
42 reps=5000;
43 burn=4000;
44 mm=1;
45 for m=1:reps;
46 %gibbs sampling
47 %step 2 sample factor loadings
48 fload=[];
49 floadr=[];
50 error=[];
51 for i=1:NN
52     y=data(:,i);
53     if index(i)==0
```

FIGURE 12. Code for the FAVAR model

The Matlab code for this example (example4.m) can be seen in figures 12, 13, 14, 15, 16 and 17.

Lines 3 and 4 load the $T \times M$ panel of UK data and the variable names ($M = 40$). Line 6 reads a variable called index. The first column is a $M \times 1$ vector which equals 1 if the corresponding data series in the panel has to be first differenced. The second column is a $M \times 1$ vector which equals 1 if the corresponding data series is a fast-moving variable (like an asset price) and will have a contemporaneous relationship with the policy interest rate i.e. $\gamma_i \neq 0$ for this variable. Lines 10 to 23 transform the data to stationarity and standardises it. Lines 25 to 27 read the bank rate and standardises it. Line 35 extracts three principal components from the dataset to use as starting values for the three factors in this example. Line 36 defines $\beta_{0|0}$ =[pmat(1,:) z(1) zeros(1,N)]. Notice that there are 8 state



$$Z_t = \{F_{1t}, F_{2t}, F_{3t}\}$$

```
54         x=pmat;
55     else
```

$$Z_t = \{F_{1t}, F_{2t}, F_{3t}, FFR_t\}$$

```
56         x=[pmat z];
57     end
58     M=inv(x'*x)*(x'*y);
```

$$H_i^* = \left(\Sigma_{H_i}^{-1} + \tfrac{1}{R_{ii}} Z_t' Z_t\right)^{-1} \left(\Sigma_{H_i}^{-1} H_{i0} + \tfrac{1}{R_{ii}} Z_t' X_{it}\right) \text{ with a flat prior}$$

$$V_i^* = \left(\Sigma_{H_i}^{-1} + \tfrac{1}{R_{ii}} Z_t' Z_t\right)^{-1} \text{ with a flat prior}$$

```
59     V=rmat(i)*inv(x'*x);
60     %draw
61     ff=M+(randn(1,cols(x))*cholx(V))';
62
63     %save
64     if index(i)==0;
65         fload=[fload;ff'];
66         floadr=[floadr;0];
67     else
68            fload=[fload;ff(1:end-1)'];
69         floadr=[floadr;ff(end)];
70     end
71     error=[error y-x*ff];
72 end
73 %for identification top K by K block of fload is identity
74 fload(1:KK,1:KK)=eye(KK);
75 %for identification top K by 1 block of Floadr is zero
76 floadr(24:24+KK-1,1)=zeros(KK,1);
77 %step 3 sample variance of the idiosyncratic components from inverse
78 %gamma
```

sample the variance of the error terms of the observation equation R_{ii} from the inverse Gamma distribution with scale parameter $(X_{it} - Z_t H_i)'(X_{it} - Z_t H_i) + R_{ii0}$ with degrees of freedom $T + V_{R0}$ where T is the length of the estimation sample.

```
79 rmat=[];
80 for i=1:NN
81     rmati= IG(0,0,error(:,i));
82     rmat=[rmat rmati];
83 end
84 %step 4 sample VAR coefficients
85 Y=[pmat z];
86 X=[lag0(Y,1) lag0(Y,2) ones(rows(Y),1)];
87 Y=Y(2:end,:);
88 X=X(2:end,:);
89 M=vec(inv(X'*X)*(X'*Y));  %conditional mean
```

$$B^* = (\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t)^{-1} \left(\Sigma_B^{-1} vec(B_0) + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t vec(\hat{B})\right) \text{ with a flat prior}$$

```
90 V=kron(Sigma,inv(X'*X)); %conditional variance
```

$$D^* = (\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t)^{-1} \text{ with a flat prior}$$

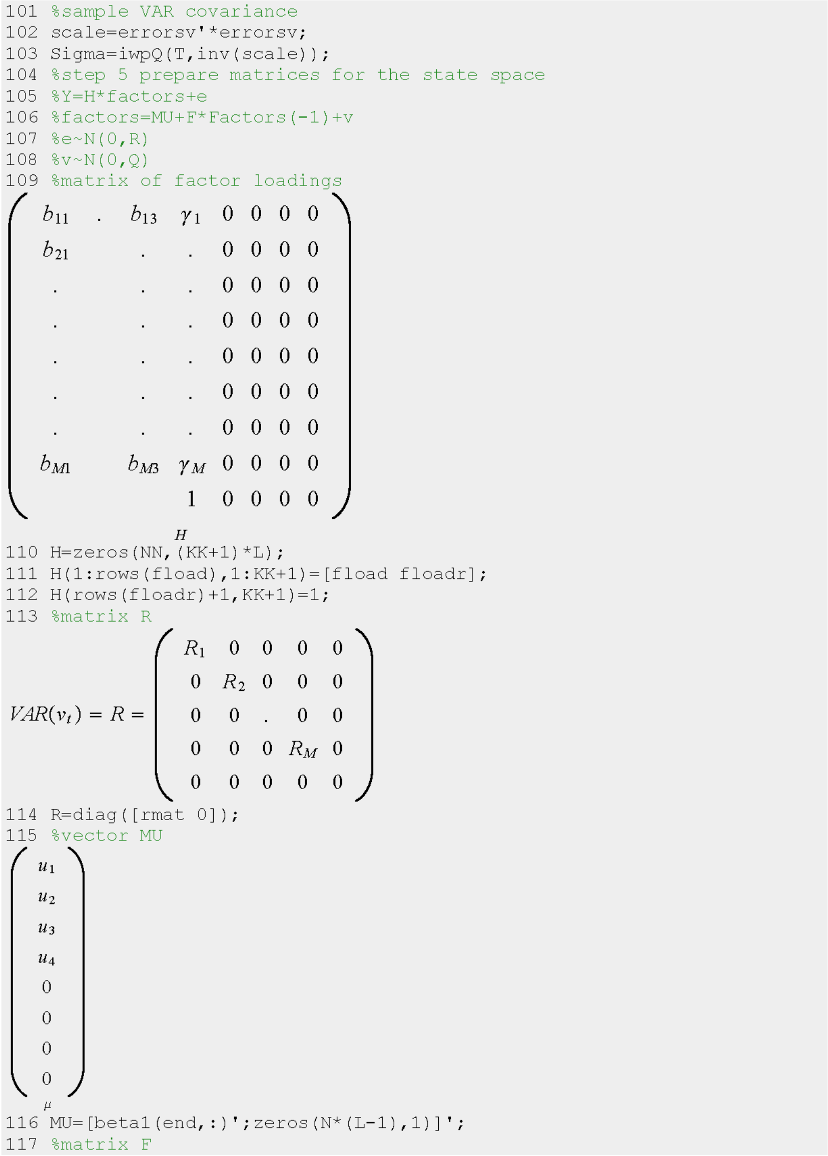
```
91 chck=-1;                    %make sure VAR is stationary
92 while chck<0
93 beta=M+(randn(1,N*(N*L+1))*cholx(V))';  %draw for VAR coefficients
94 S=stability(beta,N,L);
95 if S==0
96     chck=10;
97 end
98 end
99 beta1=reshape(beta,N*L+1,N);
100 errorsv=Y-X*beta1;
```

Conditional on the factors F_t and the VAR coefficients B the error covariance Ω has a inverse Wishart posterior with scale matrix $(Y_t - \bar{X}_t B)'(Y_t - \bar{X}_t B) + \Omega_0$ and degrees of freedom T+V_0.

Figure 13. Code for the FAVAR model (continued)

variables: 3 factors, the interest rate and the first lags of these 4 state variables and thus $\beta_{0|0}$ is 1×8. Line 38 sets $p_{0|0}$ as a 8×8 identity matrix. We arbitrarily set $R_{ii} = 1$ and $\Omega = I$ to start the algorithm on lines 39 and 40. Note that following Bernanke *et al.* (2005) we will not use prior distributions for the regression or VAR coefficients which will imply that the conditional posteriors collapse to OLS formulae.

Lines 48 to 72 sample the factor loadings. The code loops through the 40 data series and selects each as the dependent variable (line 52) to be regressed on the factors only for slow moving series (line 54) or the factors and the policy interest rate for fast moving series (line 56). Line 58 calculates the mean of the conditional posterior



```
101 %sample VAR covariance
102 scale=errorsv'*errorsv;
103 Sigma=iwpQ(T,inv(scale));
104 %step 5 prepare matrices for the state space
105 %Y=H*factors+e
106 %factors=MU+F*Factors(-1)+v
107 %e~N(0,R)
108 %v~N(0,Q)
109 %matrix of factor loadings
```

$$\underset{H}{\begin{pmatrix} b_{11} & . & b_{13} & \gamma_1 & 0 & 0 & 0 & 0 \\ b_{21} & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ . & & . & . & 0 & 0 & 0 & 0 \\ b_{M1} & & b_{M3} & \gamma_M & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}$$

```
110 H=zeros(NN,(KK+1)*L);
111 H(1:rows(fload),1:KK+1)=[fload floadr];
112 H(rows(floadr)+1,KK+1)=1;
113 %matrix R
```

$$VAR(v_t) = R = \begin{pmatrix} R_1 & 0 & 0 & 0 & 0 \\ 0 & R_2 & 0 & 0 & 0 \\ 0 & 0 & . & 0 & 0 \\ 0 & 0 & 0 & R_M & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
114 R=diag([rmat 0]);
115 %vector MU
```

$$\underset{\mu}{\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}$$

```
116 MU=[beta1(end,:)';zeros(N*(L-1),1)]';
117 %matrix F
```

FIGURE 14. Code for the FAVAR model (continued)

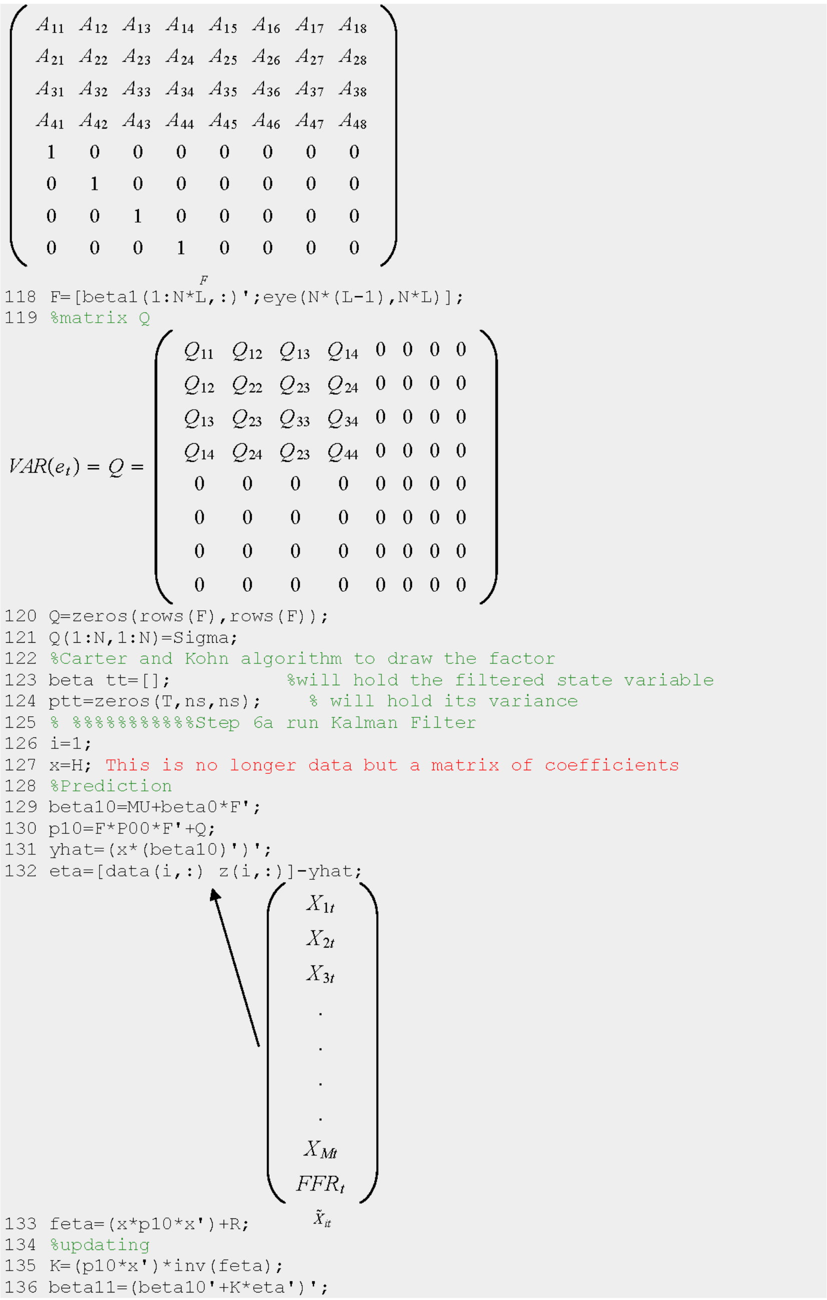
distribution of the factor loadings without the priors

$$H_i^* = (Z_t' Z_t)^{-1} (Z_t' X_{it})$$

and line 59 calculates the variance of this distribution (without the prior information).

$$V_i^* = \left(\frac{1}{R_{ii}} Z_t' Z_t\right)^{-1}$$

The coefficients b_{ij} are stored in the matrix fload and the coefficients γ_i in floadr. Lines 74 and 76 impose the identification conditions and fix the top 3×3 block of fload to an identity matrix and top 3×1 block of floadr to 0.



$$\underbrace{\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} & A_{15} & A_{16} & A_{17} & A_{18} \\ A_{21} & A_{22} & A_{23} & A_{24} & A_{25} & A_{26} & A_{27} & A_{28} \\ A_{31} & A_{32} & A_{33} & A_{34} & A_{35} & A_{36} & A_{37} & A_{38} \\ A_{41} & A_{42} & A_{43} & A_{44} & A_{45} & A_{46} & A_{47} & A_{48} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{F}$$

```
118 F=[beta1(1:N*L,:)';eye(N*(L-1),N*L)];
119 %matrix Q
```

$$VAR(e_t) = Q = \begin{pmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} & 0 & 0 & 0 & 0 \\ Q_{12} & Q_{22} & Q_{23} & Q_{24} & 0 & 0 & 0 & 0 \\ Q_{13} & Q_{23} & Q_{33} & Q_{34} & 0 & 0 & 0 & 0 \\ Q_{14} & Q_{24} & Q_{23} & Q_{44} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

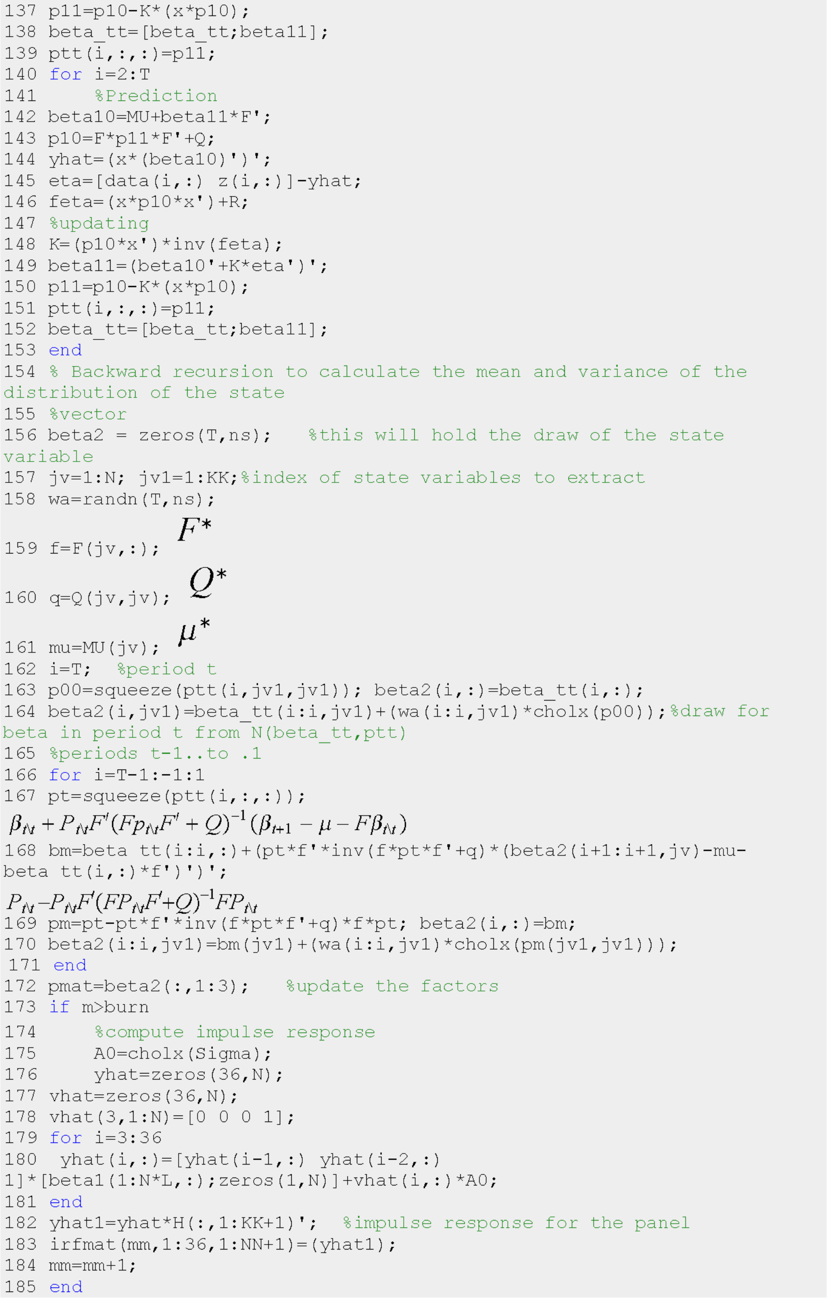
```
120 Q=zeros(rows(F),rows(F));
121 Q(1:N,1:N)=Sigma;
122 %Carter and Kohn algorithm to draw the factor
123 beta tt=[];          %will hold the filtered state variable
124 ptt=zeros(T,ns,ns);    % will hold its variance
125 % %%%%%%%%%%%Step 6a run Kalman Filter
126 i=1;
127 x=H; This is no longer data but a matrix of coefficients
128 %Prediction
129 beta10=MU+beta0*F';
130 p10=F*P00*F'+Q;
131 yhat=(x*(beta10)')';
132 eta=[data(i,:) z(i,:)]-yhat;
```

$$\underbrace{\begin{pmatrix} X_{1t} \\ X_{2t} \\ X_{3t} \\ . \\ . \\ . \\ . \\ X_{Mt} \\ FFR_t \end{pmatrix}}_{\tilde{X}_{it}}$$

```
133 feta=(x*p10*x')+R;
134 %updating
135 K=(p10*x')*inv(feta);
136 beta11=(beta10'+K*eta')';
```

Figure 15. Code for the FAVAR model (continued)

Lines 79 to 83 sample R_{ii} from the inverse Gamma distribution (using the function IG in the functions folder) with prior degrees of freedom and the prior scale matrix set to 0 (hence using information from the data only). Lines 85 and 86 set up the left hand side and the right hand side variables for the VAR model using the factors (pmat) and the policy rate. Lines 89 and 90 calculate the mean and variance of the conditional distribution of the VAR coefficients (without prior information these are just OLS). Line 93 draws the VAR coefficients ensuring stability. Lines 102 and 103 draw the covariance matrix Ω from the inverse Wishart distribution. We now have a draw for all parameters of the state space representation so we build the matrices necessary to cast the FAVAR into the state space form. Lines 110 to 112 build the matrix H seen in equation 7.2. Line 114 builds the covariance matrix of the error term



```
137 p11=p10-K*(x*p10);
138 beta_tt=[beta_tt;beta11];
139 ptt(i,:,:)=p11;
140 for i=2:T
141     %Prediction
142 beta10=MU+beta11*F';
143 p10=F*p11*F'+Q;
144 yhat=(x*(beta10)')';
145 eta=[data(i,:) z(i,:)]-yhat;
146 feta=(x*p10*x')+R;
147 %updating
148 K=(p10*x')*inv(feta);
149 beta11=(beta10'+K*eta')';
150 p11=p10-K*(x*p10);
151 ptt(i,:,:)=p11;
152 beta_tt=[beta_tt;beta11];
153 end
154 % Backward recursion to calculate the mean and variance of the
distribution of the state
155 %vector
156 beta2 = zeros(T,ns);   %this will hold the draw of the state
variable
157 jv=1:N; jv1=1:KK;%index of state variables to extract
158 wa=randn(T,ns);
```

F^*

```
159 f=F(jv,:);
```

Q^*

```
160 q=Q(jv,jv);
```

μ^*

```
161 mu=MU(jv);
162 i=T;  %period t
163 p00=squeeze(ptt(i,jv1,jv1)); beta2(i,:)=beta_tt(i,:);
164 beta2(i,jv1)=beta_tt(i:i,jv1)+(wa(i:i,jv1)*cholx(p00));%draw for
beta in period t from N(beta_tt,ptt)
165 %periods t-1..to .1
166 for i=T-1:-1:1
167 pt=squeeze(ptt(i,:,:));
```

$$\beta_{t\backslash t}+P_{t\backslash t}F'(Fp_{t\backslash t}F'+Q)^{-1}(\beta_{t+1}-\mu-F\beta_{t\backslash t})$$

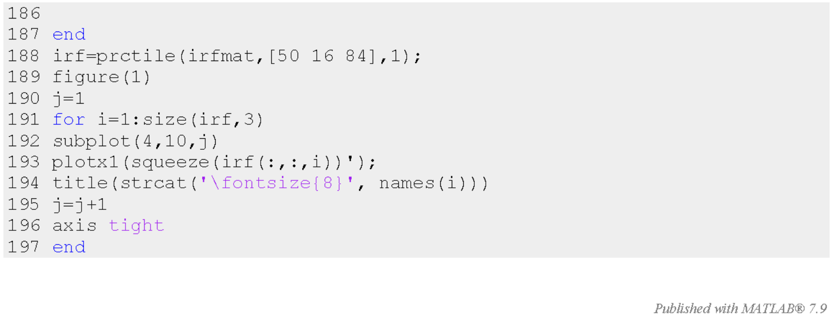
```
168 bm=beta tt(i:i,:)+(pt*f'*inv(f*pt*f'+q)*(beta2(i+1:i+1,jv)-mu-
beta tt(i,:)*f')')';
```

$$P_{t\backslash t}-P_{t\backslash t}F'(FP_{t\backslash t}F'+Q)^{-1}FP_{t\backslash t}$$

```
169 pm=pt-pt*f'*inv(f*pt*f'+q)*f*pt; beta2(i,:)=bm;
170 beta2(i:i,jv1)=bm(jv1)+(wa(i:i,jv1)*cholx(pm(jv1,jv1)));
 171 end
172 pmat=beta2(:,1:3);   %update the factors
173 if m>burn
174     %compute impulse response
175     A0=cholx(Sigma);
176     yhat=zeros(36,N);
177 vhat=zeros(36,N);
178 vhat(3,1:N)=[0 0 0 1];
179 for i=3:36
180  yhat(i,:)=[yhat(i-1,:) yhat(i-2,:)
1]*[beta1(1:N*L,:);zeros(1,N)]+vhat(i,:)*A0;
181 end
182 yhat1=yhat*H(:,1:KK+1)';  %impulse response for the panel
183 irfmat(mm,1:36,1:NN+1)=(yhat1);
184 mm=mm+1;
185 end
```

FIGURE 16. Code for the FAVAR model (continued)

R. Line 116 builds the matrix μ seen in equation 7.4. Line 118 builds the matrix F, while line 120 builds the matrix Q. With the matrices of the state space representation in hand we start the Carter and Kohn algorithm by running the Kalman filter from lines 123 to 153. Note a minor difference to the previous example is that the observation equation now does not have a regressor on the right hand side. Hence on line 127 x is set equal to the matrix H. Line 156 starts the backward recursion. Recall that the last 5 state variables represent identities and Q is singular. Therefore we will only work with the first 3 rows (and columns for covariance matrices) of μ, F, Q and B_{t+1}. Lines 159 to 161 create μ^*, F^*, Q^*. Lines 168 and 169 are the modified Carter and Kohn updating equations. Line 172 sets the factors pmat equal to the last draw using the Carter and Kohn algorithm and we return to the first step of the



```
186
187 end
188 irf=prctile(irfmat,[50 16 84],1);
189 figure(1)
190 j=1
191 for i=1:size(irf,3)
192 subplot(4,10,j)
193 plotx1(squeeze(irf(:,:,i))');
194 title(strcat('\fontsize{8}', names(i)))
195 j=j+1
196 axis tight
197 end
```

Published with MATLAB® 7.9

Figure 17. Code for the FAVAR model (continued)

Gibbs sampler. Once past the burn-in period we calculate an impulse response of the factors to a shock to the bank rate in the transition equation using a Cholesky decomposition of the covariance matrix to form the $A0$ matrix. Line 182 uses the observation equation of the model to calculate the impulse response of all the underlying data series. The estimated impulse responses are shown in 18.

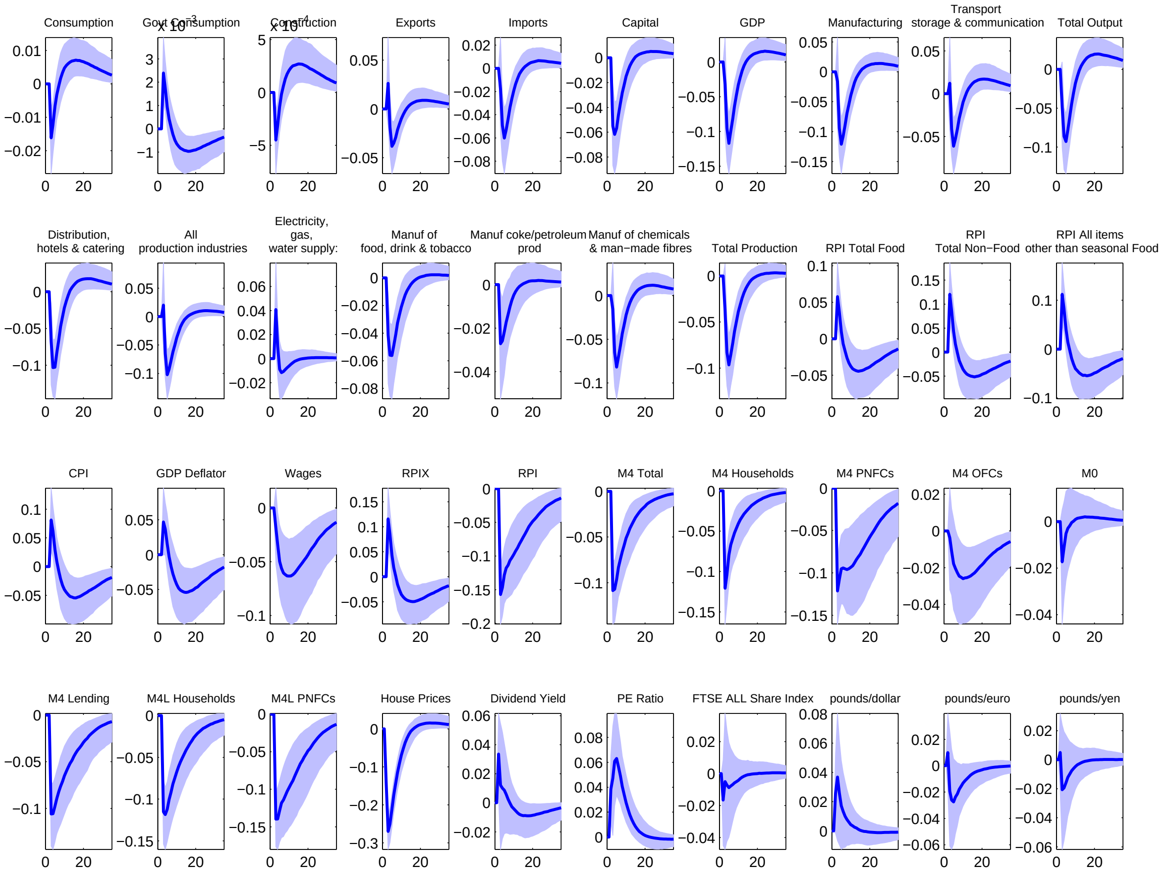


FIGURE 18. Impulse response of UK Macroeconomic series to a monetary policy shock using a FAVAR model.

8. Gibbs Sampling for a Mixed Frequency VAR

Suppose that a researcher wants to estimate a VAR model using two variables: (1) Y_t a quarterly series and (2) X_t a monthly series. The data matrix Z_t might look as follows:

$$Z_t = \begin{pmatrix} na & X_1 \\ na & X_2 \\ Y_3 & X_3 \\ na & X_4 \\ na & X_5 \\ Y_6 & X_6 \\ . & . \\ . & . \\ Y_T & X_T \end{pmatrix}$$

In other words, if one were to consider the VAR at monthly frequency then Y_t has missing observations in two months of every quarter. However, we can treat these missing observations as unobserved monthly observations on Y_t and re-write the model as a state space model (see Schorfheide and Song (2015)).

The observation equation for this model is defined as follows

$$\underbrace{\begin{pmatrix} 0 & X_1 \\ 0 & X_2 \\ Y_3 & X_3 \\ 0 & X_4 \\ 0 & X_5 \\ Y_6 & X_6 \\ . & . \\ . & . \\ Y_T & X_T \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} \text{ for } t = 3, 6, 9... \tag{8.1}$$

This equation states that when an observation for Y_t is available (in period 3, 6, 9 etc), the quarterly observed data is an average of the unobserved monthly data. For example, the equation implies that $Y_3 = 1/3\hat{Y}_t + 1/3\hat{Y}_{t-1} + 1/3\hat{Y}_{t-2}$ where $\hat{Y}_t$ denotes the unobserved monthly data on Y_t. This can be changed to reflect other assumptions. For example it can be assumed that the observed data is the sum of monthly observations by changing 1/3 to 1. As X_t is observed at the monthly frequency the second row of the H matrix specifies an identity.

When an observation for Y_t is unavailable, the observation equation changes to:

$$\underbrace{\begin{pmatrix} 0 & X_1 \\ 0 & X_2 \\ Y_3 & X_3 \\ 0 & X_4 \\ 0 & X_5 \\ Y_6 & X_6 \\ . & . \\ . & . \\ Y_T & X_T \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} + \begin{pmatrix} e_t \\ 0 \end{pmatrix} \text{ for } t = 1, 2, 4... \tag{8.2}$$

where $var(e_t)$ is large. When observations Y_t are missing, the first row of H is zero. The variance of e_t is set to a large number. Recall from the description of the update step of the Kalman filter that this assumption effectively means that missing observations on Y_t are ignored when calculating the updated estimate of $\hat{Y}_t$. Therefore, the observation equation for this model changes over time depending on whether observations on Y_t are missing.

The transition equation stays fixed over time and is defined as

$$\underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} = \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\mu} + \underbrace{\begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}}_{F} \underbrace{\begin{pmatrix} \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \\ \hat{Y}_{t-3} \\ X_{t-3} \end{pmatrix}}_{\beta_{t-1}} + \underbrace{\begin{pmatrix} v_{1,t} \\ v_{2,t} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{v_t} \tag{8.3}$$

where $var(v) = Q = \begin{pmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$

If $\hat{Y}_t$ was observed, then the model collapses to a BVAR. This observation provides the intuition behind the Gibbs algorithm for this model. The algorithm consists of the following steps:

(1) Set priors and starting values. The prior for the VAR parameters μ, F and Q can be set using any of the priors for VARs considered in the previous chapter. For example, one may consider setting an independent Normal inverse Wishart prior. Collecting the VAR coefficients in the matrix B and the non-zero elements of Q in the matrix $\Omega = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{12} & Q_{22} \end{pmatrix}$ this prior can be represented as $p(B) \sim N(B_0, \Sigma_B)$ and $p(\Omega) \sim IW(\Omega_0, V_0)$. The Kalman filter requires the initial value of the state vector $\beta_t = \begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}$.

An initial estimate of $\hat{Y}_t$ can be obtained by using a simple interpolation method. For example, using repeated observations to fill in the months with missing data.

(2) Conditional on $\hat{Y}_t$ and the error covariance matrix Ω, the posterior for the VAR coefficients B (recall $B = \{\mu, F\}$ the coefficients in the transition equation of the model) in vectorised form is normal (see Chapter 2) and given as $H\left(B|\hat{Y}_t, \Omega\right) \sim N(B^*, D^*)$ where B_0, Σ_B

$$\begin{aligned} B^* &= \left(\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t\right)^{-1}\left(\Sigma_B^{-1} vec(B_0) + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t vec(\hat{B})\right) \\ D^* &= \left(\Sigma_B^{-1} + \Omega^{-1} \otimes \bar{X}_t'\bar{X}_t\right)^{-1} \end{aligned}$$

where $\bar{X}_t = \{\hat{Y}_{t-1}, X_{t-1}, \hat{Y}_{t-2}, X_{t-2}, \hat{Y}_{t-3}, X_{t-3}, 1\}$ and $\hat{B}$ is the OLS estimate of B using $\bar{Y}_t = \{\hat{Y}_t, X_t\}, \bar{X}_t$.

3. Conditional on $\hat{Y}_t$ and the VAR coefficients B the error covariance Ω has a inverse Wishart posterior with scale matrix $\left(\bar{Y}_t - \bar{X}_t B\right)'\left(\bar{Y}_t - \bar{X}_t B\right) + \Omega_0$ and degrees of freedom $T + V_0$.

(3) Finally, given a draw of the VAR parameters, the state variable $\hat{Y}_t$ is drawn using the Carter and Kohn algorithm. As in the previous example, the backward recursion needs a modification to account for the fact that Q is singular. This modification implies that we use $\mu^*, F^*, Q^*, \beta^*_{t+1}$ in equations 3.13 and 3.14 where $\mu^*, F^*, Q^*, \beta^*_{t+1}$ denote the first jv rows of μ, F, Q, β_{t+1}. In our example $jv = 2$ as the top 2×2 block of Q is non-singular.

(4) Repeat steps 2 to 4 until convergence

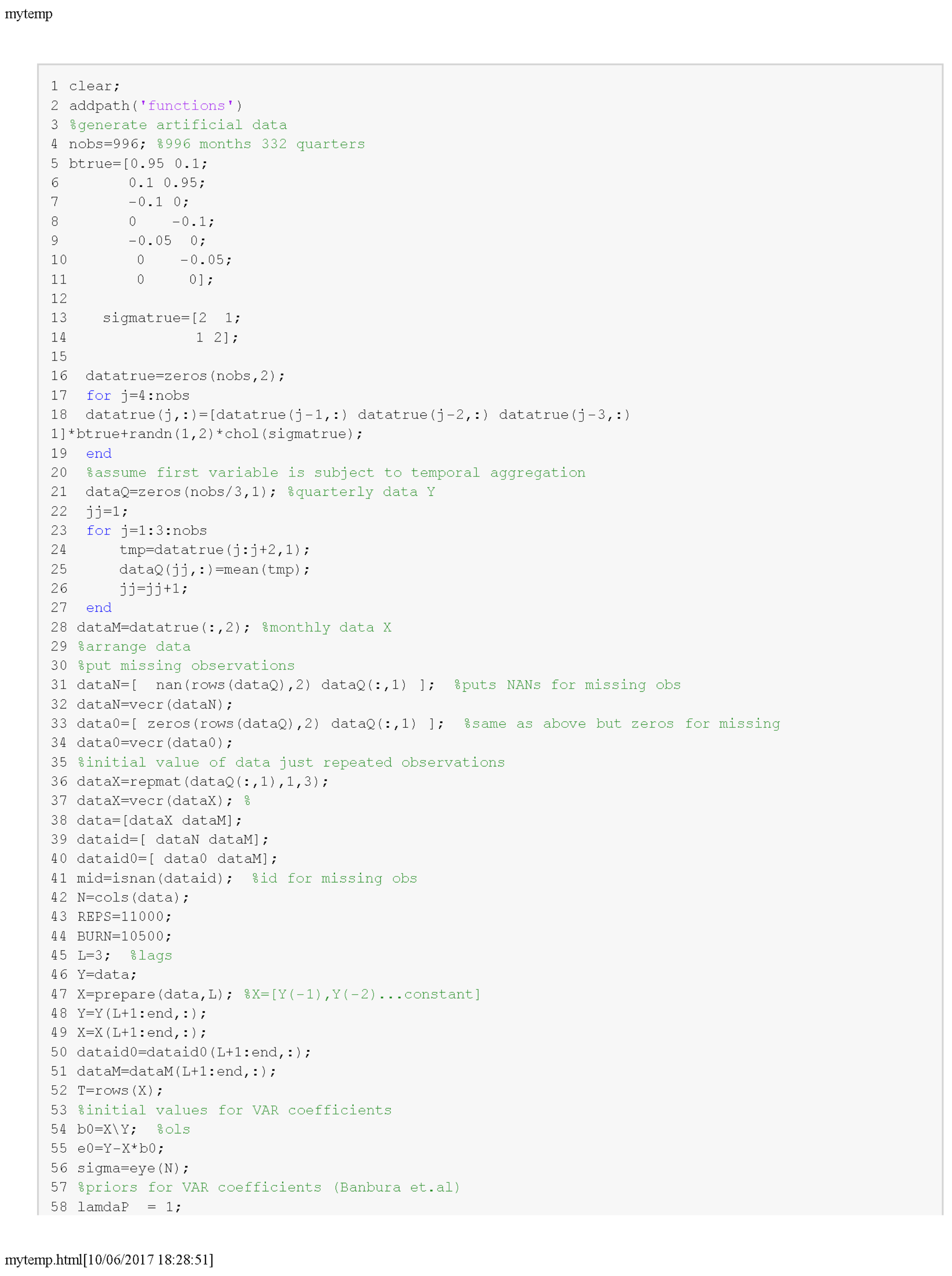
The code for the mixed frequency VAR is provided in figures 19 to 21. This code is based on artificial data generated for two variables at the monthly frequency. Lines 20 to 26, average the observations of the first variable to produce a quarterly series Y_t. The point of the example is to test if the mixed frequency VAR (that uses quarterly data for the first variable Y_t and monthly data X_t for the second variable) outlined above can be used to recover the original monthly series. Lines 36 and 37 form an initial estimate of $\hat{Y}_t$ using repeated observations. Note that the lag length is set to 3. This is the minimum lag length allowed by the structure of the observation equation. Lines 58 to 79 set the priors for the VAR model via artificial or dummy observations (see Chapter 2). Lines 83 to 87 set the initial state and its covariance to be used in the Kalman filter. The Gibbs sampler begins on line 90. The first step of the sampling algorithm is coded on lines 92 to 103 which draws the VAR coefficients. The VAR covariance is drawn on lines 105 to 107 from the inverse Wishart distribution. The final step of the algorithm using the Carter Kohn algorithm begins on line 109 with the Kalman filter. The matrices of the transition equation of the state space system are created on lines 110 to 112. Within the Kalman filter on line 121, we check if Y_t is missing and set the H matrix and the variance of error term e_t accordingly. The backward recursion is on lines 154 to 176. Note that once $\hat{Y}_t$ is drawn the data for the VAR is updated on lines 178 to 182.

Figure 22 shows that the posterior estimates of $\hat{Y}_t$ are close to the underlying true data.

9. Further reading

- Kim and Nelson (1999) chapter 3 is an excellent intuitive introduction to state space models.
- Hamilton (1994) chapter provides a more formal derivation of the Kalman filter.
- Kim and Nelson (1999) chapter 8 provides a detailed description of Gibbs sampling for state space models.
- Code and a monograph by Gary Koop:

https://sites.google.com/site/garykoop/home/computer-code-2.

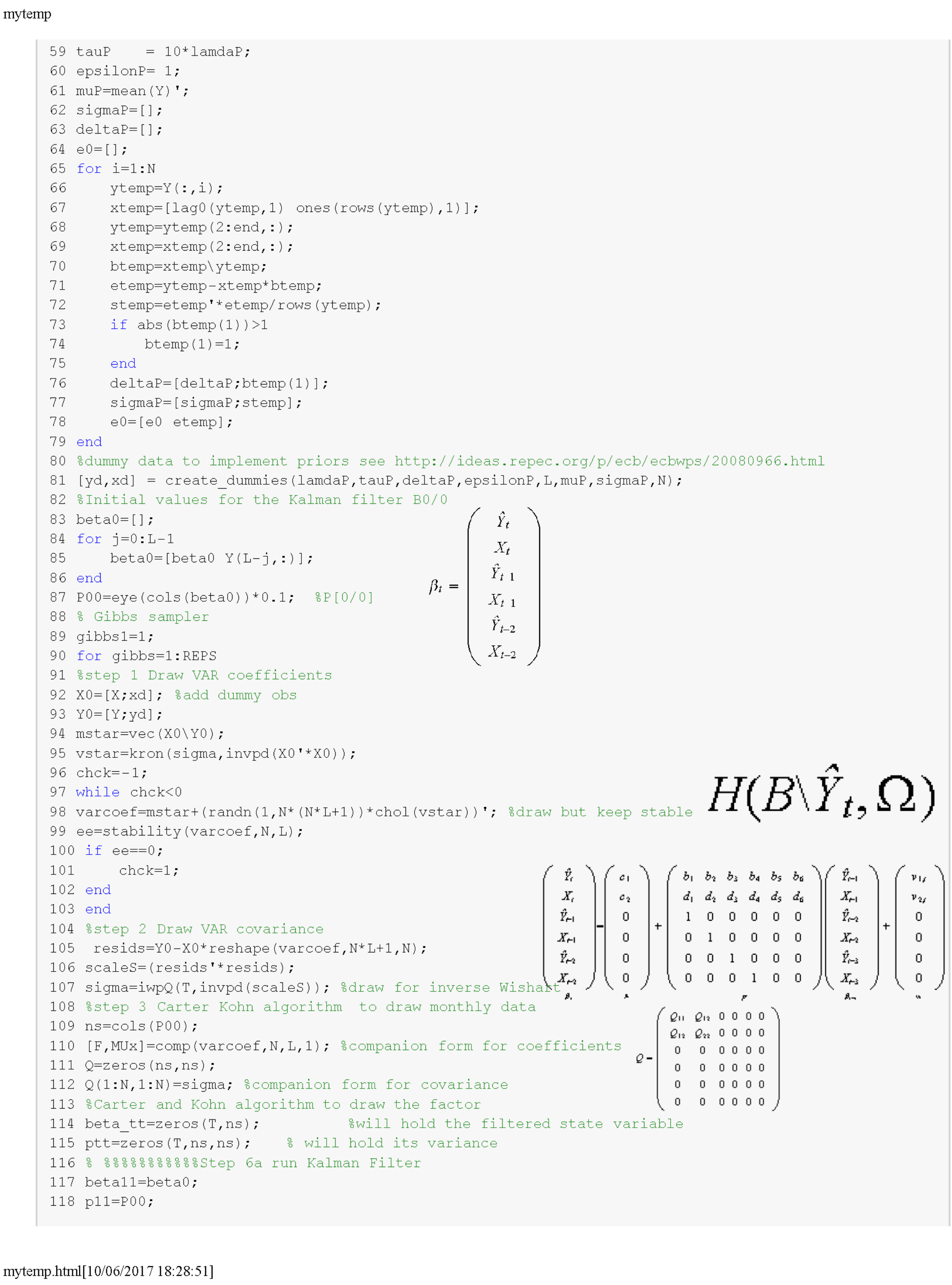


mytemp

```
clear;
addpath('functions')
%generate artificial data
nobs=996; %996 months 332 quarters
btrue=[0.95 0.1;
       0.1 0.95;
       -0.1 0;
       0    -0.1;
       -0.05  0;
       0    -0.05;
       0     0];

   sigmatrue=[2  1;
              1 2];

 datatrue=zeros(nobs,2);
 for j=4:nobs
 datatrue(j,:)=[datatrue(j-1,:) datatrue(j-2,:) datatrue(j-3,:) 1]*btrue+randn(1,2)*chol(sigmatrue);
 end
 %assume first variable is subject to temporal aggregation
 dataQ=zeros(nobs/3,1); %quarterly data Y
 jj=1;
 for j=1:3:nobs
     tmp=datatrue(j:j+2,1);
     dataQ(jj,:)=mean(tmp);
     jj=jj+1;
 end
dataM=datatrue(:,2); %monthly data X
%arrange data
%put missing observations
dataN=[  nan(rows(dataQ),2) dataQ(:,1) ];  %puts NANs for missing obs
dataN=vecr(dataN);
data0=[ zeros(rows(dataQ),2) dataQ(:,1) ];  %same as above but zeros for missing
data0=vecr(data0);
%initial value of data just repeated observations
dataX=repmat(dataQ(:,1),1,3);
dataX=vecr(dataX); %
data=[dataX dataM];
dataid=[ dataN dataM];
dataid0=[ data0 dataM];
mid=isnan(dataid);  %id for missing obs
N=cols(data);
REPS=11000;
BURN=10500;
L=3;  %lags
Y=data;
X=prepare(data,L); %X=[Y(-1),Y(-2)...constant]
Y=Y(L+1:end,:);
X=X(L+1:end,:);
dataid0=dataid0(L+1:end,:);
dataM=dataM(L+1:end,:);
T=rows(X);
%initial values for VAR coefficients
b0=X\Y;  %ols
e0=Y-X*b0;
sigma=eye(N);
%priors for VAR coefficients (Banbura et.al)
lamdaP  = 1;
```

mytemp.html[10/06/2017 18:28:51]

Figure 19. Code for mixed frequency VAR



mytemp

```
tauP    = 10*lamdaP;
epsilonP= 1;
muP=mean(Y)';
sigmaP=[];
deltaP=[];
e0=[];
for i=1:N
    ytemp=Y(:,i);
    xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
    ytemp=ytemp(2:end,:);
    xtemp=xtemp(2:end,:);
    btemp=xtemp\ytemp;
    etemp=ytemp-xtemp*btemp;
    stemp=etemp'*etemp/rows(ytemp);
    if abs(btemp(1))>1
        btemp(1)=1;
    end
    deltaP=[deltaP;btemp(1)];
    sigmaP=[sigmaP;stemp];
    e0=[e0 etemp];
end
%dummy data to implement priors see http://ideas.repec.org/p/ecb/ecbwps/20080966.html
[yd,xd] = create_dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
%Initial values for the Kalman filter B0/0
beta0=[];
for j=0:L-1
    beta0=[beta0 Y(L-j,:)];
end
P00=eye(cols(beta0))*0.1;  %P[0/0]
% Gibbs sampler
gibbs1=1;
for gibbs=1:REPS
%step 1 Draw VAR coefficients
X0=[X;xd]; %add dummy obs
Y0=[Y;yd];
mstar=vec(X0\Y0);
vstar=kron(sigma,invpd(X0'*X0));
chck=-1;
while chck<0
varcoef=mstar+(randn(1,N*(N*L+1))*chol(vstar))'; %draw but keep stable
ee=stability(varcoef,N,L);
if ee==0;
    chck=1;
end
end
%step 2 Draw VAR covariance
 resids=Y0-X0*reshape(varcoef,N*L+1,N);
scaleS=(resids'*resids);
sigma=iwpQ(T,invpd(scaleS)); %draw for inverse Wishart
%step 3 Carter Kohn algorithm  to draw monthly data
ns=cols(P00);
 [F,MUx]=comp(varcoef,N,L,1); %companion form for coefficients
 Q=zeros(ns,ns);
 Q(1:N,1:N)=sigma; %companion form for covariance
 %Carter and Kohn algorithm to draw the factor
 beta_tt=zeros(T,ns);          %will hold the filtered state variable
 ptt=zeros(T,ns,ns);    % will hold its variance
 % %%%%%%%%%%%Step 6a run Kalman Filter
 beta11=beta0;
 p11=P00;
```

$$\beta_t = \begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t\ 1} \\ X_{t\ 1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}$$

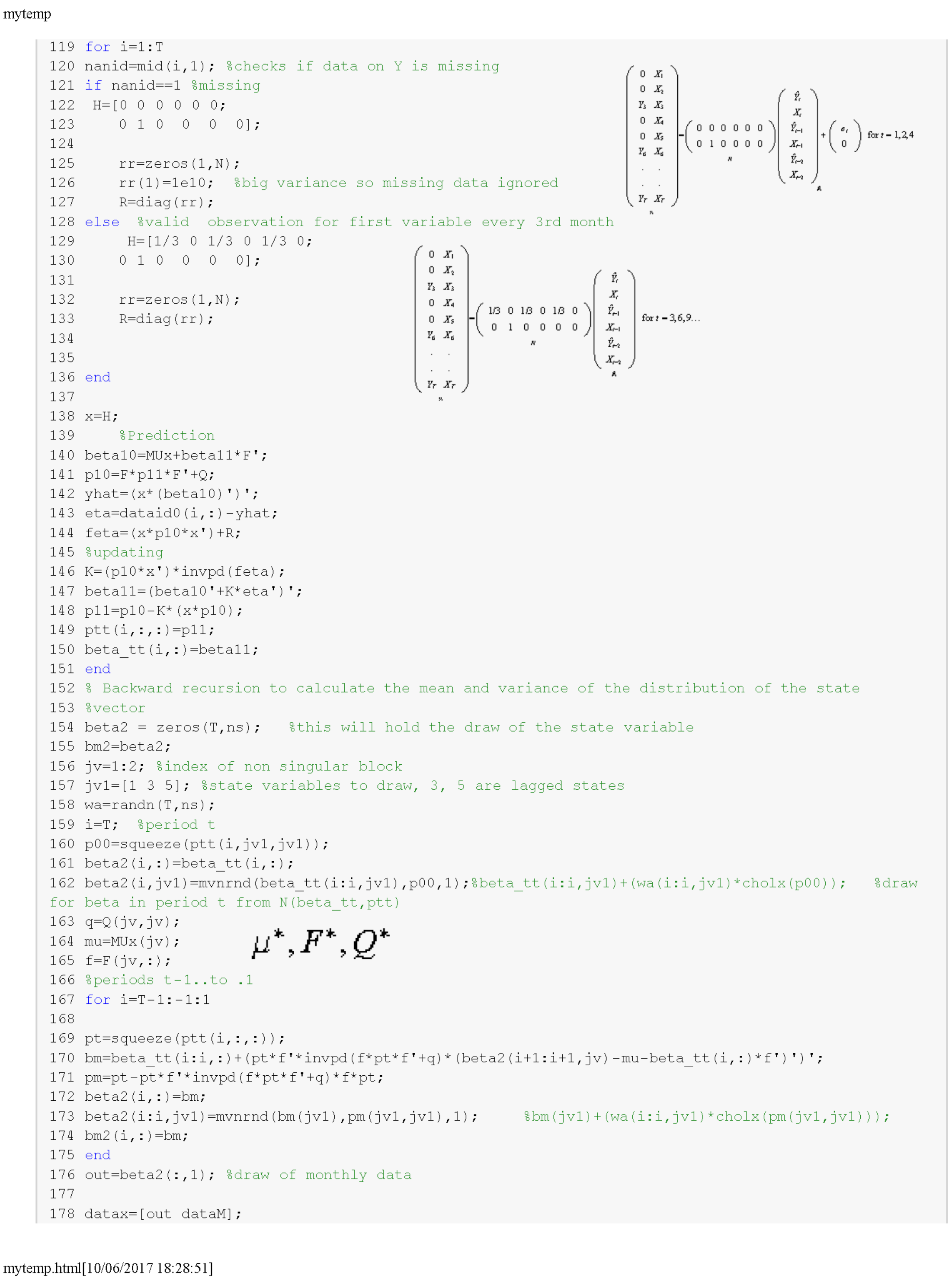
$$H(B\backslash\hat{Y}_t,\Omega)$$

$$\underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} = \underbrace{\begin{pmatrix} c_1 \\ c_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\mu} + \underbrace{\begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}}_{F} \underbrace{\begin{pmatrix} \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \\ \hat{Y}_{t-3} \\ X_{t-3} \end{pmatrix}}_{\beta_{t-1}} + \underbrace{\begin{pmatrix} v_{1t} \\ v_{2t} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{v}$$

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

mytemp.html[10/06/2017 18:28:51]

Figure 20. Code for the mixed frequency VAR



mytemp

```
for i=1:T
nanid=mid(i,1); %checks if data on Y is missing
if nanid==1 %missing
 H=[0 0 0 0 0 0;
    0 1 0  0  0  0];

    rr=zeros(1,N);
    rr(1)=1e10;  %big variance so missing data ignored
    R=diag(rr);
```

$$\underbrace{\begin{pmatrix} 0 & X_1 \\ 0 & X_2 \\ Y_3 & X_3 \\ 0 & X_4 \\ 0 & X_5 \\ Y_6 & X_6 \\ . & . \\ . & . \\ Y_T & X_T \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} + \begin{pmatrix} e_t \\ 0 \end{pmatrix} \text{ for } t = 1,2,4$$

```
else  %valid  observation for first variable every 3rd month
     H=[1/3 0 1/3 0 1/3 0;
    0 1 0  0  0  0];

    rr=zeros(1,N);
    R=diag(rr);


end
```

$$\underbrace{\begin{pmatrix} 0 & X_1 \\ 0 & X_2 \\ Y_3 & X_3 \\ 0 & X_4 \\ 0 & X_5 \\ Y_6 & X_6 \\ . & . \\ . & . \\ Y_T & X_T \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} 1/3 & 0 & 1/3 & 0 & 1/3 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} \hat{Y}_t \\ X_t \\ \hat{Y}_{t-1} \\ X_{t-1} \\ \hat{Y}_{t-2} \\ X_{t-2} \end{pmatrix}}_{\beta_t} \text{ for } t = 3,6,9\ldots$$

```

x=H;
    %Prediction
beta10=MUx+beta11*F';
p10=F*p11*F'+Q;
yhat=(x*(beta10)')';
eta=dataid0(i,:)-yhat;
feta=(x*p10*x')+R;
%updating
K=(p10*x')*invpd(feta);
beta11=(beta10'+K*eta')';
p11=p10-K*(x*p10);
ptt(i,:,:)=p11;
beta_tt(i,:)=beta11;
end
% Backward recursion to calculate the mean and variance of the distribution of the state
%vector
beta2 = zeros(T,ns);   %this will hold the draw of the state variable
bm2=beta2;
jv=1:2; %index of non singular block
jv1=[1 3 5]; %state variables to draw, 3, 5 are lagged states
wa=randn(T,ns);
i=T;  %period t
p00=squeeze(ptt(i,jv1,jv1));
beta2(i,:)=beta_tt(i,:);
beta2(i,jv1)=mvnrnd(beta_tt(i:i,jv1),p00,1);%beta_tt(i:i,jv1)+(wa(i:i,jv1)*cholx(p00));   %draw for beta in period t from N(beta_tt,ptt)
q=Q(jv,jv);
mu=MUx(jv);
f=F(jv,:);
```

$$\mu^*, F^*, Q^*$$

```
%periods t-1..to .1
for i=T-1:-1:1

pt=squeeze(ptt(i,:,:));
bm=beta_tt(i:i,:)+(pt*f'*invpd(f*pt*f'+q)*(beta2(i+1:i+1,jv)-mu-beta_tt(i,:)*f')')';
pm=pt-pt*f'*invpd(f*pt*f'+q)*f*pt;
beta2(i,:)=bm;
beta2(i:i,jv1)=mvnrnd(bm(jv1),pm(jv1,jv1),1);     %bm(jv1)+(wa(i:i,jv1)*cholx(pm(jv1,jv1)));
bm2(i,:)=bm;
end
out=beta2(:,1); %draw of monthly data

datax=[out dataM];
```

mytemp.html[10/06/2017 18:28:51]

FIGURE 21. Code for the mixed frequency VAR.

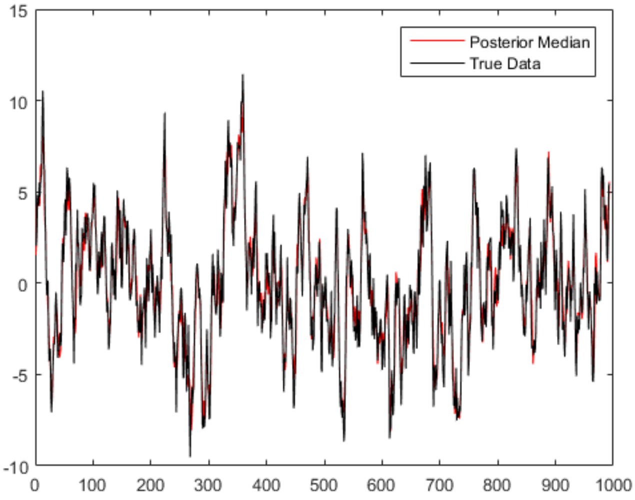


FIGURE 22. Posterior estimate of $\hat{Y}_t$

CHAPTER 4

Gibbs Sampling for Markov switching models

The recent financial crisis has again highlighted the fact that relationships between economic variables may be subject to sudden shifts. The time-varying parameter model introduced in the previous chapter offers one method for dealing with such structural change. However, TVP models may be ill suited to deal with this problem if the structural change is abrupt. This chapter discusses the estimation of Markov switching models that are well equipped to deal with abrupt regime shifts. As in the case of state-space models, a Gibbs sampling approach to estimation offers a powerful method to estimate these models. The material in this chapter draws heavily on material in Hamilton (1994) and Kim and Nelson (1999).

1. Switching regressions

Before considering Markov switching regressions, recall that a basic regression with dummy variables (or a switching regression) is defined as:

$$\begin{aligned} y_t &= x_t b_{S_t} + v_t, v_t \tilde{} N(0, \sigma^2_{S_t}) \\ b_{S_t} &= b_0(1-S_t) + b_1 S_t \\ \sigma^2_{S_t} &= \sigma^2_0(1-S_t) + \sigma^2_1 S_t \end{aligned} \tag{1.1}$$

where S_t for $t = 1, 2, ...T$ denotes a dummy variable that indicates when a structural (or regime) shift takes place. If S_t is known then this just a linear regression and methods introduced in Chapter 1 apply. We are interested in a situation where S_t is unknown – i.e. the researcher has to estimate when the regime change occurred and the associated regression parameters in each regime. It is instructive to consider how the likelihood function of the model can be obtained. The likelihood function at time t in this case is defined as

$$f(y_t|I_{t-1}) = \sum_{i=0}^{1} f(y_t|S_t = i, I_{t-1}) \times f(S_t = i|I_{t-1}) \tag{1.2}$$

where I_t denotes information at time t. Here the first term on the RHS is the likelihood conditional on the value of S_t. The second term is the probability of being in the regime. Thus for regime $i = 1$ this equals: $\underbrace{\frac{1}{\sqrt{2\pi\sigma^2_1}} \exp\left(\frac{-(y_t - x_t b_1)'(y_t - x_t b_1)'}{2\sigma^2_1}\right)}_{\text{Likelihood}} \underbrace{\Pr[S_t = 1]}_{\text{probability}}$. Therefore the likelihood function can be written as

$$\begin{aligned} f(y_t|I_{t-1}) = {} & \frac{1}{\sqrt{2\pi\sigma^2_0}} \exp\left(\frac{-(y_t - x_t b_0)'(y_t - x_t b_0)'}{2\sigma^2_0}\right) \Pr[S_t = 0] \\ & + \frac{1}{\sqrt{2\pi\sigma^2_1}} \exp\left(\frac{-(y_t - x_t b_1)'(y_t - x_t b_1)'}{2\sigma^2_1}\right) \Pr[S_t = 1] \end{aligned} \tag{1.3}$$

The log likelihood of the model is $\sum_{t=1}^{T} \ln f(y_t|I_{t-1})$. The key thing to note about equation 1.3 is that it represents a weighted average of the likelihood conditional on each regime with weights given by the probability of being in that regime at a given time. Thus to calculate the likelihood function one needs to calculate the term $\Pr[S_t = i]$ for each t. As S_t is unobserved, this problem is similar to the estimation of an unobserved state variable dealt with in the previous chapter. In other words, a filtering algorithm (like the Kalman filter) is required. But before considering this approach, one needs to define the 'transition equation' for S_t. It is this choice which leads to the definition of Markov switching models.

2. Markov Switching regressions

The Markov switching (MS) regression with two regimes labelled 0 and 1 is defined as

$$y_t = x_t b_{S_t} + v_t, v_t \tilde{} N(0, \sigma^2_{S_t})$$

$$\begin{aligned} b_{S_t} &= b_0(1-S_t) + b_1 S_t \\ \sigma^2_{S_t} &= \sigma^2_0(1-S_t) + \sigma^2_1 S_t \end{aligned}$$

We assume that S_t is unobserved (but takes on two values 0 and 1) and follows a first order Markov chain. In other words, S_t depends on S_{t-1} with associated probabilities given by

$$\begin{aligned}\Pr\left[S_t = 0|S_{t-1} = 0\right] &= p_{00} \\ \Pr\left[S_t = 1|S_{t-1} = 0\right] &= p_{01} = 1 - p_{00} \\ \Pr\left[S_t = 1|S_{t-1} = 1\right] &= p_{11} \\ \Pr\left[S_t = 0|S_{t-1} = 1\right] &= p_{10} = 1 - p_{11}\end{aligned}$$

Thus p_{ij} refers to the probability that the current regime is j given that the regime in the previous period was i. Values for p_{00}, p_{11} close to 1 imply that once in one of these regimes, the process is highly likely to remain in the same regime for some time – i.e. the regimes are persistent. These transition probabilties can be conveniently summarised in a transition probability matrix

$$P = \begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{pmatrix}$$

Note that the columns of this matrix sum to 1. Of course the model can be extended to allow for M regimes. For example in the case of three regimes S_t can be equal to $0, 1$ or 2 with transition probability matrix:

$$P = \begin{pmatrix} p_{00} & p_{10} & p_{20} \\ p_{01} & p_{11} & p_{21} \\ p_{02} & p_{12} & p_{22} \end{pmatrix}$$

A filtering algorithm to calculate the probability terms $\Pr\left[S_t = i|I_t\right]$ is described in Hamilton (1994). Denote this $M \times 1$ vector of probabilities as $\xi_{t|t}$ where the subscript denotes the estimate at time t given information at that time period t. The Hamilton filter proceeds in two steps which are applied at each point in the sample $t = 1, 2, ..., T$. Assume that an intial value $\xi_{t-1|t-1}$ is available. The following steps are applied time t:

(1) Prediction Step: The state variable is predicted one period forward

$$\xi_{t|t-1} = P\xi_{t-1|t-1} \tag{2.1}$$

where P is the matrix of transition probabilites. Note that $\xi_{t|t-1}$ is an estimate of $f\left(S_t = i|I_{t-1}\right)$. In other words, in the two regime case it is a 2×1 vector $\xi_{t|t-1} = \begin{pmatrix} \Pr\left[S_t = 0|I_{t-1}\right] \\ \Pr\left[S_t = 1|I_{t-1}\right] \end{pmatrix}$

(2) Update Step: Update the predicted estimate with information in the data at time t, i.e. estimate $\xi_{t|t} = f(S_t|I_t)$.Note that $f(S_t|I_t) = f(S_t|I_{t-1}, y_t)$. This can be obtained via formula

$$\xi_{t|t} = f(S_t|I_{t-1}, y_t) = \frac{f\left(y_t|S_t = i, I_{t-1}\right) \odot f\left(S_t = i|I_{t-1}\right)}{\sum\limits_{i=0}^{M-1} f\left(y_t|S_t = i, I_{t-1}\right) \odot f\left(S_t = i|I_{t-1}\right)} \tag{2.2}$$

where $\odot$ denotes element by element multiplication. In the two regime case, the numerator of equation 2.2 is simply the vector: $\begin{pmatrix} \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-(y_t - x_t b_0)'(y_t - x_t b_0)'}{2\sigma_0^2}\right) \times \Pr\left[S_t = 0|I_{t-1}\right] \\ \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-(y_t - x_t b_1)'(y_t - x_t b_1)'}{2\sigma_1^2}\right) \times \Pr\left[S_t = 1|I_{t-1}\right] \end{pmatrix}$. This vector denotes the joint density $f\left(S_t, y_t|I_{t-1}\right)$. Notice that the denominator of of equation 2.2 sums across the M regimes and the weighted average is the marginal density $f\left(y_t|I_{t-1}\right)$ or the likelihood function. Thus equation 2.2 is simply a division of a joint density by the marginal to obtain the conditional distribution. $\xi_{t|t}$ is the input on the RHS of equation 2.1 in the next time period.

To start the filter, the initial value $\xi_{0|0}$ can be calculated as the unconditional probability

$$\xi_{0|0} = \pi = (A'A)^{-1}A'E$$

where $A = \begin{pmatrix} I_M - P \\ 1_{1\times M} \end{pmatrix}$ and $E = \begin{pmatrix} 0_{M\times 1} \\ 1 \end{pmatrix}$.

Applying these two steps provides the likelihood function of the model $f\left(y_t|I_{t-1}\right)$ and $\xi_{t|t}$ or $f(S_t|I_{t-1}, y_t)$ for $t = 1, 2, ...T$. These latter probabilities will be used in the Gibbs sampling algorithm for estimating this model.

3. A Gibbs sampling algorithm for MS models

Consider the two regime MS regression introduced above

$$\begin{aligned}y_t &= x_t b_{S_t} + v_t, v_t\tilde{}N(0, \sigma_{S_t}^2) \\ b_{S_t} &= b_0(1 - S_t) + b_1 S_t \\ \sigma_{S_t}^2 &= \sigma_0^2(1 - S_t) + \sigma_1^2 S_t\end{aligned} \tag{3.1}$$

where S_t follows a first order Markov chain with transition probability matrix P. The model has four sets of unknowns: the $M = 2$ coefficients b_{S_t}, the $M = 2$ variances $\sigma_{S_t}^2$, the elements of P and the state variable $\tilde{S}_t = [S_1....S_T]$. A Gibbs algorithm thus samples from the following conditional posterior distributions:

(1) Conditional on $P, \sigma^2_{S_t}$ and $\tilde{S}_t$ sample b_{S_t} from its conditional posterior distribution.
(2) Conditional on P, b_{S_t} and $\tilde{S}_t$ sample $\sigma^2_{S_t}$ from its conditional posterior distribution.
(3) Conditional on $\sigma^2_{S_t}, b_{S_t}$ and $\tilde{S}_t$ sample P from its conditional posterior distribution.
(4) Conditional on $\sigma^2_{S_t}, b_{S_t}$ and P sample $\tilde{S}_t$ from its conditional posterior distribution.

With a value of $\tilde{S}_t$ in hand, the model collapses to a set of linear regressions on subsamples:

$$\begin{aligned} y_{0,t} &= b_0 x_{0,t} + v_{0,t}, v_{0,t}\tilde{}N(0,\sigma_0^2) \\ y_{1,t} &= b_1 x_{1,t} + v_{1,t}, v_{1,t}\tilde{}N(0,\sigma_1^2) \end{aligned}$$

where $y_{0,t}, x_{0,t}$ and $y_{1,t}, x_{1,t}$ represent the data selected when $\tilde{S}_t = 0$ and $\tilde{S}_t = 1$ respectively.

With a normal prior for b_0 and b_1, the conditional posterior in step 1 is also normal and is simply the posterior for the linear regression model conditional on knowing the error variance (see equation 2.10). The only difference is that given S_t two regressions apply, one in each sub-sample. Similarly, with an Inverse Gamma prior for σ_0^2 and σ_1^2, the conditional posterior in step 2 is also Inverse Gamma, i.e. the posterior for the error variance of a regression with known coefficients (see equation 2.16).

Therefore the first two steps of the algorithm are standard. Steps 3 and 4 require new concepts. We turn to these next.

3.1. The conditional posterior for P. A conjugate prior for each column of P is the Dirichlet distribution. With M regimes, this distribution depends on M parameters α_i for $i = 1, 2, ..M$. The PDF is: $f(x_1, .., x_{M-1}, \alpha_1, .., \alpha_M) \propto \Pi_{i=1}^{M} x_i^{\alpha_i - 1}$ with x_M given implicitly by $1 - \sum_{i=1}^{M-1} x_i$. The mean of the distribution is given as $\frac{\alpha_i}{\tilde{\alpha}}$ while the variance can be calculated as $\frac{\alpha_i(\tilde{\alpha}-\alpha_i)}{\tilde{\alpha}^2(\tilde{\alpha}+1)}$ where $\tilde{\alpha} = \sum_{i=1}^{M} \alpha_i$.

Consider the case of two regimes with $P = \begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{pmatrix}$. Then an example of the Dirichlet prior might be $p(p_{00})\tilde{}D(\alpha_{00}, \alpha_{01})$ and $p(p_{11})\tilde{}D(\alpha_{11}, \alpha_{10})$ where $D(.)$ represents the Dirichlet distribution. Suppose that we choose $\alpha_{00} = 15$ and $\alpha_{01} = 1$. This implies that the mean of the prior for p_{00} equals 0.94, while the variance is 0.003. Therefore, this prior would represent the strong belief that regime 0 is quite persistent.

Combining this prior with the likelihood results in a conditional posterior which is also Dirichlet

$$H(P_j|S_t)\tilde{}D(\alpha_{j1} + \eta_{j1}, \alpha_{j2} + \eta_{j2}, .., \alpha_{jM} + \eta_{jM}) \tag{3.2}$$

where $j = 1, 2, ..M$ refers to the column numbers of the transition probability matrix. Thus, in the two regime example $H(p_{00}|S)\tilde{}D(\alpha_{00} + \eta_{00}, \alpha_{01} + \eta_{01})$ and $H(p_{11}|S)\tilde{}D(\alpha_{11} + \eta_{11}, \alpha_{10} + \eta_{10})$.

The parameter η_{ji} refers to the number of times regime j is followed by regime i. This can be counted using the draw of $\tilde{S}_t$. Given this state variable, this conditional posterior does not depend on the data or the other parameters in the model.

Random numbers can be drawn from the Dirichlet distribution using the following algorithm:

ALGORITHM 4. *To draw from a M dimensional Dirichlet distribution $f(x_1, .., x_{M-1}, \alpha_1, .., \alpha_M)$, first draw $y_1, ..., y_M$ from the Gamma distribution with shape parameter $\alpha_1, .., \alpha_M$. Then the quantity $\frac{y_i}{\sum_{i=1}^{M} y_i}$ provides a draw from the Dirichlet distribution.*

3.2. The conditional posterior of $\tilde{S}_t$. This conditional posterior can be derived using the same method used to derive the Carter and Kohn recursion for state-space models (see Kim and Nelson (1999)). We want to derive the conditional distribution $H\left(\tilde{S}_t|\sigma^2_{S_t}, b_{S_t}, P, \tilde{Y}_t\right)$ where $\tilde{S}_t = [S_1....S_T]$ and the data is denoted by the matrix $\tilde{Y}_t = [y_1, x_1, ..., y_1, x_1]$.

This distribution can be simplified in the following way. First note that $H\left(\tilde{S}_t|\tilde{Y}_t\right) = H\left(S_1, S_2, ...S_T|\tilde{Y}_T\right)$ where we suppress conditioning on model parameters to make the notation simpler. The joint density $H\left(S_1, S_2, ...S_T|\tilde{Y}_T\right)$ can be factored as

$$\begin{aligned} & H\left(S_1, S_2, ...S_T|\tilde{Y}_T\right) \\ = & H\left(S_T|\tilde{Y}_T\right) H\left(S_1, S_2, ...S_{T-1}|\tilde{Y}_T, S_T\right) \\ = & H\left(S_T|\tilde{Y}_T\right) H\left(S_{T-1}|S_T, \tilde{Y}_T\right) H\left(S_1, S_2, ...S_{T-2}|\tilde{Y}_T, S_T, S_{T-1}\right) \\ = & H\left(S_T|\tilde{Y}_T\right) H\left(S_{T-1}|S_T, \tilde{Y}_T\right) H\left(S_{T-2}|S_T, S_{T-1}, \tilde{Y}_T\right) ...H\left(S_1|S_2, ..., S_{T-1}, S_T, \tilde{Y}_T\right) \end{aligned}$$

However, given the Markov property of S_t, only S_{t+1} and $\tilde{Y}_t$ are relevant for S_t.
For example, the term $H\left(S_1|S_2, ..., S_{T-1}, S_T, \tilde{Y}_T\right)$ can be simplified to $H\left(S_1|S_2, \tilde{Y}_1\right)$ because given $S_2, \tilde{Y}_1$ the data

and the state variable at other time periods do not contain any additional information about S_1. This implies that the last line can be written as

$$\begin{aligned} & H\left(S_T|\tilde{Y}_T\right)H\left(S_{T-1}|S_T,\tilde{Y}_{T-1}\right)H\left(S_{T-2}|S_{T-1},\tilde{Y}_{T-2}\right)...H\left(S_1|S_2,\tilde{Y}_1\right) \\ = & H\left(S_T|\tilde{Y}_T\right)\prod_{t=1}^{T-1}H\left(S_t|S_{t+1},\tilde{Y}_t\right) \end{aligned}$$

Therefore, the conditional posterior for the state variable is given by:

$$H\left(\tilde{S}_t|\sigma^2_{S_t},b_{S_t},P,\tilde{Y}_t\right)=H\left(S_T|\tilde{Y}_T\right)\prod_{t=1}^{T-1}H\left(S_t|S_{t+1},\tilde{Y}_t\right) \tag{3.3}$$

The key task is to sample $\tilde{S}_T$ from this density. As in the case of state-space models in the previous chapter this sampling proceeds in two steps:

(1) Drawing from $H\left(S_T|\tilde{Y}_T\right)$: Run the Hamilton filter to obtain the probability $\xi_{t|t}$ or $f(S_t|I_{t-1},y_t)$ for $t=1,2,...T$. At time T, one can draw S_T from the discrete distribution $H\left(S_T|\tilde{Y}_T\right)$ using $\xi_{T|T}$ as the probability associated with each value S_T takes. In the two regime case, one calculates $\Pr\left(S_T=0|\tilde{Y}_T\right)=\frac{f(S_T=0|I_{t-1},y_t)}{\sum_{i=0}^{M-1}f(S_T=i|I_{t-1},y_t)}$ and draws $u\tilde{\ }U(0,1)$. If $u\geq\Pr\left(S_T=0|\tilde{Y}_T\right)$, then $S_T=1$, else $S_T=0$.

(2) Drawing from $H\left(S_t|S_{t+1},\tilde{Y}_t\right)$. Note that $H\left(S_t|S_{t+1},\tilde{Y}_t\right)=\frac{H\left(S_t,S_{t+1}|\tilde{Y}_t\right)}{H\left(S_{t+1}|\tilde{Y}_t\right)}$. The numerator can again be factored into a 'conditional' and 'marginal': $H\left(S_t,S_{t+1}|\tilde{Y}_t\right)=H\left(S_{t+1}|S_t,\tilde{Y}_t\right)H\left(S_t|\tilde{Y}_t\right)$. Note that as all information about the states contained in $\tilde{Y}_t$ is present in S_t, $\tilde{Y}_t$ can be removed from the first term on the RHS: $H\left(S_t,S_{t+1}|\tilde{Y}_t\right)=H\left(S_{t+1}|S_t\right)H\left(S_t|\tilde{Y}_t\right)$. Therefore

$$\begin{aligned} H\left(S_t|S_{t+1},\tilde{Y}_t\right) &= \frac{H\left(S_{t+1}|S_t\right)H\left(S_t|\tilde{Y}_t\right)}{H\left(S_{t+1}|\tilde{Y}_t\right)} \\ &\propto H\left(S_{t+1}|S_t\right)H\left(S_t|\tilde{Y}_t\right) \end{aligned}$$

As discussed in Kim and Nelson (1999), $H\left(S_{t+1}|S_t\right)$ is just the transition probability while $H\left(S_t|\tilde{Y}_t\right)$ refers to the 'filter' probability $\xi_{t|t}=f(S_t|I_{t-1},y_t)$. *Drawing from* $H\left(S_{t+1}|S_t\right)H\left(S_t|\tilde{Y}_t\right)$ *proceeds backwards in time starting from* $T-1$ *and going back to period* 1. Consider the two regime case. Recall that $P=\begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11}\end{pmatrix}$ and denote the two elements of $f(S_t|I_{t-1},y_t)$ as $\Pr\left[S_t=0|I_t\right]$ and $\Pr\left[S_t=1|I_t\right]$. If $S_{t+1}=0$, then at time t one calculates

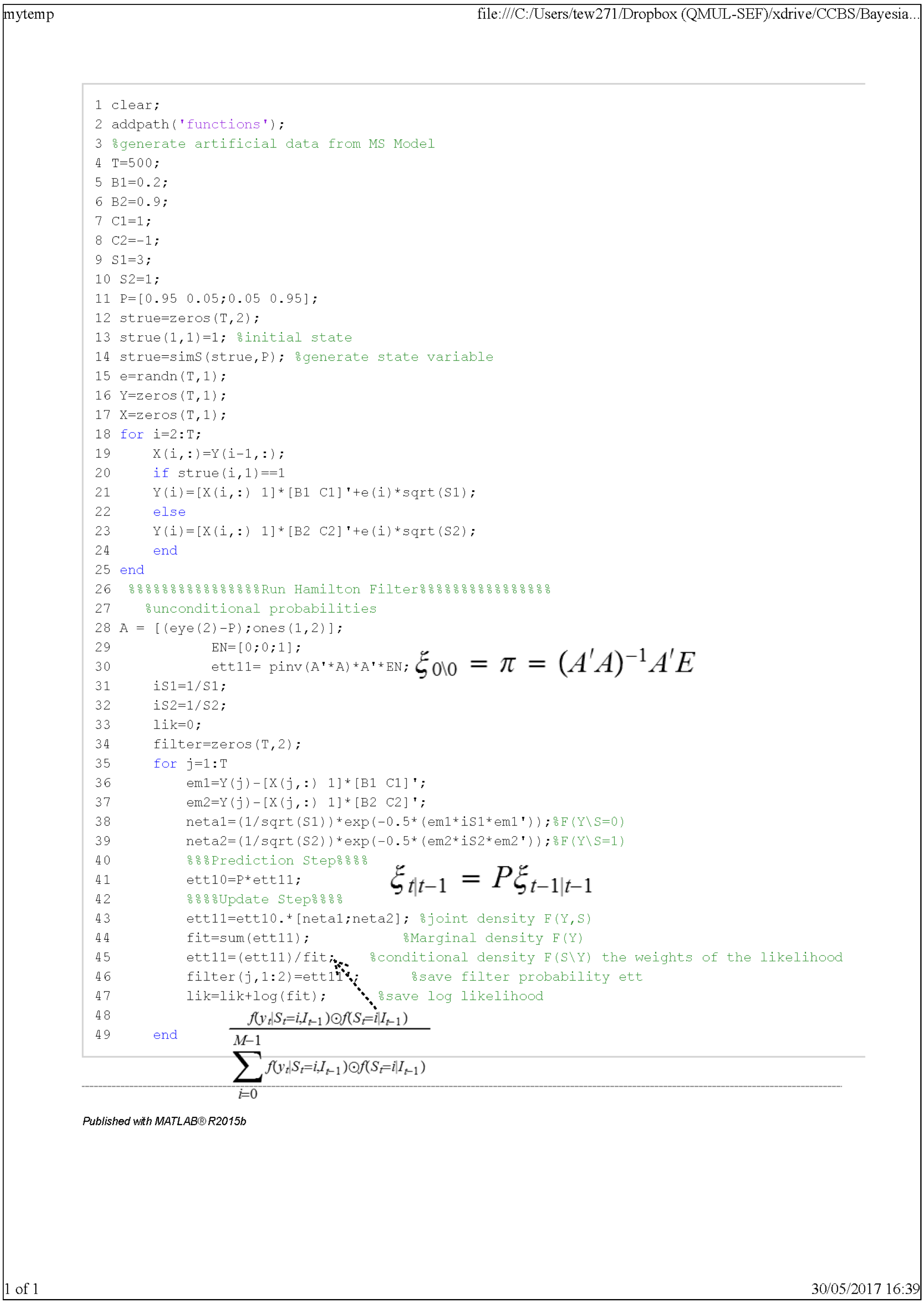
$$\begin{aligned} \Pr\left[S_t=0|S_{t+1}=0,\tilde{Y}_t\right] &= p_{00}\times\Pr\left[S_t=0|I_t\right] \\ \Pr\left[S_t=1|S_{t+1}=0,\tilde{Y}_t\right] &= p_{10}\times\Pr\left[S_t=1|I_t\right] \end{aligned}$$

and compares $\frac{\Pr\left[S_t=0|S_{t+1}=0,\tilde{Y}_t\right]}{\sum_{i=0}^{1}\Pr\left[S_t=i|S_{t+1}=0,\tilde{Y}_t\right]}$ to $u\tilde{\ }U(0,1)$. If u is greater or equal to than quantity $S_t=1$ otherwise $S_t=0$. The same procedure is repeated if $S_{t+1}=1$ using:

$$\begin{aligned} \Pr\left[S_t=0|S_{t+1}=1,\tilde{Y}_t\right] &= p_{01}\times\Pr\left[S_t=0|I_t\right] \\ \Pr\left[S_t=1|S_{t+1}=1,\tilde{Y}_t\right] &= p_{11}\times\Pr\left[S_t=1|I_t\right] \end{aligned}$$

This is repeated for $T-1,T-2,...,1$ to deliver a draw from the conditional posterior of $\tilde{S}_t$.

3.2.1. *Label switching.* The labels attached to each regime (for e.g. regime 0 and 1 in the 2 regime model) can switch during this algorithm. This is because the value of the likelihood is unaffected by switching the labels of the regime. Therefore, without some identifying restrictions, the marginal posteriors obtained from this algorithm can be be multi-modal. A simple way to proceed is to assume that one regime is associated with a higher (lower) value of a particular parameter. For example, one can assume that $\sigma^2_0>\sigma^2_1$ and use rejection sampling to ensure that the saved draws are consistent with this condition.



mytemp file:///C:/Users/tew271/Dropbox (QMUL-SEF)/xdrive/CCBS/Bayesia...

```
1 clear;
2 addpath('functions');
3 %generate artificial data from MS Model
4 T=500;
5 B1=0.2;
6 B2=0.9;
7 C1=1;
8 C2=-1;
9 S1=3;
10 S2=1;
11 P=[0.95 0.05;0.05 0.95];
12 strue=zeros(T,2);
13 strue(1,1)=1; %initial state
14 strue=simS(strue,P); %generate state variable
15 e=randn(T,1);
16 Y=zeros(T,1);
17 X=zeros(T,1);
18 for i=2:T;
19     X(i,:)=Y(i-1,:);
20     if strue(i,1)==1
21     Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S1);
22     else
23     Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S2);
24     end
25 end
26  %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
27    %unconditional probabilities
28 A = [(eye(2)-P);ones(1,2)];
29           EN=[0;0;1];
30           ett11= pinv(A'*A)*A'*EN;
```

$$\xi_{0\backslash 0} = \pi = (A'A)^{-1}A'E$$

```
31     iS1=1/S1;
32     iS2=1/S2;
33     lik=0;
34     filter=zeros(T,2);
35     for j=1:T
36         em1=Y(j)-[X(j,:) 1]*[B1 C1]';
37         em2=Y(j)-[X(j,:) 1]*[B2 C2]';
38         neta1=(1/sqrt(S1))*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
39         neta2=(1/sqrt(S2))*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
40         %%%Prediction Step%%%%
41         ett10=P*ett11;
```

$$\xi_{t|t-1} = P\xi_{t-1|t-1}$$

```
42         %%%%Update Step%%%%
43         ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
44         fit=sum(ett11);            %Marginal density F(Y)
45         ett11=(ett11)/fit;      %conditional density F(S\Y) the weights of the likelihood
46         filter(j,1:2)=ett11;         %save filter probability ett
47         lik=lik+log(fit);       %save log likelihood
48
49     end
```

$$\frac{f(y_t|S_t{=}i,I_{t-1})\odot f(S_t{=}i|I_{t-1})}{\sum_{i=0}^{M-1} f(y_t|S_t{=}i,I_{t-1})\odot f(S_t{=}i|I_{t-1})}$$

Published with MATLAB® R2015b

1 of 1 30/05/2017 16:39

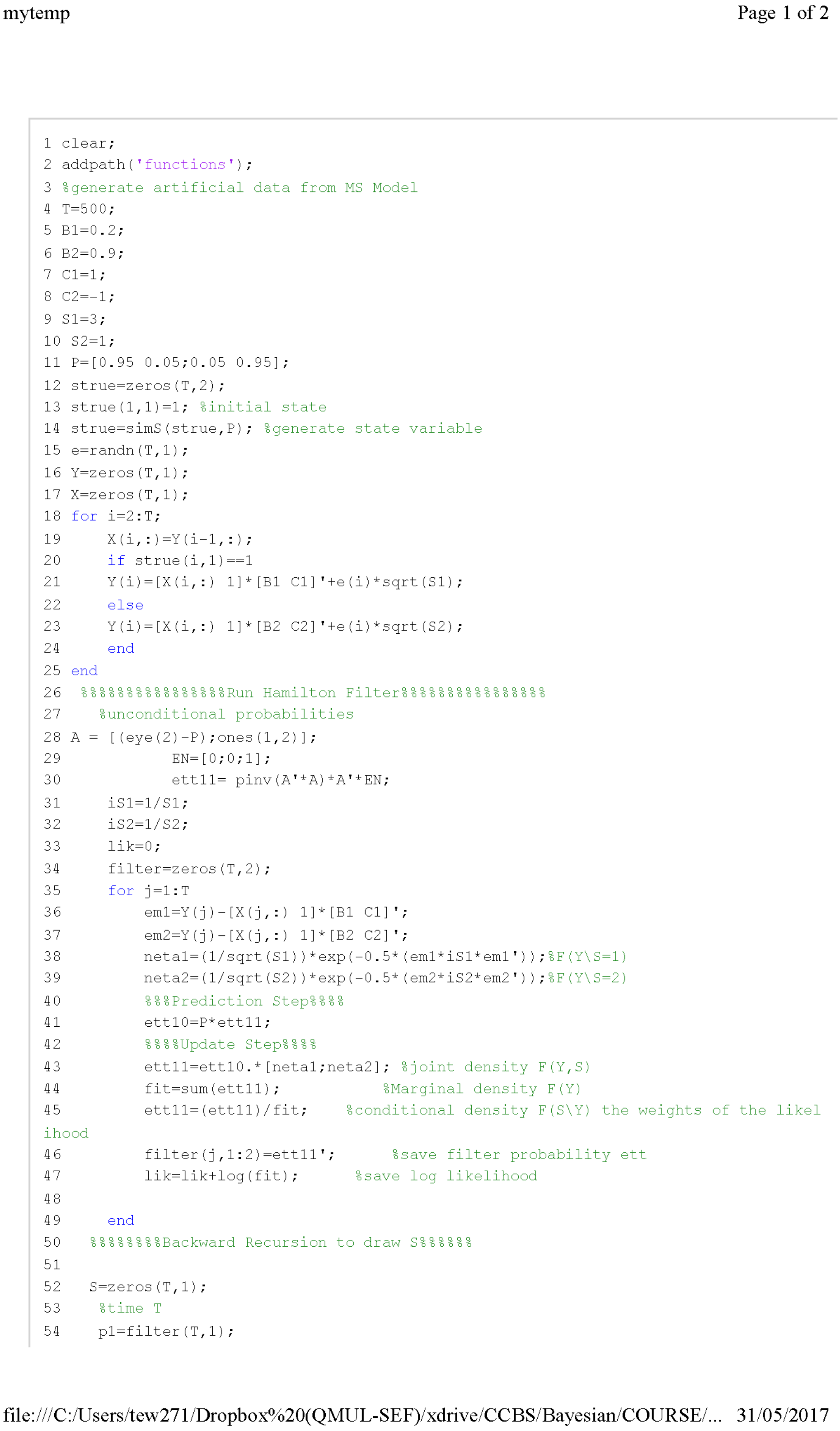
FIGURE 1. The Hamilton filter in Matlab

4. The Hamilton filter in Matlab

Implementing this Gibbs sampler in Matlab requires the researcher to be familiar with coding the Hamilton filter and the backward recursion discussed above in Matlab. In this section, we start with the Hamilton filter (see figure 1 and example1.m). Lines 4 to 25 generate artificial data from a simple 2 regime MS model

$$y_t = C_{S_t} + B_{S_t} + e_t, e_t \tilde{} N(0, \sigma^2_{S_t}) \qquad (4.1)$$

where S_t follows a first order Markov Chain. Line 28 creates the initial state $\xi_{0|0} = (A'A)^{-1}A'E$. The prediction step of the algorithm is on line 41. The conditional densities calculated on lines 38 and 39 are used in line 43 to



mytemp Page 1 of 2

```
1 clear;
2 addpath('functions');
3 %generate artificial data from MS Model
4 T=500;
5 B1=0.2;
6 B2=0.9;
7 C1=1;
8 C2=-1;
9 S1=3;
10 S2=1;
11 P=[0.95 0.05;0.05 0.95];
12 strue=zeros(T,2);
13 strue(1,1)=1; %initial state
14 strue=simS(strue,P); %generate state variable
15 e=randn(T,1);
16 Y=zeros(T,1);
17 X=zeros(T,1);
18 for i=2:T;
19     X(i,:)=Y(i-1,:);
20     if strue(i,1)==1
21     Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S1);
22     else
23     Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S2);
24     end
25 end
26  %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
27    %unconditional probabilities
28 A = [(eye(2)-P);ones(1,2)];
29           EN=[0;0;1];
30           ett11= pinv(A'*A)*A'*EN;
31     iS1=1/S1;
32     iS2=1/S2;
33     lik=0;
34     filter=zeros(T,2);
35     for j=1:T
36         em1=Y(j)-[X(j,:) 1]*[B1 C1]';
37         em2=Y(j)-[X(j,:) 1]*[B2 C2]';
38         neta1=(1/sqrt(S1))*exp(-0.5*(em1*iS1*em1'));%F(Y\S=1)
39         neta2=(1/sqrt(S2))*exp(-0.5*(em2*iS2*em2'));%F(Y\S=2)
40         %%%Prediction Step%%%%
41         ett10=P*ett11;
42         %%%%Update Step%%%%
43         ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
44         fit=sum(ett11);           %Marginal density F(Y)
45         ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likel
ihood
46         filter(j,1:2)=ett11';      %save filter probability ett
47         lik=lik+log(fit);      %save log likelihood
48
49     end
50   %%%%%%%%Backward Recursion to draw S%%%%%%
51
52   S=zeros(T,1);
53    %time T
54    p1=filter(T,1);
```

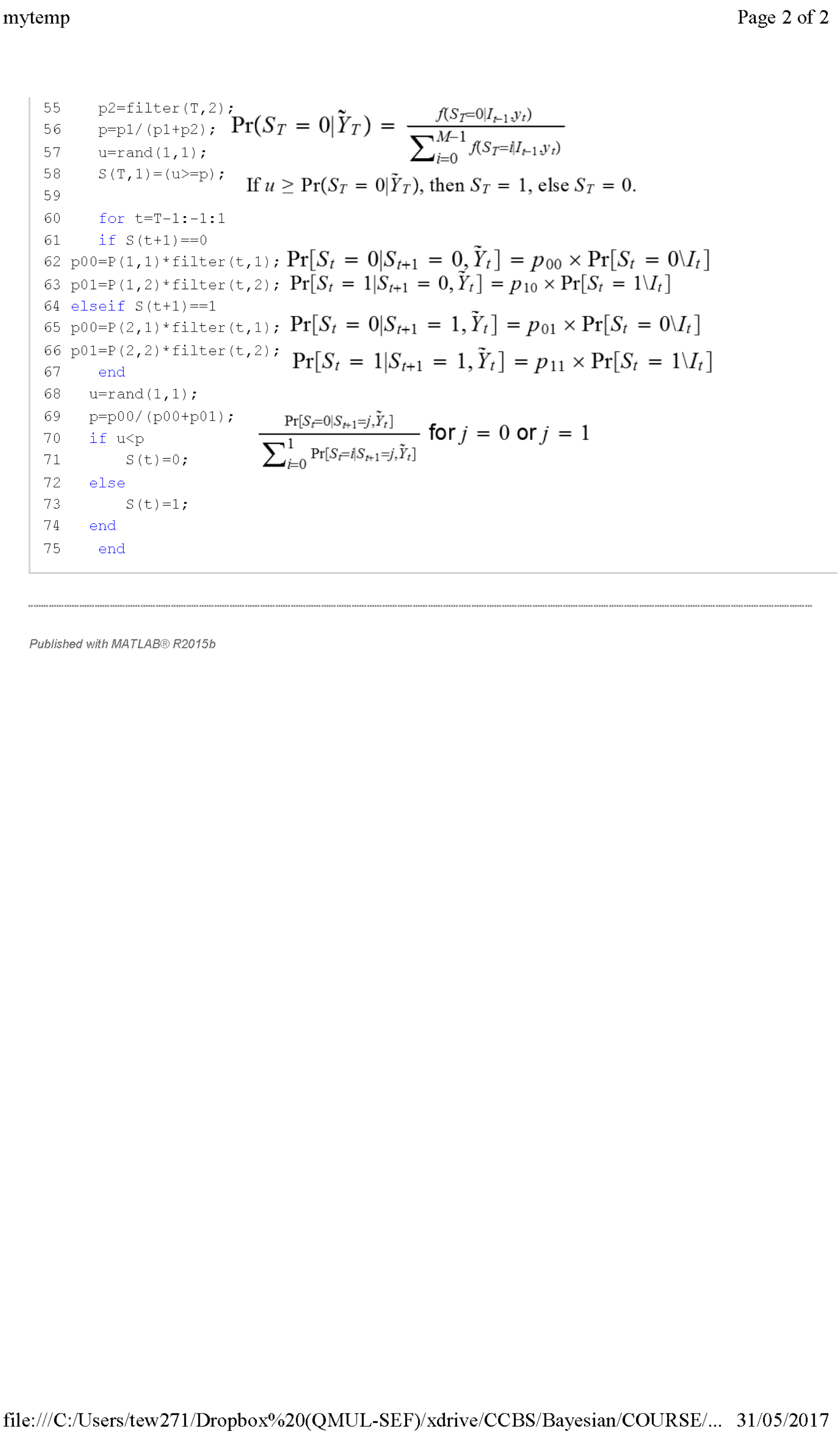
file:///C:/Users/tew271/Dropbox%20(QMUL-SEF)/xdrive/CCBS/Bayesian/COURSE/... 31/05/2017

FIGURE 2. Backward recursion in Matlab

obtain $f(y_t|S_t = i, I_{t-1}) \odot f(S_t = i|I_{t-1})$ (see equation 2.2). Line 44 sums this object across regimes and finally the updated estimates $\xi_{t|t}$ are obtained on line 45. Note that the output from line 45 is used as input in the RHS of the prediction equation (line 41) in the next time period. Also note that in this demonstration, the filter is run using the true values of the model parameters. This will change when we run the full Gibbs algorithm below.

5. The backward recursion to draw $\tilde{S}_t$ in Matlab

The code for the backward recursion to draw $\tilde{S}_t$ is shown in figures 2 and 3. Up to line 50, this example is identical to the one shown in figure 1 (see example2.m). Once the hamilton filter has been run and the probabilties $\xi_{t|t}$ saved



mytemp Page 2 of 2

```
55     p2=filter(T,2);
56     p=p1/(p1+p2);
57     u=rand(1,1);
58     S(T,1)=(u>=p);
59
60     for t=T-1:-1:1
61     if S(t+1)==0
62  p00=P(1,1)*filter(t,1);
63  p01=P(1,2)*filter(t,2);
64  elseif S(t+1)==1
65  p00=P(2,1)*filter(t,1);
66  p01=P(2,2)*filter(t,2);
67     end
68    u=rand(1,1);
69    p=p00/(p00+p01);
70    if u<p
71        S(t)=0;
72    else
73        S(t)=1;
74    end
75     end
```

$$\Pr(S_T = 0|\tilde{Y}_T) = \frac{f(S_T=0|I_{t-1},y_t)}{\sum_{i=0}^{M-1} f(S_T=i|I_{t-1},y_t)}$$

If $u \geq \Pr(S_T = 0|\tilde{Y}_T)$, then $S_T = 1$, else $S_T = 0$.

$$\Pr[S_t = 0|S_{t+1} = 0, \tilde{Y}_t] = p_{00} \times \Pr[S_t = 0\backslash I_t]$$

$$\Pr[S_t = 1|S_{t+1} = 0, \tilde{Y}_t] = p_{10} \times \Pr[S_t = 1\backslash I_t]$$

$$\Pr[S_t = 0|S_{t+1} = 1, \tilde{Y}_t] = p_{01} \times \Pr[S_t = 0\backslash I_t]$$

$$\Pr[S_t = 1|S_{t+1} = 1, \tilde{Y}_t] = p_{11} \times \Pr[S_t = 1\backslash I_t]$$

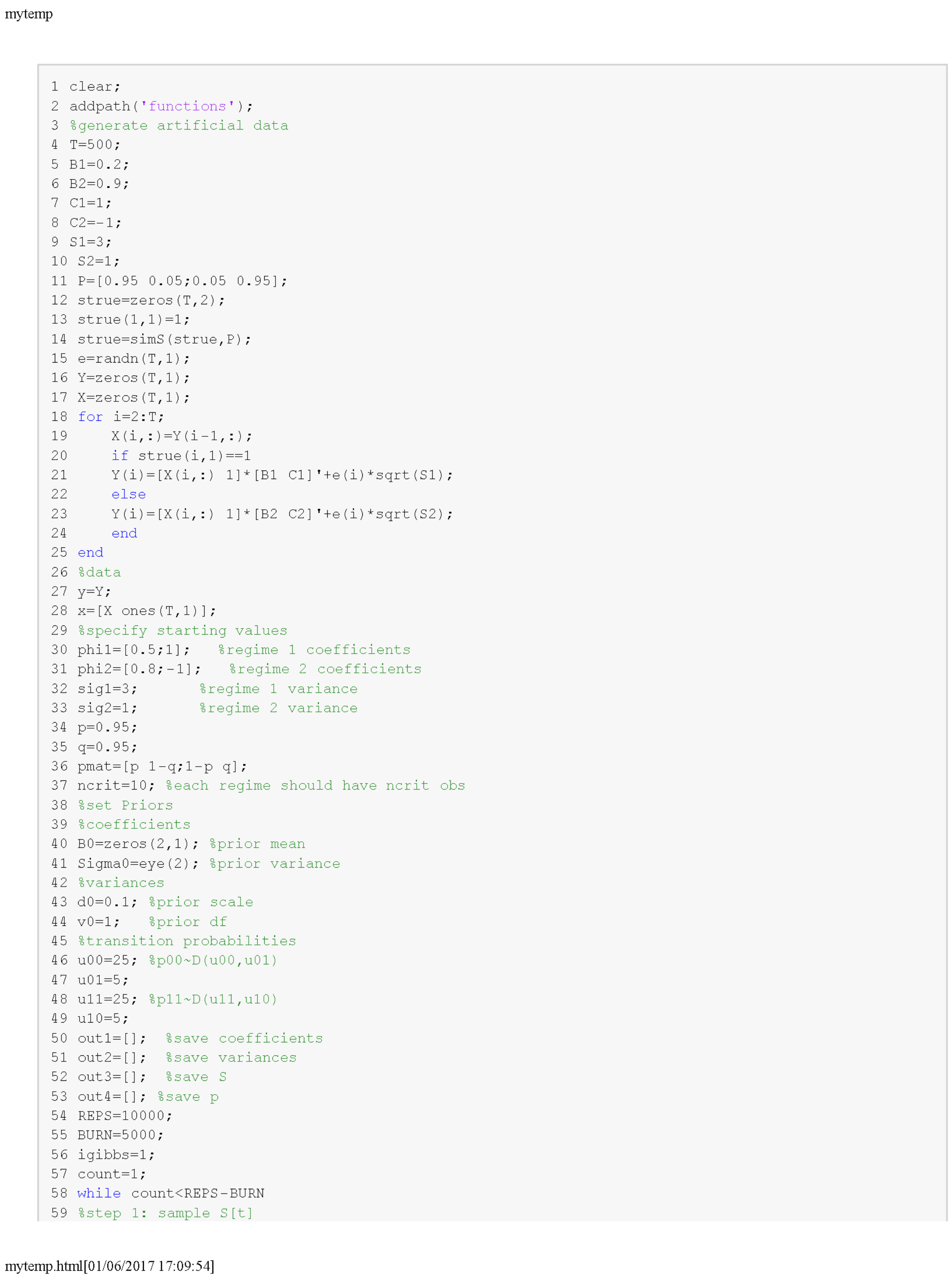
$$\frac{\Pr[S_t=0|S_{t+1}=j,\tilde{Y}_t]}{\sum_{i=0}^{1} \Pr[S_t=i|S_{t+1}=j,\tilde{Y}_t]} \text{ for } j = 0 \text{ or } j = 1$$

Published with MATLAB® R2015b

file:///C:/Users/tew271/Dropbox%20(QMUL-SEF)/xdrive/CCBS/Bayesian/COURSE/... 31/05/2017

FIGURE 3. Backward recursion in Matlab

in the matrix *filter* we are ready to proceed with the backward recursion. Lines 54 to 58 deal with time period T. Line 56 calculates $\Pr\left(S_T = 0|\tilde{Y}_T\right)$ and line 58 draws from a discrete distribution with probabilities $\Pr\left(S_T = 0|\tilde{Y}_T\right)$ and $\Pr\left(S_T = 1|\tilde{Y}_T\right)$. Line 60 begins the loop that begins in period T-1 and goes back to period 1. Lines 61 to 63 deal with the scenario when $S_{t+1} = 0$ and calculate $H\left(S_{t+1}|S_t\right) H\left(S_t|\tilde{Y}_t\right)$ (lines 62,63). Lines 64 to 67 carry out the same calculation when $S_{t+1} = 1$. Finally, S_t is drawn from this distribution on lines 69 to 73.



mytemp

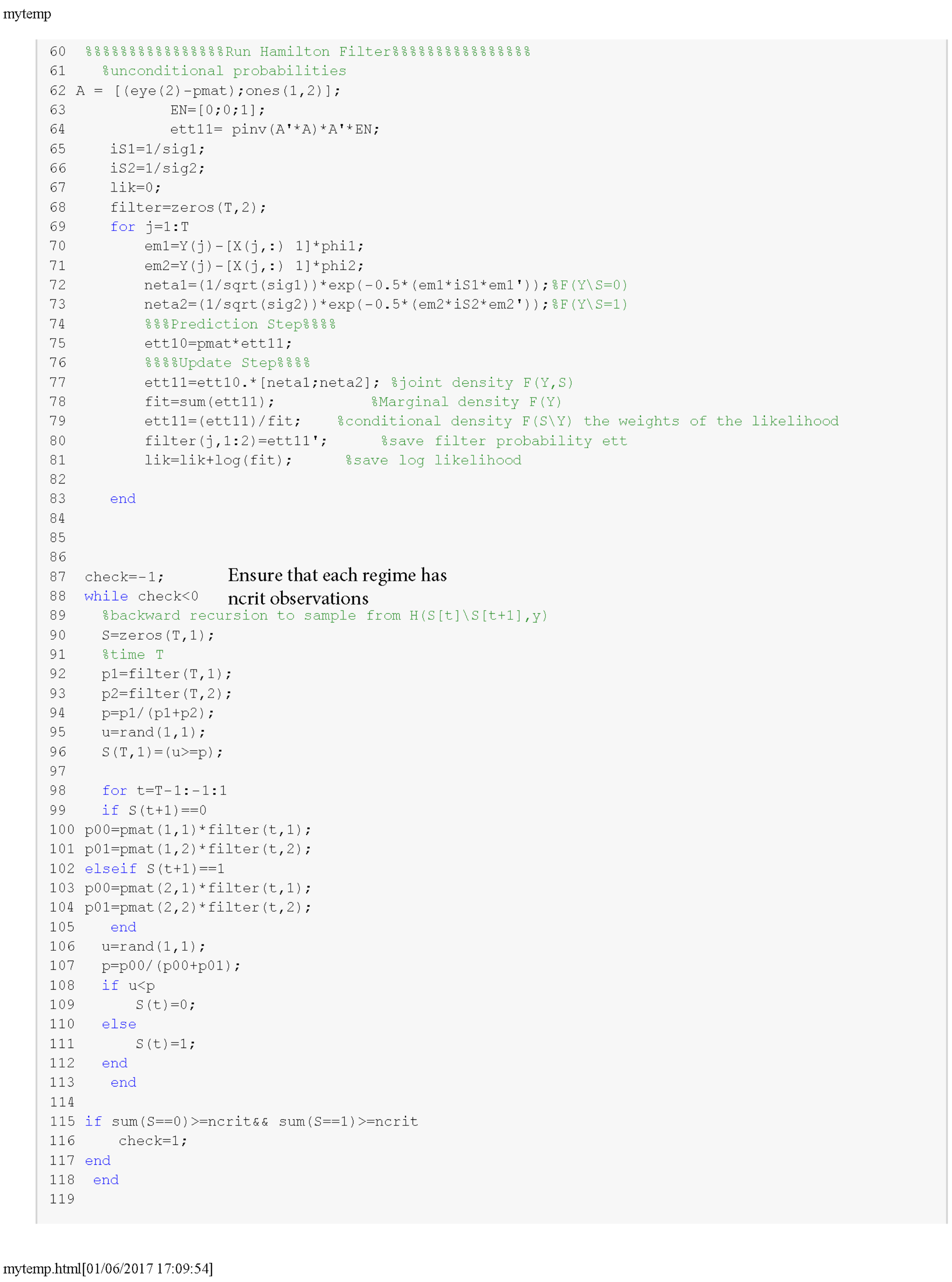
```
1 clear;
2 addpath('functions');
3 %generate artificial data
4 T=500;
5 B1=0.2;
6 B2=0.9;
7 C1=1;
8 C2=-1;
9 S1=3;
10 S2=1;
11 P=[0.95 0.05;0.05 0.95];
12 strue=zeros(T,2);
13 strue(1,1)=1;
14 strue=simS(strue,P);
15 e=randn(T,1);
16 Y=zeros(T,1);
17 X=zeros(T,1);
18 for i=2:T;
19     X(i,:)=Y(i-1,:);
20     if strue(i,1)==1
21     Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S1);
22     else
23     Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S2);
24     end
25 end
26 %data
27 y=Y;
28 x=[X ones(T,1)];
29 %specify starting values
30 phi1=[0.5;1];   %regime 1 coefficients
31 phi2=[0.8;-1];   %regime 2 coefficients
32 sig1=3;       %regime 1 variance
33 sig2=1;       %regime 2 variance
34 p=0.95;
35 q=0.95;
36 pmat=[p 1-q;1-p q];
37 ncrit=10; %each regime should have ncrit obs
38 %set Priors
39 %coefficients
40 B0=zeros(2,1); %prior mean
41 Sigma0=eye(2); %prior variance
42 %variances
43 d0=0.1; %prior scale
44 v0=1;   %prior df
45 %transition probabilities
46 u00=25; %p00~D(u00,u01)
47 u01=5;
48 u11=25; %p11~D(u11,u10)
49 u10=5;
50 out1=[];  %save coefficients
51 out2=[];  %save variances
52 out3=[];  %save S
53 out4=[]; %save p
54 REPS=10000;
55 BURN=5000;
56 igibbs=1;
57 count=1;
58 while count<REPS-BURN
59 %step 1: sample S[t]
```

mytemp.html[01/06/2017 17:09:54]

Figure 4. Gibbs sampler for an MS model.

6. Gibbs Sampler for the MS model in Matlab

We now describe the code for the full Gibbs algorithm to estimate the basic Markov switching model used in the two examples above (see equation 4.1) . The code for this model is shown in figures 4to 6. Lines 4 to 25 generate artificial data from the MS model. Lines 30 to 36 set starting values for the parameters. These might be obtained by first maximising the likelihood of the MS model and use these estimates to initialise the Gibbs sampler. Lines 40 to 44 set the priors for the coefficients and the error variances. The same prior is used in both regimes (normal for the coefficients, inverse Gamma for variances). Lines 46 to 49 set the dirichlet prior for p_{00} and p_{11}. The chosen values of the parameters, 25 and 5 imply a prior mean of 0.83 and variance of 0.16. Lines 59 to 118 implement the first step of the sampler. As described above, this involves running the Hamilton filter (lines 62 to 83) and a backward recursion to draw $\tilde{S}_t$ (lines 87 to 118). The while loop around these lines ensures that both regimes have at least



mytemp

```
60  %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
61    %unconditional probabilities
62 A = [(eye(2)-pmat);ones(1,2)];
63            EN=[0;0;1];
64            ett11= pinv(A'*A)*A'*EN;
65     iS1=1/sig1;
66     iS2=1/sig2;
67     lik=0;
68     filter=zeros(T,2);
69     for j=1:T
70         em1=Y(j)-[X(j,:) 1]*phi1;
71         em2=Y(j)-[X(j,:) 1]*phi2;
72         neta1=(1/sqrt(sig1))*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
73         neta2=(1/sqrt(sig2))*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
74         %%%Prediction Step%%%%
75         ett10=pmat*ett11;
76         %%%%Update Step%%%%
77         ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
78         fit=sum(ett11);           %Marginal density F(Y)
79         ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
80         filter(j,1:2)=ett11';      %save filter probability ett
81         lik=lik+log(fit);      %save log likelihood
82
83     end
84
85
86
87  check=-1;
88  while check<0
89    %backward recursion to sample from H(S[t]\S[t+1],y)
90    S=zeros(T,1);
91    %time T
92    p1=filter(T,1);
93    p2=filter(T,2);
94    p=p1/(p1+p2);
95    u=rand(1,1);
96    S(T,1)=(u>=p);
97
98    for t=T-1:-1:1
99    if S(t+1)==0
100 p00=pmat(1,1)*filter(t,1);
101 p01=pmat(1,2)*filter(t,2);
102 elseif S(t+1)==1
103 p00=pmat(2,1)*filter(t,1);
104 p01=pmat(2,2)*filter(t,2);
105     end
106    u=rand(1,1);
107    p=p00/(p00+p01);
108    if u<p
109        S(t)=0;
110    else
111        S(t)=1;
112    end
113     end
114
115 if sum(S==0)>=ncrit&& sum(S==1)>=ncrit
116     check=1;
117 end
118  end
119
```

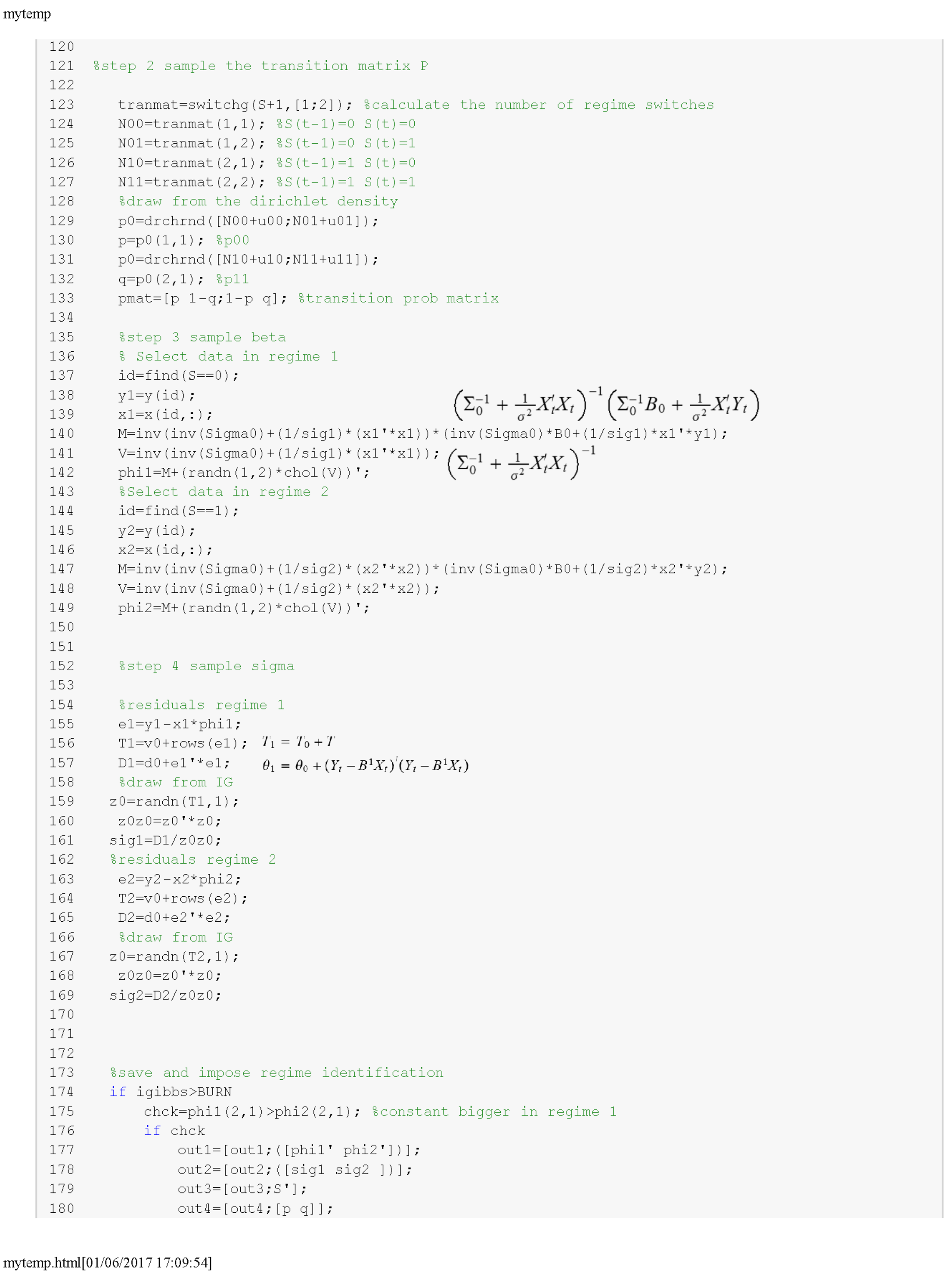
Ensure that each regime has ncrit observations

mytemp.html[01/06/2017 17:09:54]

Figure 5. Gibbs Sampler for an MS model

ncrit=10 observations. Lines 123 to 133 draw the transition probabilties. The function switchg requires as input, $\tilde{S}_t$ and the two values taken by this variable. It returns a 2×2 matrix $\begin{pmatrix} \eta_{00} & \eta_{01} \\ \eta_{10} & \eta_{11} \end{pmatrix}$ where η_{ji} refers to the number of times regime j is followed by regime i. The function drchrnd produces a draw from the dirichlet distribution using algorithm 4. Given $\tilde{S}_t$, the remaining steps are straightforward: The sample is split into observations where $\tilde{S}_t = 0$ and where $\tilde{S}_t = 1$. The regression coefficients and error variances are drawn from the normal and inverse Gamma conditional posteriors separately in each of the sub-samples.

Figure 7 shows the estimated marginal posterior distributions using 10,000 iterations and a burn-in of 5000. The true values are shown as vertical lines. This run of the sampler approximates the true values fairly well. As shown in the bottom panel, the estimate of $\Pr[S_t = 1]$ obtained as $\frac{1}{5000}\sum_{j=1}^{5000} I\left[\tilde{S}_t^{(j)} = 1\right]$ where $I\left[\tilde{S}_t^{(j)} = 1\right]$ is a dummy variable that equals 1 if $\tilde{S}_t = 1$ and 0 otherwise for the jth draw tracks the realisation of this regime closely.



mytemp

```

%step 2 sample the transition matrix P

   tranmat=switchg(S+1,[1;2]); %calculate the number of regime switches
   N00=tranmat(1,1); %S(t-1)=0 S(t)=0
   N01=tranmat(1,2); %S(t-1)=0 S(t)=1
   N10=tranmat(2,1); %S(t-1)=1 S(t)=0
   N11=tranmat(2,2); %S(t-1)=1 S(t)=1
   %draw from the dirichlet density
   p0=drchrnd([N00+u00;N01+u01]);
   p=p0(1,1); %p00
   p0=drchrnd([N10+u10;N11+u11]);
   q=p0(2,1); %p11
   pmat=[p 1-q;1-p q]; %transition prob matrix

   %step 3 sample beta
   % Select data in regime 1
   id=find(S==0);
   y1=y(id);
   x1=x(id,:);
```

$$\left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}\left(\Sigma_0^{-1}B_0+\frac{1}{\sigma^2}X_t'Y_t\right)$$

```
   M=inv(inv(Sigma0)+(1/sig1)*(x1'*x1))*(inv(Sigma0)*B0+(1/sig1)*x1'*y1);
   V=inv(inv(Sigma0)+(1/sig1)*(x1'*x1));
```

$$\left(\Sigma_0^{-1}+\frac{1}{\sigma^2}X_t'X_t\right)^{-1}$$

```
   phi1=M+(randn(1,2)*chol(V))';
   %Select data in regime 2
   id=find(S==1);
   y2=y(id);
   x2=x(id,:);
   M=inv(inv(Sigma0)+(1/sig2)*(x2'*x2))*(inv(Sigma0)*B0+(1/sig2)*x2'*y2);
   V=inv(inv(Sigma0)+(1/sig2)*(x2'*x2));
   phi2=M+(randn(1,2)*chol(V))';


   %step 4 sample sigma

   %residuals regime 1
   e1=y1-x1*phi1;
   T1=v0+rows(e1);
```

$$T_1 = T_0 + T$$

```
   D1=d0+e1'*e1;
```

$$\theta_1 = \theta_0 + (Y_t - B^1X_t)'(Y_t - B^1X_t)$$

```
   %draw from IG
  z0=randn(T1,1);
   z0z0=z0'*z0;
  sig1=D1/z0z0;
  %residuals regime 2
   e2=y2-x2*phi2;
   T2=v0+rows(e2);
   D2=d0+e2'*e2;
   %draw from IG
  z0=randn(T2,1);
   z0z0=z0'*z0;
  sig2=D2/z0z0;



  %save and impose regime identification
  if igibbs>BURN
      chck=phi1(2,1)>phi2(2,1); %constant bigger in regime 1
      if chck
          out1=[out1;([phi1' phi2'])];
          out2=[out2;([sig1 sig2 ])];
          out3=[out3;S'];
          out4=[out4;[p q]];
```

mytemp.html[01/06/2017 17:09:54]

FIGURE 6. Gibbs sampler for an MS model

7. Extensions

In this section, we consider several extensions of the basic MS model described above.

7.1. Markov switching VAR. We consider the following MSVAR model

$$\begin{aligned} Y_t &= c_{s_t} + \sum_{p=1}^{P} B_{S_t} Y_{t-p} + v_t \\ & v_t \tilde{} N(0, \Omega_{S_t}) \end{aligned}$$

where S_t follows a first order Markov chain with M regimes and transition matrix P. Only minor changes are required in the Gibbs sampling algorithm described above:

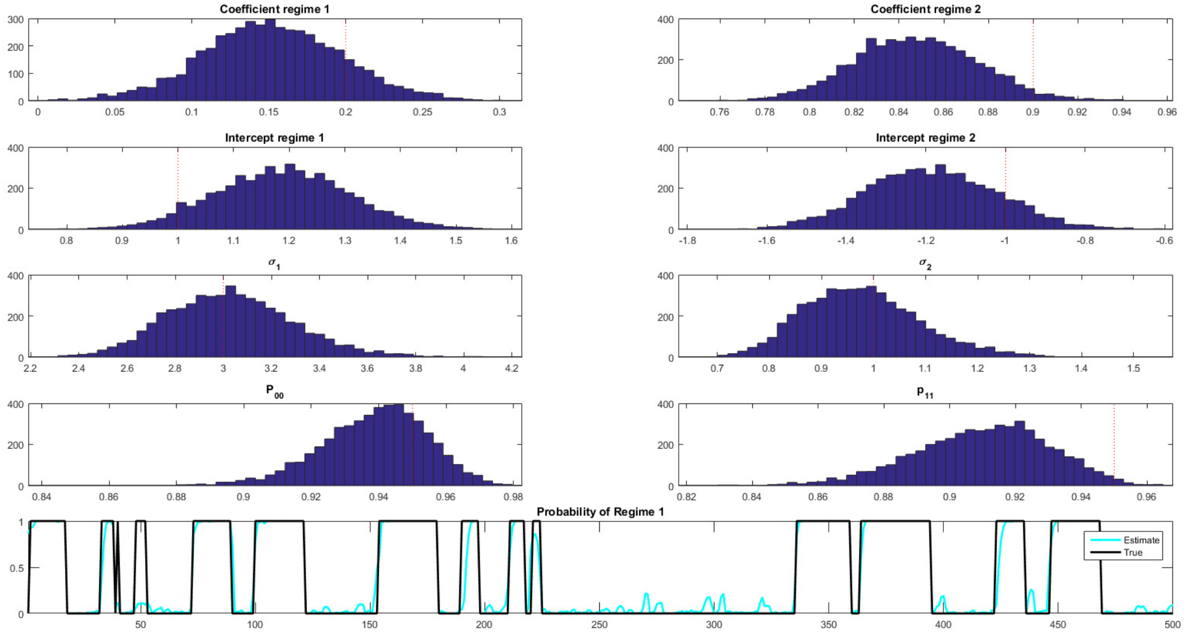


Figure 7. Output from 10,000 iterations of the Gibbs Sampler for the MS model

(1) Conditional posterior of $\tilde{S}_t$. When running the Hamilton filter, the conditional likelihood for observation t changes to $f\left(y_t|S_t = i, I_{t-1}\right) = 2\pi^{-0.5}\det\left(\Omega_{S_t}^{-1}\right)^{0.5}\exp\left(-0.5\left(Y_t - X_t b_{S_t}\right)'\Omega_{S_t}^{-1}\left(Y_t - X_t b_{S_t}\right)\right)$ where X denotes the lags and the intercept while b is the matrix of coefficients including B and the intercepts c in one $NP+1\times N$ matrix (see equation 2.2). There is no change in the backward recursion or the draw of the transition probabilties.

(2) Conditional posterior of the VAR parameters. Given $\tilde{S}_t$, the VAR parameters are drawn in each regime by simply using the conditional posterior distributions discussed in Chapter 2. In particular, given a Normal prior for the VAR coefficients $p(b_{S_t})\tilde{}N\left(\tilde{b}_0, H\right)$ and a inverse Wishart prior for the error covariance $p(\Omega_{S_t})\tilde{}IW\left(\bar{S},\alpha\right)$ the conditional posterior in each regime is Normal for the VAR coefficients $H\left(b_{S_t}|\Omega_{S_t}, P, S_t, Y_t\right)\tilde{}N\left(M^*_{S_t}, V^*_{S_t}\right)$ and inverse Wishart for $\Omega_{S_t}: H\left(\Omega_{S_t}|b_{S_t}, P, S_t, Y_t\right)\tilde{}IW\left(\bar{\Omega}_{S_t}, T+\alpha\right)$. Note that:

$$\begin{aligned} M^*_{S_t} &= \left(H^{-1} + \Omega_{S_t}^{-1}\otimes X'_{S_t}X_{S_t}\right)^{-1}\left(H^{-1}\tilde{b}_0 + \Omega_{S_t}^{-1}\otimes X'_{S_t}X_{S_t}\hat{b}_{S_t}\right) \\ V^*_{S_t} &= \left(H^{-1} + \Omega_{S_t}^{-1}\otimes X'_{S_t}X_{S_t}\right)^{-1} \end{aligned} \tag{7.1}$$

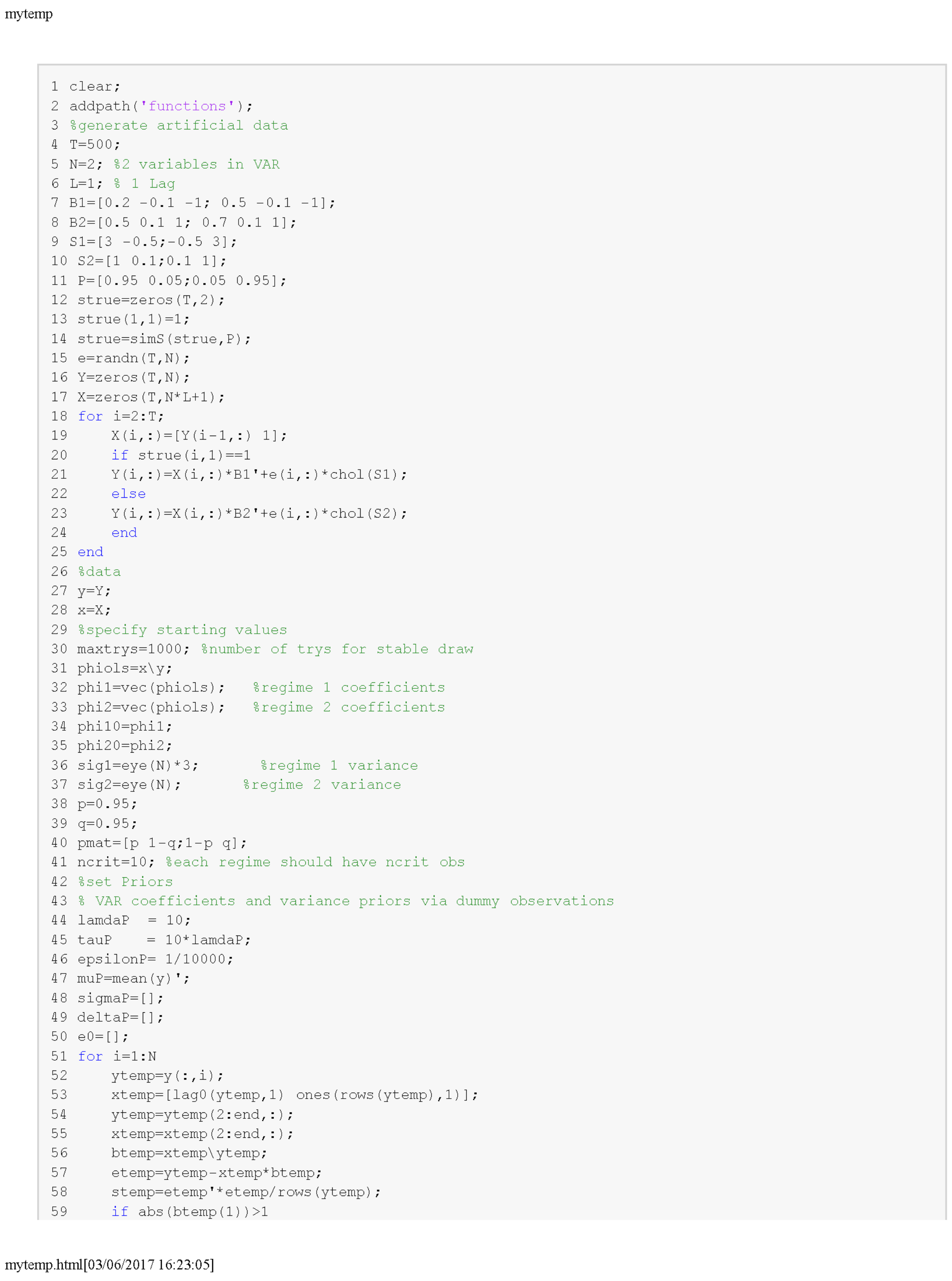
where $\hat{b}_{S_t}$ is the OLS estimate in regime $S_t = i$ and X_{S_t} denotes X selected when $\tilde{S}_t = i$. In addition $\bar{\Omega}_{S_t} = \left(Y_{S_t} - X_{S_t}\hat{b}_{S_t}\right)'\left(Y_{S_t} - X_{S_t}\hat{b}_{S_t}\right) + \bar{S}$. Note that this draw can also be implemented if the priors on the VAR parameters are implemented via dummy observations. As discussed in Chapter 2, the conditional posteriors are simpler in this case.

The code for this algorithm is shown in figures 8 to 11. First note that lines 95 and 96 uses the multivariate normal density as mentioned above. The second change is on line 160 onwards. Once the sample is divided into the two regimes, the VAR parameters are drawn separately. Note that in this code, the prior on the VAR parameters is implemented via dummy observations. The conditional posteriors have a simpler form in this case (see Chapter 2).

7.2. AR model with switching mean and variance. Consider the following two regime MS model

$$y_t - \mu_{S_t^*} = \rho\left(y_{t-1} - \mu_{S_{t-1}^*}\right) + e_t, e_t\tilde{}N(0, \sigma^2_{S_t}) \tag{7.2}$$

where $S_t^* = 0, 1$ denotes the state variable with transition probability matrix $P^* = \begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{pmatrix}$. Note that unlike the simple MS model considered above, a lag of the state variable S_{t-1}^* appears on the RHS. Therefore, in order to calculate the likelihood function using the procedure described in equation 1.3, one needs to be able to track both S_t^* and S_{t-1}^*. As described in Hamilton (1994), this easily achieved by defining a new state variable S_t that takes on



mytemp

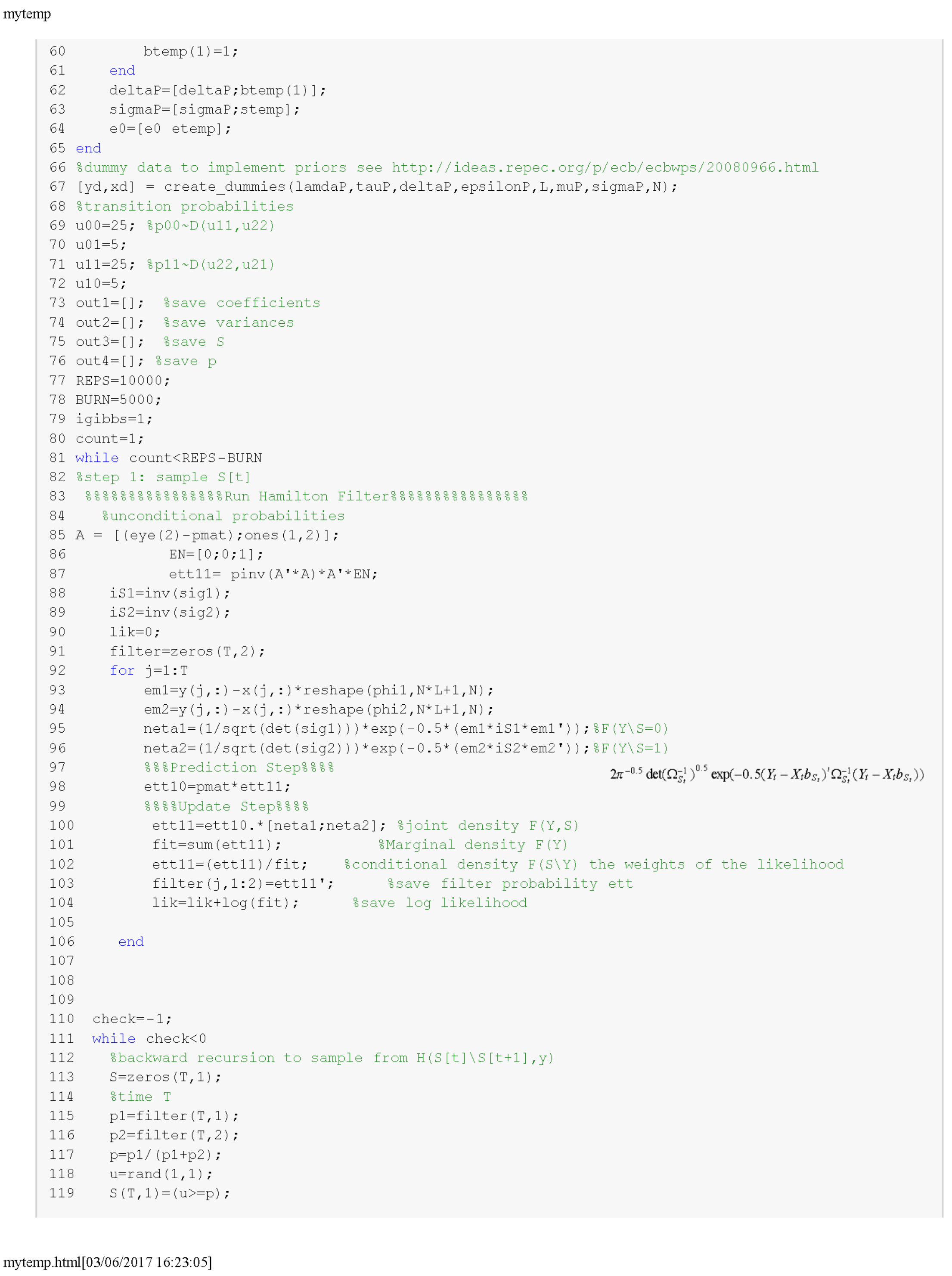
```
clear;
addpath('functions');
%generate artificial data
T=500;
N=2; %2 variables in VAR
L=1; % 1 Lag
B1=[0.2 -0.1 -1; 0.5 -0.1 -1];
B2=[0.5 0.1 1; 0.7 0.1 1];
S1=[3 -0.5;-0.5 3];
S2=[1 0.1;0.1 1];
P=[0.95 0.05;0.05 0.95];
strue=zeros(T,2);
strue(1,1)=1;
strue=simS(strue,P);
e=randn(T,N);
Y=zeros(T,N);
X=zeros(T,N*L+1);
for i=2:T;
    X(i,:)=[Y(i-1,:) 1];
    if strue(i,1)==1
    Y(i,:)=X(i,:)*B1'+e(i,:)*chol(S1);
    else
    Y(i,:)=X(i,:)*B2'+e(i,:)*chol(S2);
    end
end
%data
y=Y;
x=X;
%specify starting values
maxtrys=1000; %number of trys for stable draw
phiols=x\y;
phi1=vec(phiols);   %regime 1 coefficients
phi2=vec(phiols);   %regime 2 coefficients
phi10=phi1;
phi20=phi2;
sig1=eye(N)*3;       %regime 1 variance
sig2=eye(N);       %regime 2 variance
p=0.95;
q=0.95;
pmat=[p 1-q;1-p q];
ncrit=10; %each regime should have ncrit obs
%set Priors
% VAR coefficients and variance priors via dummy observations
lamdaP  = 10;
tauP    = 10*lamdaP;
epsilonP= 1/10000;
muP=mean(y)';
sigmaP=[];
deltaP=[];
e0=[];
for i=1:N
    ytemp=y(:,i);
    xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
    ytemp=ytemp(2:end,:);
    xtemp=xtemp(2:end,:);
    btemp=xtemp\ytemp;
    etemp=ytemp-xtemp*btemp;
    stemp=etemp'*etemp/rows(ytemp);
    if abs(btemp(1))>1
```

mytemp.html[03/06/2017 16:23:05]

Figure 8. Gibbs Sampler for a Markov Switching VAR

values $1, 2, 3, 4$

$$
\begin{aligned}
S_t &= 1 \text{ if } S_t^* = 0 \text{ and } S_{t-1}^* = 0 \\
S_t &= 2 \text{ if } S_t^* = 1 \text{ and } S_{t-1}^* = 0 \\
S_t &= 3 \text{ if } S_t^* = 0 \text{ and } S_{t-1}^* = 1 \\
S_t &= 4 \text{ if } S_t^* = 1 \text{ and } S_{t-1}^* = 1
\end{aligned}
$$



mytemp

```
        btemp(1)=1;
    end
    deltaP=[deltaP;btemp(1)];
    sigmaP=[sigmaP;stemp];
    e0=[e0 etemp];
end
%dummy data to implement priors see http://ideas.repec.org/p/ecb/ecbwps/20080966.html
[yd,xd] = create_dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
%transition probabilities
u00=25; %p00~D(u11,u22)
u01=5;
u11=25; %p11~D(u22,u21)
u10=5;
out1=[];  %save coefficients
out2=[];  %save variances
out3=[];  %save S
out4=[]; %save p
REPS=10000;
BURN=5000;
igibbs=1;
count=1;
while count<REPS-BURN
%step 1: sample S[t]
 %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
   %unconditional probabilities
A = [(eye(2)-pmat);ones(1,2)];
           EN=[0;0;1];
           ett11= pinv(A'*A)*A'*EN;
    iS1=inv(sig1);
    iS2=inv(sig2);
    lik=0;
    filter=zeros(T,2);
    for j=1:T
        em1=y(j,:)-x(j,:)*reshape(phi1,N*L+1,N);
        em2=y(j,:)-x(j,:)*reshape(phi2,N*L+1,N);
        neta1=(1/sqrt(det(sig1)))*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
        neta2=(1/sqrt(det(sig2)))*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
        %%%Prediction Step%%%%
        ett10=pmat*ett11;
        %%%%Update Step%%%%
        ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
        fit=sum(ett11);           %Marginal density F(Y)
        ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
        filter(j,1:2)=ett11';      %save filter probability ett
        lik=lik+log(fit);      %save log likelihood

    end



 check=-1;
 while check<0
   %backward recursion to sample from H(S[t]\S[t+1],y)
   S=zeros(T,1);
   %time T
   p1=filter(T,1);
   p2=filter(T,2);
   p=p1/(p1+p2);
   u=rand(1,1);
   S(T,1)=(u>=p);
```

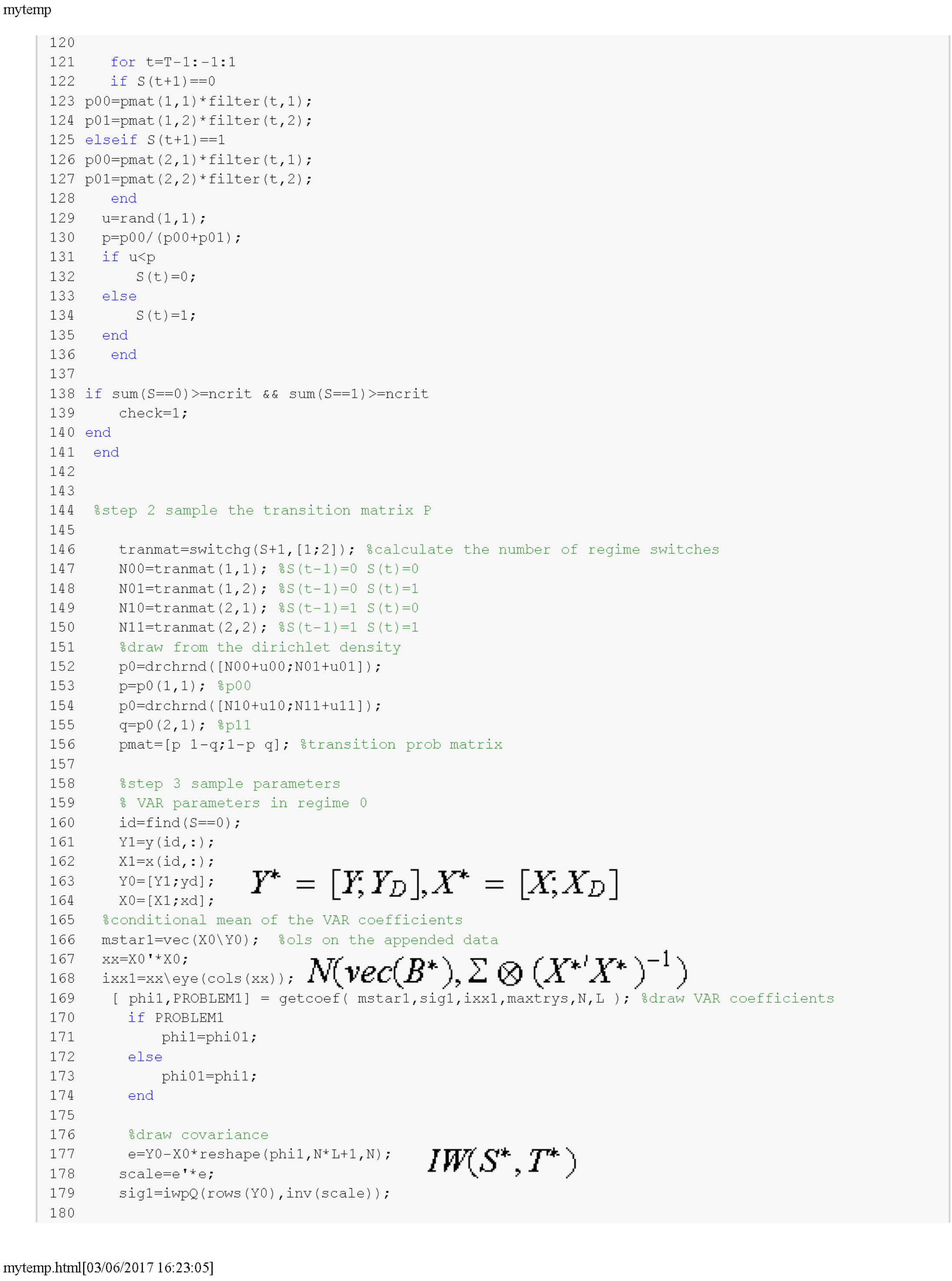
$2\pi^{-0.5}\det(\Omega_{S_t}^{-1})^{0.5}\exp(-0.5(Y_t - X_t b_{S_t})'\Omega_{S_t}^{-1}(Y_t - X_t b_{S_t}))$

mytemp.html[03/06/2017 16:23:05]

FIGURE 9. Gibbs Sampler for a Markov Switching VAR

The transition probability matrix for the new state variable is given as:

$$
\begin{aligned}
P &= \begin{pmatrix} p_{11} & p_{21} & p_{31} & p_{41} \\ p_{12} & p_{22} & p_{32} & p_{42} \\ p_{13} & p_{23} & p_{33} & p_{43} \\ p_{14} & p_{24} & p_{34} & p_{44} \end{pmatrix} \\
&= \begin{pmatrix} p_{00} & 0 & p_{00} & 0 \\ p_{01} & 0 & p_{01} & 0 \\ 0 & p_{10} & 0 & p_{10} \\ 0 & p_{11} & 0 & p_{11} \end{pmatrix}
\end{aligned}
\tag{7.3}
$$



mytemp

```
   for t=T-1:-1:1
   if S(t+1)==0
p00=pmat(1,1)*filter(t,1);
p01=pmat(1,2)*filter(t,2);
elseif S(t+1)==1
p00=pmat(2,1)*filter(t,1);
p01=pmat(2,2)*filter(t,2);
   end
  u=rand(1,1);
  p=p00/(p00+p01);
  if u<p
      S(t)=0;
  else
      S(t)=1;
  end
   end

if sum(S==0)>=ncrit && sum(S==1)>=ncrit
    check=1;
end
 end


 %step 2 sample the transition matrix P

    tranmat=switchg(S+1,[1;2]); %calculate the number of regime switches
    N00=tranmat(1,1); %S(t-1)=0 S(t)=0
    N01=tranmat(1,2); %S(t-1)=0 S(t)=1
    N10=tranmat(2,1); %S(t-1)=1 S(t)=0
    N11=tranmat(2,2); %S(t-1)=1 S(t)=1
    %draw from the dirichlet density
    p0=drchrnd([N00+u00;N01+u01]);
    p=p0(1,1); %p00
    p0=drchrnd([N10+u10;N11+u11]);
    q=p0(2,1); %p11
    pmat=[p 1-q;1-p q]; %transition prob matrix

    %step 3 sample parameters
    % VAR parameters in regime 0
    id=find(S==0);
    Y1=y(id,:);
    X1=x(id,:);
    Y0=[Y1;yd];
    X0=[X1;xd];
  %conditional mean of the VAR coefficients
  mstar1=vec(X0\Y0);  %ols on the appended data
  xx=X0'*X0;
  ixx1=xx\eye(cols(xx));
   [ phi1,PROBLEM1] = getcoef( mstar1,sig1,ixx1,maxtrys,N,L ); %draw VAR coefficients
     if PROBLEM1
         phi1=phi01;
     else
         phi01=phi1;
     end

     %draw covariance
     e=Y0-X0*reshape(phi1,N*L+1,N);
    scale=e'*e;
    sig1=iwpQ(rows(Y0),inv(scale));

```

$Y^{*} = [Y; Y_D], X^{*} = [X; X_D]$
$N(vec(B^{*}), \Sigma \otimes (X^{*\prime}X^{*})^{-1})$
$IW(S^{*}, T^{*})$

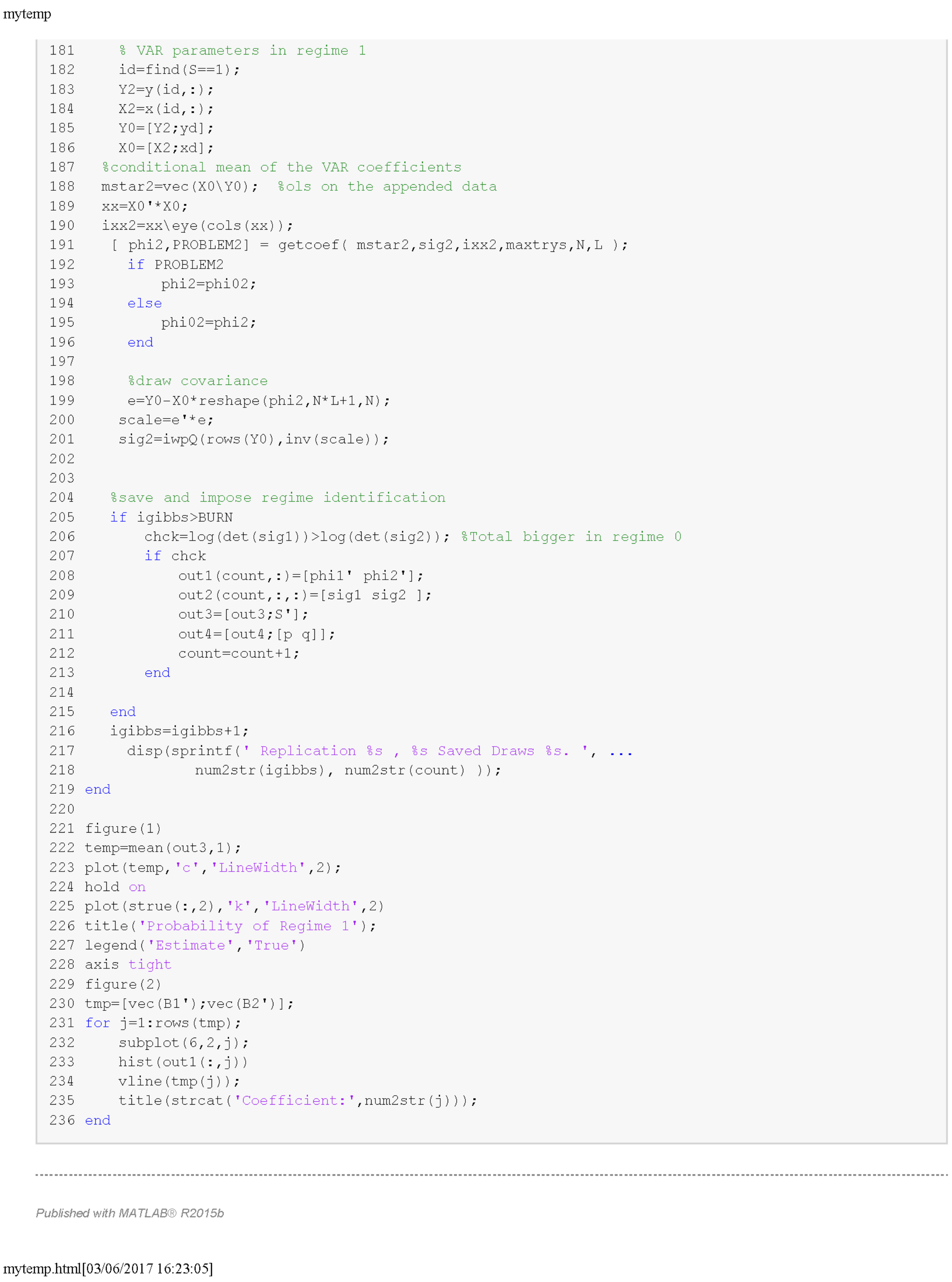
mytemp.html[03/06/2017 16:23:05]

FIGURE 10. Gibbs Sampler for a Markov Switching VAR

This matrix implies, for example that $\Pr[S_t = 1|S_{t-1} = 2] = 0$. Given that when $S_t = 2$ the regime is defined as $S_t^* = 1$ and $S_{t-1}^* = 0$ and when $S_t = 1$ the regime is defined as $S_t^* = 0$ and $S_{t-1}^* = 0$, regime 2 cannot be followed by regime 1 as it would imply a contradiction. We describe the Gibbs algorithm for this 4 regime model below. Note that as the lags in this model increase, the number of (artificial) regimes increase. Similarly, if S_t^* follows $M > 2$ regimes, then the re-parameterised model is more complex. While estimation becomes more computationally intensive, the algorithm described below still applies.

Gibbs sampling algorithm. The algorithm involves sampling from the following conditional posterior distributions:

(1) Sample from $H\left(S_t|\mu_{S_t}, \sigma^2_{S_t}, P, \rho\right)$. As before, this involves the following two steps:



mytemp

```
    % VAR parameters in regime 1
    id=find(S==1);
    Y2=y(id,:);
    X2=x(id,:);
    Y0=[Y2;yd];
    X0=[X2;xd];
  %conditional mean of the VAR coefficients
  mstar2=vec(X0\Y0);  %ols on the appended data
  xx=X0'*X0;
  ixx2=xx\eye(cols(xx));
   [ phi2,PROBLEM2] = getcoef( mstar2,sig2,ixx2,maxtrys,N,L );
     if PROBLEM2
         phi2=phi02;
     else
         phi02=phi2;
     end

     %draw covariance
     e=Y0-X0*reshape(phi2,N*L+1,N);
    scale=e'*e;
    sig2=iwpQ(rows(Y0),inv(scale));


   %save and impose regime identification
   if igibbs>BURN
       chck=log(det(sig1))>log(det(sig2)); %Total bigger in regime 0
       if chck
           out1(count,:)=[phi1' phi2'];
           out2(count,:,:)=[sig1 sig2 ];
           out3=[out3;S'];
           out4=[out4;[p q]];
           count=count+1;
       end

   end
   igibbs=igibbs+1;
     disp(sprintf(' Replication %s , %s Saved Draws %s. ', ...
             num2str(igibbs), num2str(count) ));
end

figure(1)
temp=mean(out3,1);
plot(temp,'c','LineWidth',2);
hold on
plot(strue(:,2),'k','LineWidth',2)
title('Probability of Regime 1');
legend('Estimate','True')
axis tight
figure(2)
tmp=[vec(B1');vec(B2')];
for j=1:rows(tmp);
    subplot(6,2,j);
    hist(out1(:,j))
    vline(tmp(j));
    title(strcat('Coefficient:',num2str(j)));
end
```

Published with MATLAB® R2015b

mytemp.html[03/06/2017 16:23:05]

Figure 11. Gibbs Sampler for a Markov Switching VAR

(a) Run the Hamilton filter to obtain $\xi_{t|t}$. Note that the conditional likelihoods $f(y_t|S_t)$ in the update step are given by:

$$f(y_t|S_t = 1) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-((y_t-\mu_0)-(y_{t-1}-\mu_0)\rho)'((y_t-\mu_0)-(y_{t-1}-\mu_0)\rho)'}{2\sigma_0^2}\right)$$
$$f(y_t|S_t = 2) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-((y_t-\mu_1)-(y_{t-1}-\mu_0)\rho)'((y_t-\mu_1)-(y_{t-1}-\mu_0)\rho)'}{2\sigma_1^2}\right)$$
$$f(y_t|S_t = 3) = \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-((y_t-\mu_0)-(y_{t-1}-\mu_1)\rho)'((y_t-\mu_0)-(y_{t-1}-\mu_1)\rho)'}{2\sigma_0^2}\right)$$
$$f(y_t|S_t = 4) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-((y_t-\mu_1)-(y_{t-1}-\mu_1)\rho)'((y_t-\mu_1)-(y_{t-1}-\mu_1)\rho)'}{2\sigma_1^2}\right)$$

Apart from this change, the remaining steps of the filter are unchanged.

(b) Draw from $H\left(\tilde{S}_t|\sigma^2_{S_t}, b_{S_t}, \rho, P, \tilde{Y}_t\right) = H\left(S_T|\tilde{Y}_T\right)\prod_{t=1}^{T-1} H\left(S_t|S_{t+1}, \tilde{Y}_t\right)$

(i) At time T, one can draw S_T from the discrete distribution $H\left(S_T|\tilde{Y}_T\right)$ using $\xi_{T|T}$ as the probability associated with each value S_T takes. In the four regime case, one calculates $\Pr\left(S_T = 1|\tilde{Y}_T\right) = \frac{f(S_T=1|I_{t-1},y_t)}{\sum_{i=1}^4 f(S_T=i|I_{t-1},y_t)}$ and draws $u\tilde{\ }U(0,1)$. If $u \leq \Pr\left(S_T = 1|\tilde{Y}_T\right)$, then $S_T = 1$. Otherwise calculate $\Pr\left(S_T = 2|\tilde{Y}_T\right) = \frac{f(S_T=2|I_{t-1},y_t)}{\sum_{i=2}^4 f(S_T=i|I_{t-1},y_t)}$ and draw $u\tilde{\ }U(0,1)$. If $u \leq \Pr\left(S_T = 2|\tilde{Y}_T\right)$, then $S_T = 2$. Otherwise calculate $\Pr\left(S_T = 3|\tilde{Y}_T\right) = \frac{f(S_T=3|I_{t-1},y_t)}{\sum_{i=3}^4 f(S_T=i|I_{t-1},y_t)}$ and draw $u\tilde{\ }U(0,1)$. If $u \leq \Pr\left(S_T = 3|\tilde{Y}_T\right)$, then $S_T = 3$ else $S_T = 4$.

(ii) For $t = T-1, T-2, ..., 1$ draw from $H\left(S_{t+1}|S_t\right) H\left(S_t|\tilde{Y}_t\right) \propto H\left(S_{t+1}|S_t\right) H\left(S_t|\tilde{Y}_t\right)$. As before this is carried by the following procedure.
If $S_{t+1} = 1$, then one calculates $\Pr\left[S_t = 1|S_{t+1} = 1, \tilde{Y}_t\right], \Pr\left[S_t = 2|S_{t+1} = 1, \tilde{Y}_t\right], \Pr\left[S_t = 3|S_{t+1} = 1, \tilde{Y}_t\right]$ and $\Pr\left[S_t = 4|S_{t+1} = 1, \tilde{Y}_t\right]$.
For e.g. $\Pr\left[S_t = 1|S_{t+1} = 1, \tilde{Y}_t\right] = p_{11} \times \Pr\left[S_t = 1|I_t\right]$ and $\Pr\left[S_t = 4|S_{t+1} = 1, \tilde{Y}_t\right] = p_{41} \times \Pr\left[S_t = 1|I_t\right]$.
With these probabilities in hand the draw of S_t from this discrete distribution is exactly as in step (i) above. If $S_{t+1} = 2, 3$ or 4, the same process is repeated with probabilties $\Pr\left[S_t = 1|S_{t+1} = j, \tilde{Y}_t\right], \Pr\left[S_t = 2|S_{t+1} = j, \tilde{Y}_t\right], \Pr\left[S_t = 3|S_{t+1} = j, \tilde{Y}_t\right]$ and $\Pr\left[S_t = 4|S_{t+1} = j, \tilde{Y}_t\right]$ for j=2,3,4. With the draw of $\tilde{S}_t$ in hand, the original state variable $\tilde{S}^*_t = [S^*_1, S^*_1, ..., S^*_T]$ can be constructed as $1 - \left(I\left[\tilde{S}_t = 1\right] + I\left[\tilde{S}_t = 3\right]\right)$ where $I[.]$ is an indicator function that equals 1 if the argument is true. Note that by carrying out the addition $I\left[\tilde{S}_t = 1\right] + I\left[\tilde{S}_t = 3\right]$ we are 'integrating' over the two possible values for $\tilde{S}^*_{t-1}$ given that $\tilde{S}^*_t = 0$. Calculating $1 - \left(I\left[\tilde{S}_t = 1\right] + I\left[\tilde{S}_t = 3\right]\right)$ ensures that the first regime has the label 0.

(2) Sample from $H\left(P|S_t, \mu_{S_t}, \sigma^2_{S_t}, \rho\right)$. Given $\tilde{S}^*_t$, the draw for the elements of P^* is as described in section 3.1. Then the matrix P can then be constructed easily using equation 7.3.

(3) Sample from $H\left(\rho|S_t, \mu_{S_t}, \sigma^2_{S_t}, P\right)$. Define the following time-varying parameters

$$\begin{aligned}
\mu_t &= I\left[\tilde{S}_t = 1\right]\mu_0 + I\left[\tilde{S}_t = 2\right]\mu_1 + I\left[\tilde{S}_t = 3\right]\mu_0 + I\left[\tilde{S}_t = 4\right]\mu_1 \\
\mu_{t-1} &= I\left[\tilde{S}_t = 1\right]\mu_0 + I\left[\tilde{S}_t = 2\right]\mu_0 + I\left[\tilde{S}_t = 3\right]\mu_1 + I\left[\tilde{S}_t = 4\right]\mu_1 \\
\sigma_t &= I\left[\tilde{S}_t = 1\right]\sigma_0 + I\left[\tilde{S}_t = 2\right]\sigma_1 + I\left[\tilde{S}_t = 3\right]\sigma_0 + I\left[\tilde{S}_t = 4\right]\sigma_1
\end{aligned}$$

Then the AR(1) model can be written as

$$y^*_t = \rho y^*_{t-1} + \varepsilon_t, \varepsilon_t\tilde{\ }N(0,1)$$

where

$$y^*_t = \frac{y_t - \mu_t}{\sigma_t}, y^*_{t-1} = \frac{y_{t-1} - \mu_{t-1}}{\sigma_t}$$

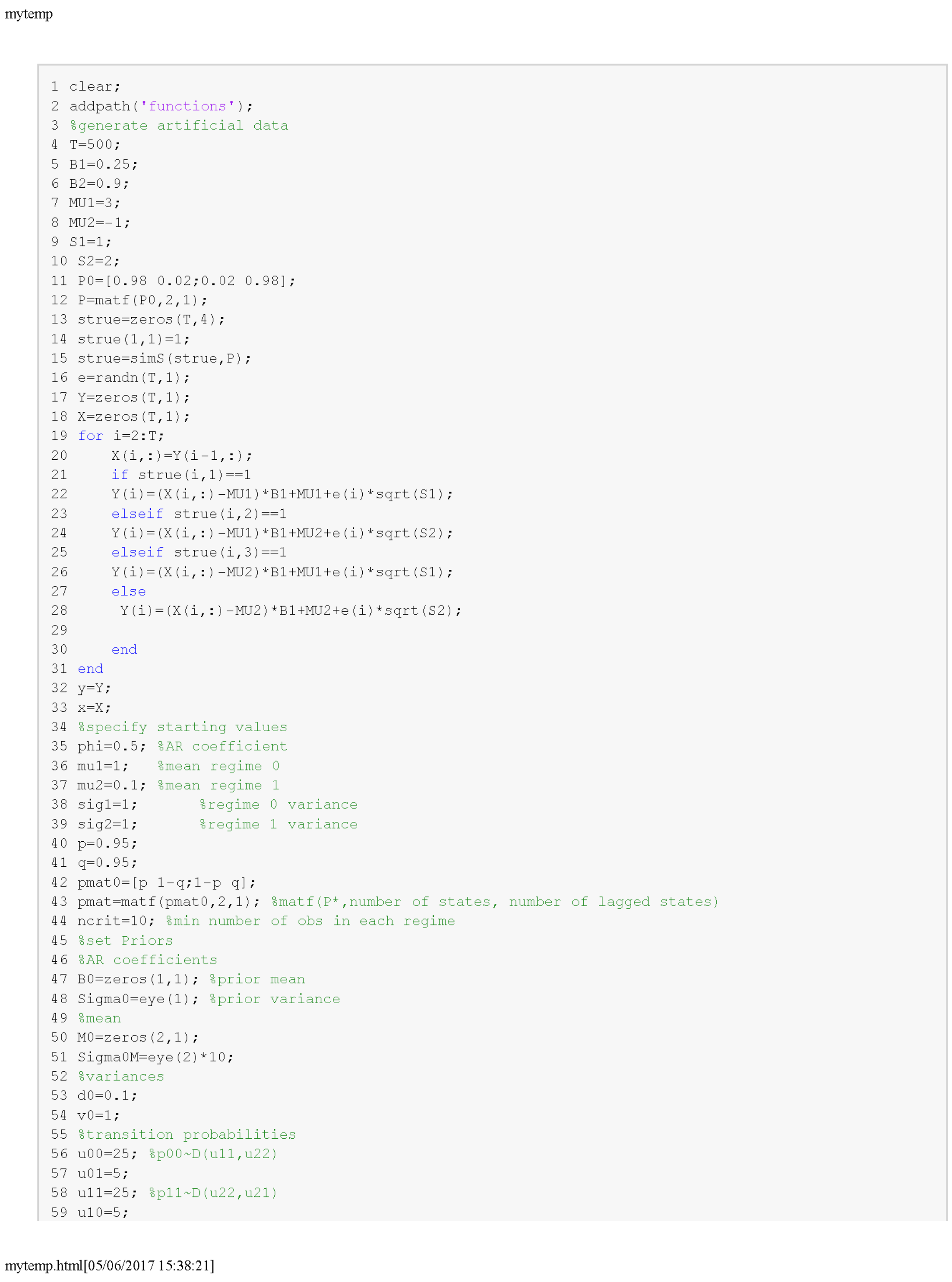
This transformed regression has fixed coefficients an error term with a variance of 1. Given an normal prior for ρ, the conditional posterior is simply the one for a linear regression with a known error variance (see Chapter 1).

(4) Sample from $H\left(\mu_{S_t}|\rho, S_t, \sigma^2_{S_t}, P\right)$. Re-write the AR(1) model as

$$\frac{y_t - py_{t-1}}{\sigma_t} = \frac{\mu_0\left(I\left[\tilde{S}^*_t = 0\right](1-\rho)\right)}{\sigma_t} + \frac{\mu_1\left(I\left[\tilde{S}^*_t = 1\right](1-\rho)\right)}{\sigma_t} + \varepsilon_t, \varepsilon_t\tilde{\ }N(0,1)$$

This simply a linear regression of $\frac{y_t - py_{t-1}}{\sigma_t}$ on 2 dummy variables. As in step 3, with a normal prior for μ_0, μ_1, the posterior of regression with a known error variance applies.

(5) Sample from $H\left(\sigma^2_{S_t}|\mu_{S_t}, \rho, S_t, P\right)$. We assume an inverse Gamma prior for σ^2_0, σ^2_1: $\Gamma^{-1}\left(\frac{\tilde{T}_0}{2}, \frac{\theta_0}{2}\right)$ We calculate the residuals $e_t = y_t - \mu_t - \rho\left(y_t - \mu_{t-1}\right)$. These residuals can be split into the two regimes given by $\tilde{S}^*_t$ and the draw for σ^2_0 and σ^2_1 is made seperately from inverse Gamma distributions: $\Gamma^{-1}\left(\frac{\tilde{T}_0+T_0}{2}, \frac{\theta_0 + e^{[0]\prime}_t e^{[0]}_t}{2}\right)$ and



mytemp

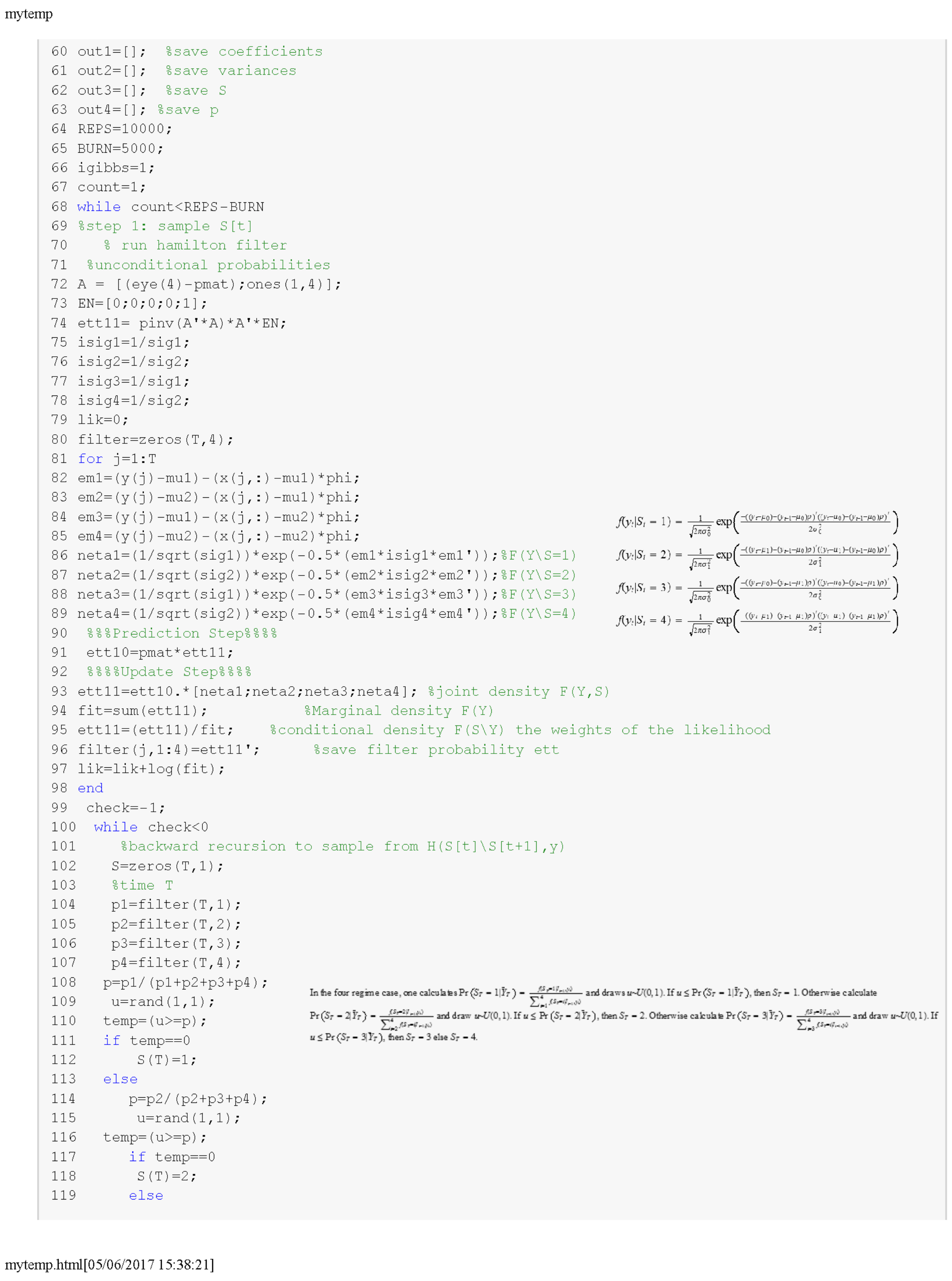
```
1 clear;
2 addpath('functions');
3 %generate artificial data
4 T=500;
5 B1=0.25;
6 B2=0.9;
7 MU1=3;
8 MU2=-1;
9 S1=1;
10 S2=2;
11 P0=[0.98 0.02;0.02 0.98];
12 P=matf(P0,2,1);
13 strue=zeros(T,4);
14 strue(1,1)=1;
15 strue=simS(strue,P);
16 e=randn(T,1);
17 Y=zeros(T,1);
18 X=zeros(T,1);
19 for i=2:T;
20     X(i,:)=Y(i-1,:);
21     if strue(i,1)==1
22     Y(i)=(X(i,:)-MU1)*B1+MU1+e(i)*sqrt(S1);
23     elseif strue(i,2)==1
24     Y(i)=(X(i,:)-MU1)*B1+MU2+e(i)*sqrt(S2);
25     elseif strue(i,3)==1
26     Y(i)=(X(i,:)-MU2)*B1+MU1+e(i)*sqrt(S1);
27     else
28      Y(i)=(X(i,:)-MU2)*B1+MU2+e(i)*sqrt(S2);
29
30     end
31 end
32 y=Y;
33 x=X;
34 %specify starting values
35 phi=0.5; %AR coefficient
36 mu1=1;   %mean regime 0
37 mu2=0.1; %mean regime 1
38 sig1=1;       %regime 0 variance
39 sig2=1;       %regime 1 variance
40 p=0.95;
41 q=0.95;
42 pmat0=[p 1-q;1-p q];
43 pmat=matf(pmat0,2,1); %matf(P*,number of states, number of lagged states)
44 ncrit=10; %min number of obs in each regime
45 %set Priors
46 %AR coefficients
47 B0=zeros(1,1); %prior mean
48 Sigma0=eye(1); %prior variance
49 %mean
50 M0=zeros(2,1);
51 Sigma0M=eye(2)*10;
52 %variances
53 d0=0.1;
54 v0=1;
55 %transition probabilities
56 u00=25; %p00~D(u11,u22)
57 u01=5;
58 u11=25; %p11~D(u22,u21)
59 u10=5;
```

mytemp.html[05/06/2017 15:38:21]

FIGURE 12. Code for AR model with Markov Switching mean.

$\Gamma^{-1}\left(\frac{\tilde{T}_0+T_1}{2},\frac{\theta_0+e_t^{[1]\prime}e_t^{[1]}}{2}\right)$ where $e_t^{[i]}$ denotes the residual selected in regime i, while T_i denotes the number of observations in regime i.

Figures 12 to 15 present the Matlab code for this model (example5.m). This example is based on artificial data generated on lines 4 to 31. Lines 34 to 59 set priors and starting values. The Hamilton filter can be seen on lines 72 to 98. Note the 4 conditional likelihoods that enter the update step on line 93. The backward recursion is coded on lines 101 to 178. As discussed above, as the model has four regimes, minor changes are required to the way the state variable is drawn. These are highlighted in the figure. Line 185 constructs the original state variable and lines 189 to 199 draw the transition probability matrix $P^* = \begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{pmatrix}$. The function matf (based on James Hamilton's



mytemp

```
60 out1=[];  %save coefficients
61 out2=[];  %save variances
62 out3=[];  %save S
63 out4=[]; %save p
64 REPS=10000;
65 BURN=5000;
66 igibbs=1;
67 count=1;
68 while count<REPS-BURN
69 %step 1: sample S[t]
70    % run hamilton filter
71  %unconditional probabilities
72 A = [(eye(4)-pmat);ones(1,4)];
73 EN=[0;0;0;0;1];
74 ett11= pinv(A'*A)*A'*EN;
75 isig1=1/sig1;
76 isig2=1/sig2;
77 isig3=1/sig1;
78 isig4=1/sig2;
79 lik=0;
80 filter=zeros(T,4);
81 for j=1:T
82 em1=(y(j)-mu1)-(x(j,:)-mu1)*phi;
83 em2=(y(j)-mu2)-(x(j,:)-mu1)*phi;
84 em3=(y(j)-mu1)-(x(j,:)-mu2)*phi;
85 em4=(y(j)-mu2)-(x(j,:)-mu2)*phi;
86 neta1=(1/sqrt(sig1))*exp(-0.5*(em1*isig1*em1'));%F(Y\S=1)
87 neta2=(1/sqrt(sig2))*exp(-0.5*(em2*isig2*em2'));%F(Y\S=2)
88 neta3=(1/sqrt(sig1))*exp(-0.5*(em3*isig3*em3'));%F(Y\S=3)
89 neta4=(1/sqrt(sig2))*exp(-0.5*(em4*isig4*em4'));%F(Y\S=4)
90  %%%Prediction Step%%%%
91  ett10=pmat*ett11;
92  %%%%Update Step%%%%
93 ett11=ett10.*[neta1;neta2;neta3;neta4]; %joint density F(Y,S)
94 fit=sum(ett11);           %Marginal density F(Y)
95 ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
96 filter(j,1:4)=ett11';      %save filter probability ett
97 lik=lik+log(fit);
98 end
99  check=-1;
100  while check<0
101     %backward recursion to sample from H(S[t]\S[t+1],y)
102    S=zeros(T,1);
103    %time T
104    p1=filter(T,1);
105    p2=filter(T,2);
106    p3=filter(T,3);
107    p4=filter(T,4);
108   p=p1/(p1+p2+p3+p4);
109    u=rand(1,1);
110   temp=(u>=p);
111   if temp==0
112       S(T)=1;
113   else
114      p=p2/(p2+p3+p4);
115       u=rand(1,1);
116   temp=(u>=p);
117      if temp==0
118       S(T)=2;
119      else
```

$$f(y_t|S_t=1)=\frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-((y_t-\mu_0)-(y_{t-1}-\mu_0)\rho)'((y_t-u_0)-(y_{t-1}-\mu_0)\rho)'}{2\sigma_0^2}\right)$$

$$f(y_t|S_t=2)=\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-((y_t-\mu_1)-(y_{t-1}-\mu_0)\rho)'((y_t-u_1)-(y_{t-1}-\mu_0)\rho)'}{2\sigma_1^2}\right)$$

$$f(y_t|S_t=3)=\frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-((y_t-\mu_0)-(y_{t-1}-\mu_1)\rho)'((y_t-\mu_0)-(y_{t-1}-\mu_1)\rho)'}{2\sigma_0^2}\right)$$

$$f(y_t|S_t=4)=\frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{((y_t\ \mu_1)\ (y_{t-1}\ \mu_1)\rho)'((y_t\ u_1)\ (y_{t-1}\ \mu_1)\rho)'}{2\sigma_1^2}\right)$$

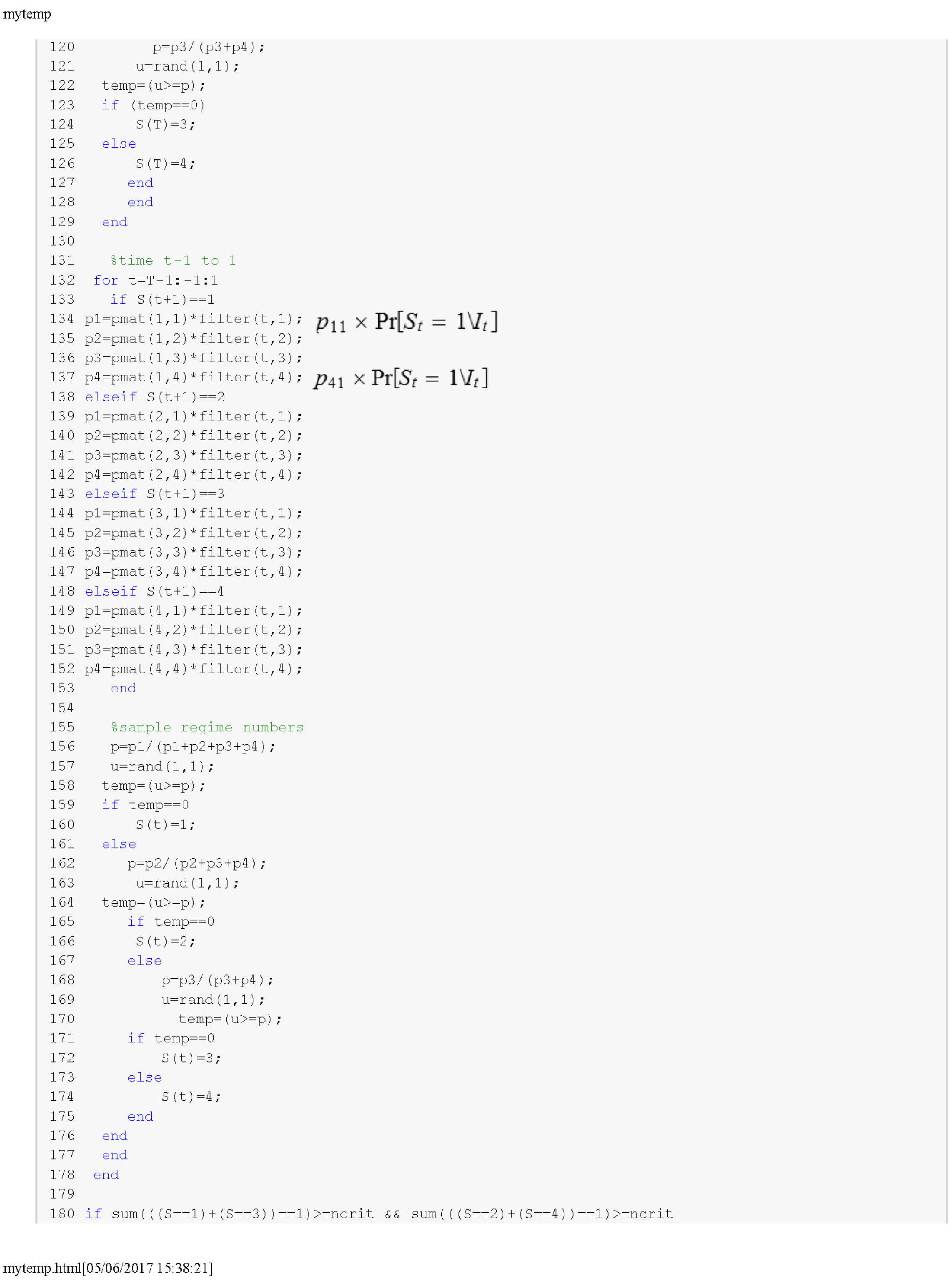
In the four regime case, one calculates $\Pr\left(S_T=1|\tilde{Y}_T\right)=\frac{f(S_T=1|\tilde{Y}_{T-1},y_t)}{\sum_{j=1}^{4} f(S_T=j|\tilde{Y}_{T-1},y_t)}$ and draws $u\sim U(0,1)$. If $u\leq\Pr\left(S_T=1|\tilde{Y}_T\right)$, then $S_T=1$. Otherwise calculate $\Pr\left(S_T=2|\tilde{Y}_T\right)=\frac{f(S_T=2|\tilde{Y}_{T-1},y_t)}{\sum_{j=2}^{4} f(S_T=j|\tilde{Y}_{T-1},y_t)}$ and draw $u\sim U(0,1)$. If $u\leq\Pr\left(S_T=2|\tilde{Y}_T\right)$, then $S_T=2$. Otherwise calculate $\Pr\left(S_T=3|\tilde{Y}_T\right)=\frac{f(S_T=3|\tilde{Y}_{T-1},y_t)}{\sum_{j=3}^{4} f(S_T=j|\tilde{Y}_{T-1},y_t)}$ and draw $u\sim U(0,1)$. If $u\leq\Pr\left(S_T=3|\tilde{Y}_T\right)$, then $S_T=3$ else $S_T=4$.

mytemp.html[05/06/2017 15:38:21]

FIGURE 13. Code for AR model with Markov Switching mean.

code), constructs $P=\begin{pmatrix} p_{00} & 0 & p_{00} & 0 \\ p_{01} & 0 & p_{01} & 0 \\ 0 & p_{10} & 0 & p_{10} \\ 0 & p_{11} & 0 & p_{11} \end{pmatrix}$ from P^*. Its arguments are P^*, number of regimes and number of lagged states. Lines 203 to 210 draw the AR coefficient, while μ_0 and μ_1 are drawn on lines 213 to 219. Lines 221 onwards split the residual series into the two regimes and draws σ_0^2 and σ_1^2.

7.3. Markov switching model with time varying transition probabilties. In the MS models considered so far, the transition probabilties are fixed. Following Filardo and Gordon (1998) amongst others, this assumption can be relaxed and the transition probabilities can be made functions of exogenous regressors. Consider the two



mytemp

```
        p=p3/(p3+p4);
      u=rand(1,1);
  temp=(u>=p);
  if (temp==0)
      S(T)=3;
  else
      S(T)=4;
     end
     end
  end

   %time t-1 to 1
 for t=T-1:-1:1
   if S(t+1)==1
p1=pmat(1,1)*filter(t,1);
p2=pmat(1,2)*filter(t,2);
p3=pmat(1,3)*filter(t,3);
p4=pmat(1,4)*filter(t,4);
elseif S(t+1)==2
p1=pmat(2,1)*filter(t,1);
p2=pmat(2,2)*filter(t,2);
p3=pmat(2,3)*filter(t,3);
p4=pmat(2,4)*filter(t,4);
elseif S(t+1)==3
p1=pmat(3,1)*filter(t,1);
p2=pmat(3,2)*filter(t,2);
p3=pmat(3,3)*filter(t,3);
p4=pmat(3,4)*filter(t,4);
elseif S(t+1)==4
p1=pmat(4,1)*filter(t,1);
p2=pmat(4,2)*filter(t,2);
p3=pmat(4,3)*filter(t,3);
p4=pmat(4,4)*filter(t,4);
   end

   %sample regime numbers
   p=p1/(p1+p2+p3+p4);
   u=rand(1,1);
  temp=(u>=p);
  if temp==0
      S(t)=1;
  else
     p=p2/(p2+p3+p4);
      u=rand(1,1);
  temp=(u>=p);
     if temp==0
      S(t)=2;
     else
         p=p3/(p3+p4);
         u=rand(1,1);
           temp=(u>=p);
     if temp==0
         S(t)=3;
     else
         S(t)=4;
     end
  end
  end
 end

if sum(((S==1)+(S==3))==1)>=ncrit && sum(((S==2)+(S==4))==1)>=ncrit
```

$p_{11} \times \Pr[S_t = 1 \backslash I_t]$

$p_{41} \times \Pr[S_t = 1 \backslash I_t]$

mytemp.html[05/06/2017 15:38:21]

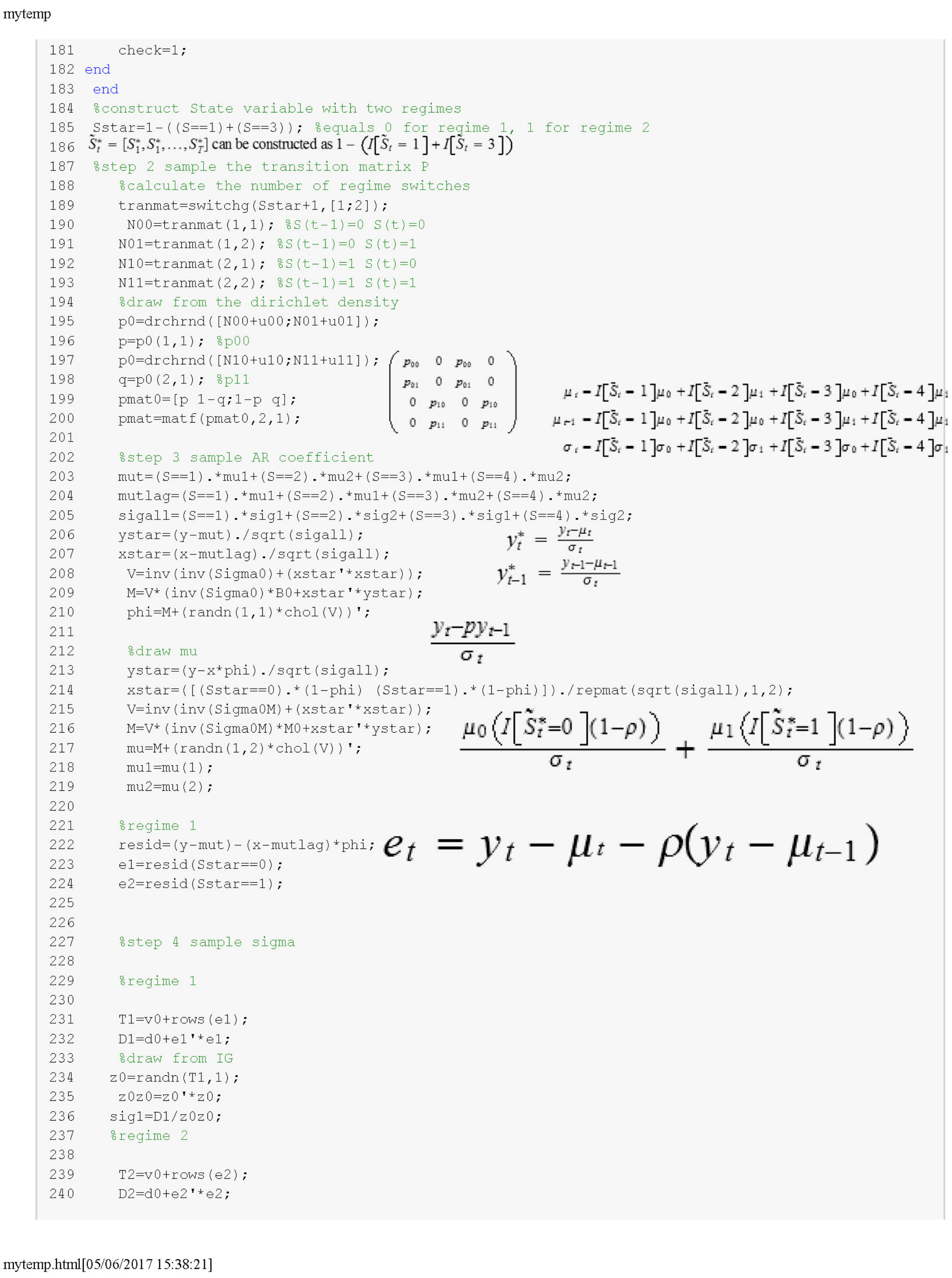
FIGURE 14. Code for AR model with Markov Switching mean.

regime MS model:

$$\begin{aligned} y_t &= x_t b_{S_t} + v_t, v_t\tilde{}N(0,\sigma^2_{S_t}) \\ b_{S_t} &= b_0(1-S_t) + b_1 S_t \\ \sigma^2_{S_t} &= \sigma^2_0(1-S_t) + \sigma^2_1 S_t \end{aligned} \tag{7.4}$$

The transition probabilties are now given by $P_t = \begin{pmatrix} p_{00}\left(z_t\right) & p_{10}\left(z_t\right) \\ p_{01}\left(z_t\right) & p_{11}\left(z_t\right) \end{pmatrix}$ where z_t denotes a set of regressors. The evolution of the state variable S_t can be described using a Probit model

$$\begin{aligned} P\left(S_t = 0\right) &= S_t^* < 0 \\ S_t^* &= \gamma_0 + \lambda z_{t-1} + \gamma_1 S_{t-1} + u_t, u_t\tilde{}N(0,1) \end{aligned}$$



mytemp

```
    check=1;
end
 end
 %construct State variable with two regimes
 Sstar=1-((S==1)+(S==3)); %equals 0 for regime 1, 1 for regime 2
```

$\tilde{S}_t^* = [S_1^*, S_1^*, \ldots, S_T^*]$ can be constructed as $1 - \left(I[\tilde{S}_t = 1] + I[\tilde{S}_t = 3]\right)$

```
 %step 2 sample the transition matrix P
    %calculate the number of regime switches
    tranmat=switchg(Sstar+1,[1;2]);
     N00=tranmat(1,1); %S(t-1)=0 S(t)=0
    N01=tranmat(1,2); %S(t-1)=0 S(t)=1
    N10=tranmat(2,1); %S(t-1)=1 S(t)=0
    N11=tranmat(2,2); %S(t-1)=1 S(t)=1
    %draw from the dirichlet density
    p0=drchrnd([N00+u00;N01+u01]);
    p=p0(1,1); %p00
    p0=drchrnd([N10+u10;N11+u11]);
    q=p0(2,1); %p11
    pmat0=[p 1-q;1-p q];
    pmat=matf(pmat0,2,1);
```

$$\begin{pmatrix} p_{00} & 0 & p_{00} & 0 \\ p_{01} & 0 & p_{01} & 0 \\ 0 & p_{10} & 0 & p_{10} \\ 0 & p_{11} & 0 & p_{11} \end{pmatrix}$$

$$\mu_t = I[\tilde{S}_t = 1]\mu_0 + I[\tilde{S}_t = 2]\mu_1 + I[\tilde{S}_t = 3]\mu_0 + I[\tilde{S}_t = 4]\mu_1$$
$$\mu_{t-1} = I[\tilde{S}_t = 1]\mu_0 + I[\tilde{S}_t = 2]\mu_0 + I[\tilde{S}_t = 3]\mu_1 + I[\tilde{S}_t = 4]\mu_1$$
$$\sigma_t = I[\tilde{S}_t = 1]\sigma_0 + I[\tilde{S}_t = 2]\sigma_1 + I[\tilde{S}_t = 3]\sigma_0 + I[\tilde{S}_t = 4]\sigma_1$$

```
    %step 3 sample AR coefficient
    mut=(S==1).*mu1+(S==2).*mu2+(S==3).*mu1+(S==4).*mu2;
    mutlag=(S==1).*mu1+(S==2).*mu1+(S==3).*mu2+(S==4).*mu2;
    sigall=(S==1).*sig1+(S==2).*sig2+(S==3).*sig1+(S==4).*sig2;
    ystar=(y-mut)./sqrt(sigall);
    xstar=(x-mutlag)./sqrt(sigall);
     V=inv(inv(Sigma0)+(xstar'*xstar));
     M=V*(inv(Sigma0)*B0+xstar'*ystar);
     phi=M+(randn(1,1)*chol(V))';
```

$$y_t^* = \frac{y_t - \mu_t}{\sigma_t}$$
$$y_{t-1}^* = \frac{y_{t-1} - \mu_{t-1}}{\sigma_t}$$

$$\frac{y_t - p y_{t-1}}{\sigma_t}$$

```
     %draw mu
     ystar=(y-x*phi)./sqrt(sigall);
     xstar=([(Sstar==0).*(1-phi) (Sstar==1).*(1-phi)])./repmat(sqrt(sigall),1,2);
     V=inv(inv(Sigma0M)+(xstar'*xstar));
     M=V*(inv(Sigma0M)*M0+xstar'*ystar);
     mu=M+(randn(1,2)*chol(V))';
     mu1=mu(1);
     mu2=mu(2);
```

$$\frac{\mu_0\left(I[\tilde{S}_t^*=0](1-\rho)\right)}{\sigma_t} + \frac{\mu_1\left(I[\tilde{S}_t^*=1](1-\rho)\right)}{\sigma_t}$$

```
    %regime 1
    resid=(y-mut)-(x-mutlag)*phi;
    e1=resid(Sstar==0);
    e2=resid(Sstar==1);
```

$$e_t = y_t - \mu_t - \rho(y_t - \mu_{t-1})$$

```


    %step 4 sample sigma

    %regime 1

    T1=v0+rows(e1);
    D1=d0+e1'*e1;
    %draw from IG
   z0=randn(T1,1);
    z0z0=z0'*z0;
   sig1=D1/z0z0;
   %regime 2

    T2=v0+rows(e2);
    D2=d0+e2'*e2;
```

mytemp.html[05/06/2017 15:38:21]

Figure 15. AR model with Markov switching mean.

where S_t^* is an unobserved latent variable. Given the normality of u_t the transition probabilties can be calculating using the normal CDF.

$$\Pr[S_t = 0|S_{t-1} = 0] = \Pr[u_t < -\gamma_0 - \lambda z_{t-1} - \gamma_1 S_{t-1}]$$

Recall that $S_{t-1} = 0$ and denote the normal CDF by $\Phi(.)$:

$$\Pr[S_t = 0|S_{t-1} = 0] = \Phi(-\gamma_0 - \lambda z_{t-1})$$

Similarly

$$\begin{aligned} \Pr[S_t = 1|S_{t-1} = 1] &= \Pr[u_t \geq -\gamma_0 - \lambda z_{t-1} - \gamma_1 S_{t-1}] \\ &= 1 - \Phi(-\gamma_0 - \lambda z_{t-1} - \gamma_1 S_{t-1}) \end{aligned}$$

The Gibbs sampling algorithm for this model thus involves extra steps to draw S_t^* and the coefficients $\Gamma = [\gamma_0, \lambda, \gamma_1]$. There are only minor modifications to the remaining steps of the algorithm. The algorithm samples from the following conditional posterior distributions:

(1) Sample from $H\left(S_t|b_{S_t}, \sigma^2_{S_t}, \Gamma, S_t^*\right)$. The only modification required to this step is the fact that there is a different transition probability matrix at each point in time. This needs to be taken into account while running the Hamilton filter–the prediction step is now given as

$$\xi_{t|t-1} = P_t \xi_{t-1|t-1}$$

Similarly, the backward recursion using $H\left(S_{t+1}|S_t\right) H\left(S_t|\tilde{Y}_t\right)$ is modified as $H\left(S_{t+1}|S_t\right) = P_{t+1}$ is different for $t = T-1, T-2, ..., 1$.

(2) Sample from $H\left(S_t^*|b_{S_t}, \sigma^2_{S_t}, \Gamma, S_t\right)$. Following Albert and Chib (1993), S_t^* can be sampled from the following truncated normal distributions for $t = 1, 2, ..., T$

$$\begin{aligned} S_t^{*}\tilde{}N_{LT}\left(m, 1\right) \text{ if } S_t &= 1 \\ S_t^{*}\tilde{}N_{RT}\left(m, 1\right) \text{ if } S_t &= 0 \end{aligned}$$

where $N_{LT}\left(m, 1\right)$ is the $N(m, 1)$ distribution left truncated at zero, while $N_{RT}\left(m, 1\right)$ is the $N(m, 1)$ distribution right truncated at zero. Note that $m = \gamma_0 + \lambda z_{t-1} + \gamma_1 S_{t-1}$.

(3) Sample from $H\left(\Gamma|S_t^*, b_{S_t}, \sigma^2_{S_t}, S_t\right)$. Given S_t^* the probability equation is a simple linear regression with a known error variance:

$$S_t^* = \gamma_0 + \lambda z_{t-1} + \gamma_1 S_{t-1} + u_t, u_t\tilde{}N(0, 1)$$

Given a normal prior $N(\Gamma_0, \Sigma_\Gamma)$ the conditional posterior is also Normal $N(M, V)$

$$\begin{aligned} V &= \left(\Sigma_\Gamma^{-1} + \tilde{z}_t'\tilde{z}_t\right)^{-1} \\ M &= V\left(\Sigma_\Gamma^{-1}\Gamma_0 + \tilde{z}_t' S_t^*\right) \end{aligned}$$

where $\tilde{z}_t = [1, z_{t-1}, S_{t-1}]$.

(4) Sample from $H\left(b_{S_t}|S_t, \sigma^2_{S_t}, \Gamma, S_t^*\right)$. No changes are required in this step (see section 6).

(5) Sample from $H\left(\sigma^2_{S_t}|b_{S_t}, S_t, \Gamma, S_t^*\right)$. No changes are required in this step (see section 6).

Figures 16 to 19 show the Matlab code for this model. As before, we generate artificial data from this model (lines 4 to 40). Lines 62 and 63 set a normal prior for the coefficients of the probability equation Γ. The Gibbs sampler starts on line 75. While drawing the state variable, the Hamilton filter is run on lines 81 to 106. Note on lines 91 and 92 that the transition probability matrix changes at each point in time. Lines 110 to 143 show the code for the backward recursion. Note on lines 122 and 123 the transition probabilties are not fixed over time. The next step in the Gibbs sampler is the draw of S_t^* from truncated normal distributions. This is done on lines 149 to 161. Lines 163 to 164 calculate the transition probabilties using the normal CDF. Lines 167 to 171 draw Γ from its conditional posterior distribution. The remaining code draws from the conditional posterior of b_{S_t} and $\sigma^2_{S_t}$.

7.4. A regression with Markov switching coefficients and a structural break in the variance. As a final extension we consider the following model:

$$\begin{aligned} y_t &= x_t b_{S_t} + v_t, v_t\tilde{}N(0, \sigma^2_{V_t}) \qquad (7.5) \\ b_{S_t} &= b_0(1 - S_t) + b_1 S_t \\ \sigma^2_{V_t} &= \sigma^2_0(1 - V_t) + \sigma^2_1 V_t \end{aligned}$$

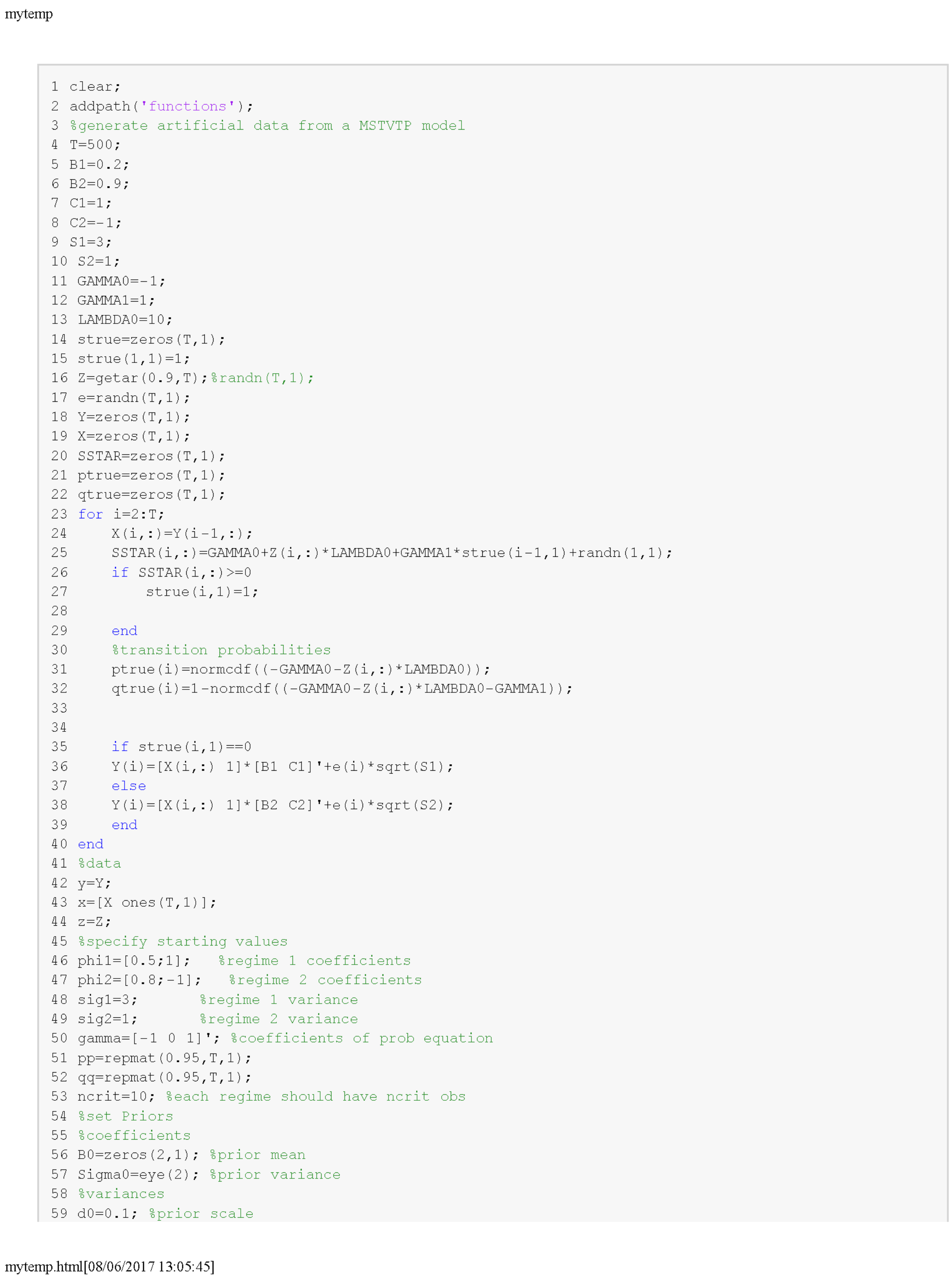
where S_t and V_t follow two independent Markov chains with transition probabilties

$$\begin{aligned} P &= \begin{pmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{pmatrix} \\ Q &= \begin{pmatrix} q_{00} & 0 \\ q_{01} & 1 \end{pmatrix} \end{aligned}$$

This model has two new features. First, switching in the coefficients and the variance occurs independently – we do not make the assumption that all parameters undergo regime shifts at the same time. Second, following Chib (1998), the matrix Q is restricted to impose one change point or break for the variance. We assume that $V_1 = 0$. With $\Pr\left[V_t = 1|V_{t-1} = 1\right] = 1$, once the process switches to regime 1, that regime persists for ever. That is, with $\Pr\left[V_t = 0|V_{t-1} = 1\right] = 0$, there is no possibility of a switch from regime 1 to regime 0.

The Gibbs sampling algorithm for this model samples from the following conditional posterior distributions:

(1) Sample from $H\left(S_t|P, Q, V_t, b_{S_t}, \sigma^2_{V_t}\right)$. As the variance switches independently, it has to be treated differently in the Hamilton filter when compared to the basic model. In particular, at each point in time, the conditional



mytemp

```
clear;
addpath('functions');
%generate artificial data from a MSTVTP model
T=500;
B1=0.2;
B2=0.9;
C1=1;
C2=-1;
S1=3;
S2=1;
GAMMA0=-1;
GAMMA1=1;
LAMBDA0=10;
strue=zeros(T,1);
strue(1,1)=1;
Z=getar(0.9,T);%randn(T,1);
e=randn(T,1);
Y=zeros(T,1);
X=zeros(T,1);
SSTAR=zeros(T,1);
ptrue=zeros(T,1);
qtrue=zeros(T,1);
for i=2:T;
    X(i,:)=Y(i-1,:);
    SSTAR(i,:)=GAMMA0+Z(i,:)*LAMBDA0+GAMMA1*strue(i-1,1)+randn(1,1);
    if SSTAR(i,:)>=0
        strue(i,1)=1;

    end
    %transition probabilities
    ptrue(i)=normcdf((-GAMMA0-Z(i,:)*LAMBDA0));
    qtrue(i)=1-normcdf((-GAMMA0-Z(i,:)*LAMBDA0-GAMMA1));


    if strue(i,1)==0
    Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S1);
    else
    Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S2);
    end
end
%data
y=Y;
x=[X ones(T,1)];
z=Z;
%specify starting values
phi1=[0.5;1];   %regime 1 coefficients
phi2=[0.8;-1];   %regime 2 coefficients
sig1=3;       %regime 1 variance
sig2=1;       %regime 2 variance
gamma=[-1 0 1]'; %coefficients of prob equation
pp=repmat(0.95,T,1);
qq=repmat(0.95,T,1);
ncrit=10; %each regime should have ncrit obs
%set Priors
%coefficients
B0=zeros(2,1); %prior mean
Sigma0=eye(2); %prior variance
%variances
d0=0.1; %prior scale
```

mytemp.html[08/06/2017 13:05:45]

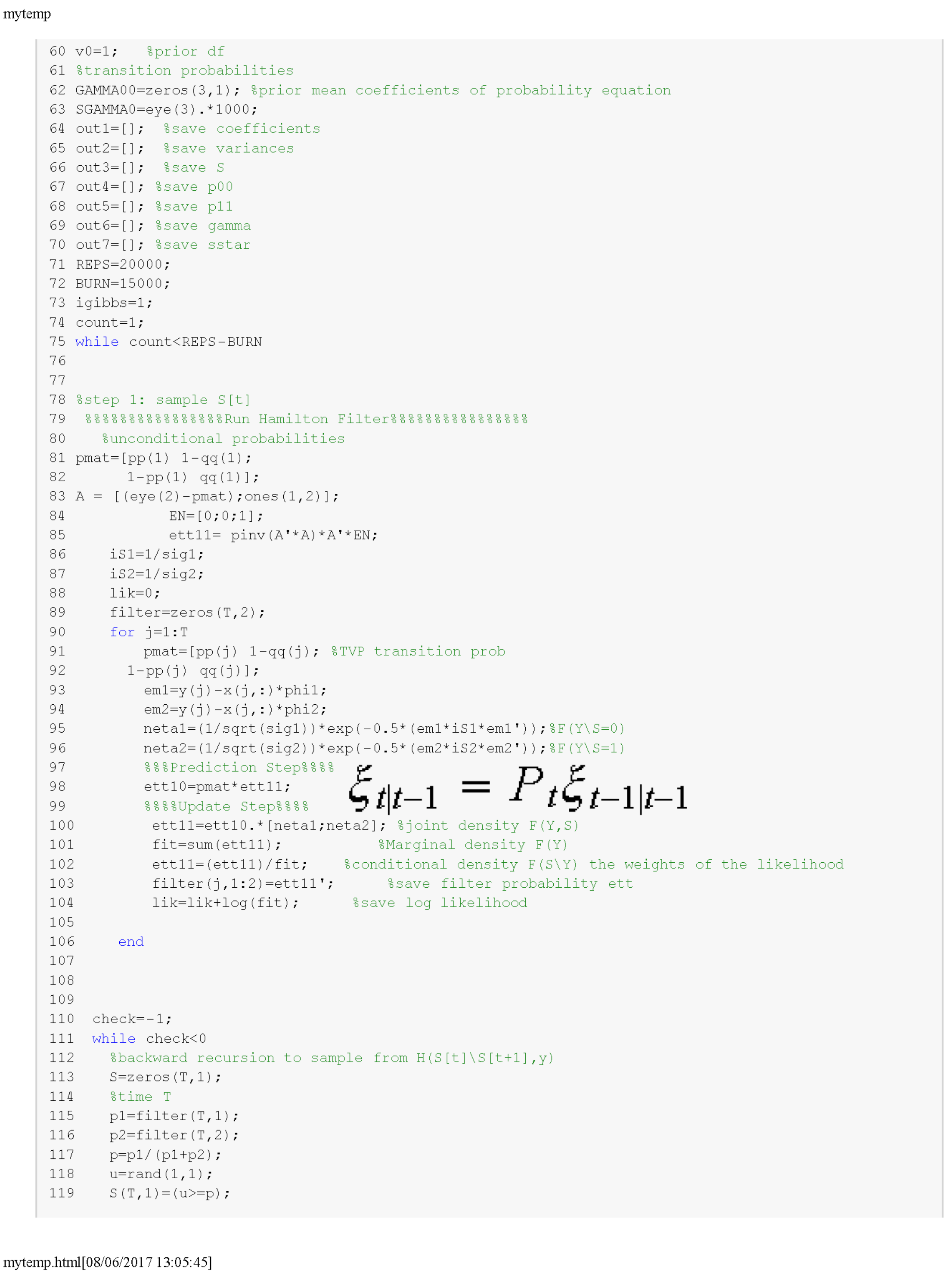
Figure 16. MS model with time-varying transition probabilities.

likelihoods are given by:

$$\begin{aligned} f\left(y_t|S_t=0\right) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_0^2}\right) \\ f\left(y_t|S_t=1\right) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_0^2}\right) \end{aligned} \quad \text{if } V_t = 0$$

$$\begin{aligned} f\left(y_t|S_t=0\right) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_1^2}\right) \\ f\left(y_t|S_t=1\right) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_1^2}\right) \end{aligned} \quad \text{if } V_t = 1$$

As we condition on V_t this change is simple to implement. There is no change in the backward recursion used to draw S_t.



mytemp

```
v0=1;   %prior df
%transition probabilities
GAMMA00=zeros(3,1); %prior mean coefficients of probability equation
SGAMMA0=eye(3).*1000;
out1=[];  %save coefficients
out2=[];  %save variances
out3=[];  %save S
out4=[]; %save p00
out5=[]; %save p11
out6=[]; %save gamma
out7=[]; %save sstar
REPS=20000;
BURN=15000;
igibbs=1;
count=1;
while count<REPS-BURN


%step 1: sample S[t]
 %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
   %unconditional probabilities
pmat=[pp(1) 1-qq(1);
      1-pp(1) qq(1)];
A = [(eye(2)-pmat);ones(1,2)];
           EN=[0;0;1];
           ett11= pinv(A'*A)*A'*EN;
    iS1=1/sig1;
    iS2=1/sig2;
    lik=0;
    filter=zeros(T,2);
    for j=1:T
        pmat=[pp(j) 1-qq(j); %TVP transition prob
      1-pp(j) qq(j)];
        em1=y(j)-x(j,:)*phi1;
        em2=y(j)-x(j,:)*phi2;
        neta1=(1/sqrt(sig1))*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
        neta2=(1/sqrt(sig2))*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
        %%%Prediction Step%%%%
        ett10=pmat*ett11;
        %%%%Update Step%%%%
         ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
         fit=sum(ett11);           %Marginal density F(Y)
         ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
         filter(j,1:2)=ett11';      %save filter probability ett
         lik=lik+log(fit);      %save log likelihood

     end



  check=-1;
  while check<0
    %backward recursion to sample from H(S[t]\S[t+1],y)
    S=zeros(T,1);
    %time T
    p1=filter(T,1);
    p2=filter(T,2);
    p=p1/(p1+p2);
    u=rand(1,1);
    S(T,1)=(u>=p);
```

$$\xi_{t|t-1} = P_t \xi_{t-1|t-1}$$

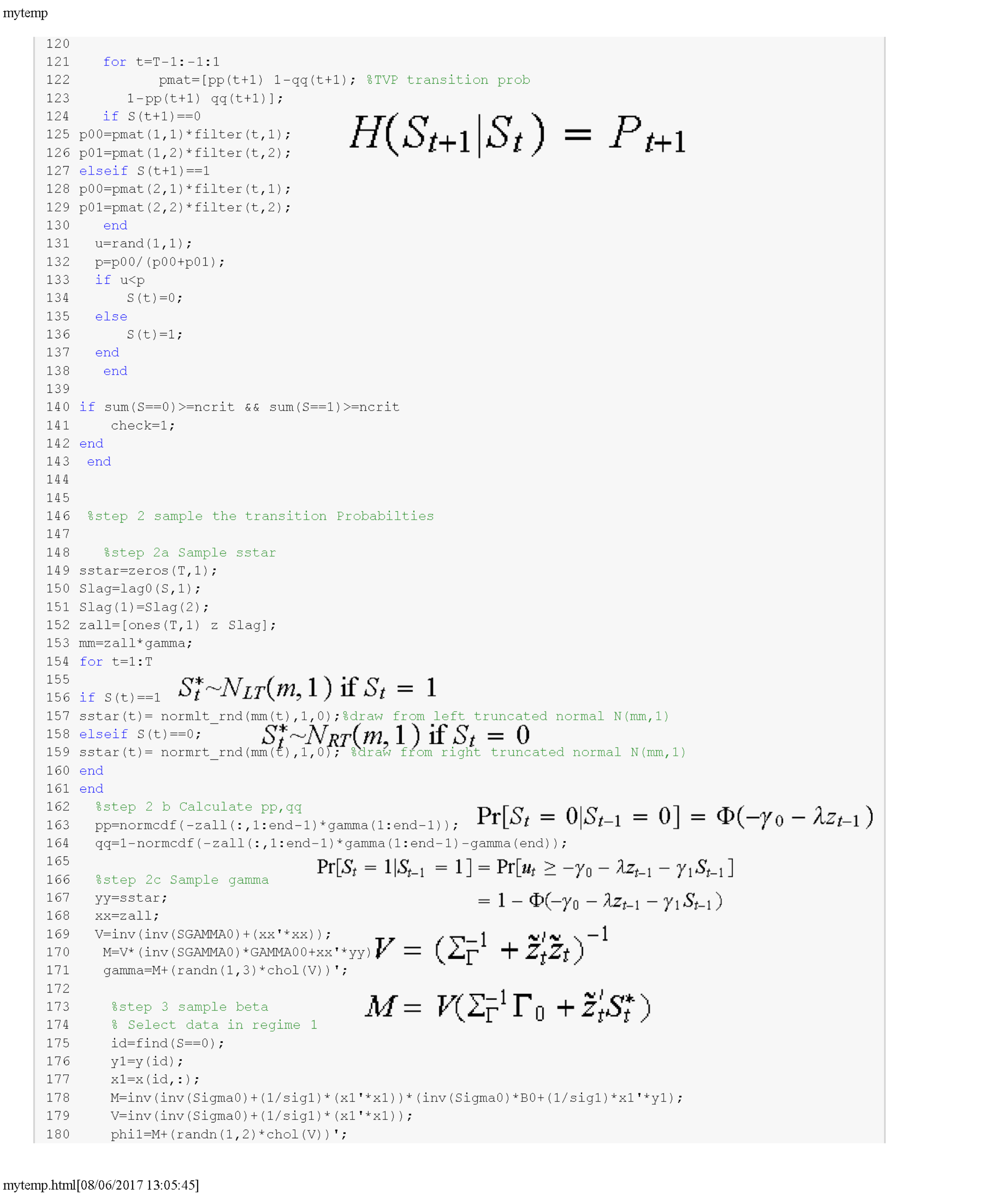
mytemp.html[08/06/2017 13:05:45]

FIGURE 17. MS model with time varying transition probabilties

(2) Sample from $H\left(V_t|P,Q,S_t,b_{S_t},\sigma^2_{V_t}\right)$. As in step 1, the conditional likelihoods in the Hamilton filter need to take into account the fact that the regression coefficients switch independently. Hence the conditional likelihoods are:

$$\begin{aligned}
f\left(y_t|V_t=0\right) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_0^2}\right) \\
f\left(y_t|V_t=1\right) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_1^2}\right)
\end{aligned} \quad \text{if } S_t = 0$$

$$\begin{aligned}
f\left(y_t|V_t=0\right) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_0^2}\right) \\
f\left(y_t|V_t=1\right) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_1^2}\right)
\end{aligned} \quad \text{if } S_t = 1$$



mytemp

```

   for t=T-1:-1:1
          pmat=[pp(t+1) 1-qq(t+1); %TVP transition prob
      1-pp(t+1) qq(t+1)];
   if S(t+1)==0
p00=pmat(1,1)*filter(t,1);
p01=pmat(1,2)*filter(t,2);
```

$$H(S_{t+1}|S_t) = P_{t+1}$$

```
elseif S(t+1)==1
p00=pmat(2,1)*filter(t,1);
p01=pmat(2,2)*filter(t,2);
   end
  u=rand(1,1);
  p=p00/(p00+p01);
  if u<p
      S(t)=0;
  else
      S(t)=1;
  end
   end

if sum(S==0)>=ncrit && sum(S==1)>=ncrit
    check=1;
end
 end


 %step 2 sample the transition Probabilties

   %step 2a Sample sstar
sstar=zeros(T,1);
Slag=lag0(S,1);
Slag(1)=Slag(2);
zall=[ones(T,1) z Slag];
mm=zall*gamma;
for t=1:T

if S(t)==1
```

$$S_t^*\sim N_{LT}(m,1) \text{ if } S_t = 1$$

```
sstar(t)= normlt_rnd(mm(t),1,0);%draw from left truncated normal N(mm,1)
elseif S(t)==0;
```

$$S_t^*\sim N_{RT}(m,1) \text{ if } S_t = 0$$

```
sstar(t)= normrt_rnd(mm(t),1,0); %draw from right truncated normal N(mm,1)
end
end
  %step 2 b Calculate pp,qq
  pp=normcdf(-zall(:,1:end-1)*gamma(1:end-1));
```

$$\Pr[S_t = 0|S_{t-1} = 0] = \Phi(-\gamma_0 - \lambda z_{t-1})$$

```
  qq=1-normcdf(-zall(:,1:end-1)*gamma(1:end-1)-gamma(end));

```

$$\begin{aligned}\Pr[S_t = 1|S_{t-1} = 1] &= \Pr[u_t \geq -\gamma_0 - \lambda z_{t-1} - \gamma_1 S_{t-1}] \\ &= 1 - \Phi(-\gamma_0 - \lambda z_{t-1} - \gamma_1 S_{t-1})\end{aligned}$$

```
  %step 2c Sample gamma
  yy=sstar;
  xx=zall;
  V=inv(inv(SGAMMA0)+(xx'*xx));
   M=V*(inv(SGAMMA0)*GAMMA00+xx'*yy)
```

$$V = (\Sigma_\Gamma^{-1} + \tilde{z}_t'\tilde{z}_t)^{-1}$$

```
   gamma=M+(randn(1,3)*chol(V))';

    %step 3 sample beta
    % Select data in regime 1
```

$$M = V(\Sigma_\Gamma^{-1}\Gamma_0 + \tilde{z}_t' S_t^*)$$

```
    id=find(S==0);
    y1=y(id);
    x1=x(id,:);
    M=inv(inv(Sigma0)+(1/sig1)*(x1'*x1))*(inv(Sigma0)*B0+(1/sig1)*x1'*y1);
    V=inv(inv(Sigma0)+(1/sig1)*(x1'*x1));
    phi1=M+(randn(1,2)*chol(V))';
```

mytemp.html[08/06/2017 13:05:45]

FIGURE 18. MS model with time-varying transition probabilties.

We also assume that $\xi_{1|1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. There is no change to be made in the backward recursion.

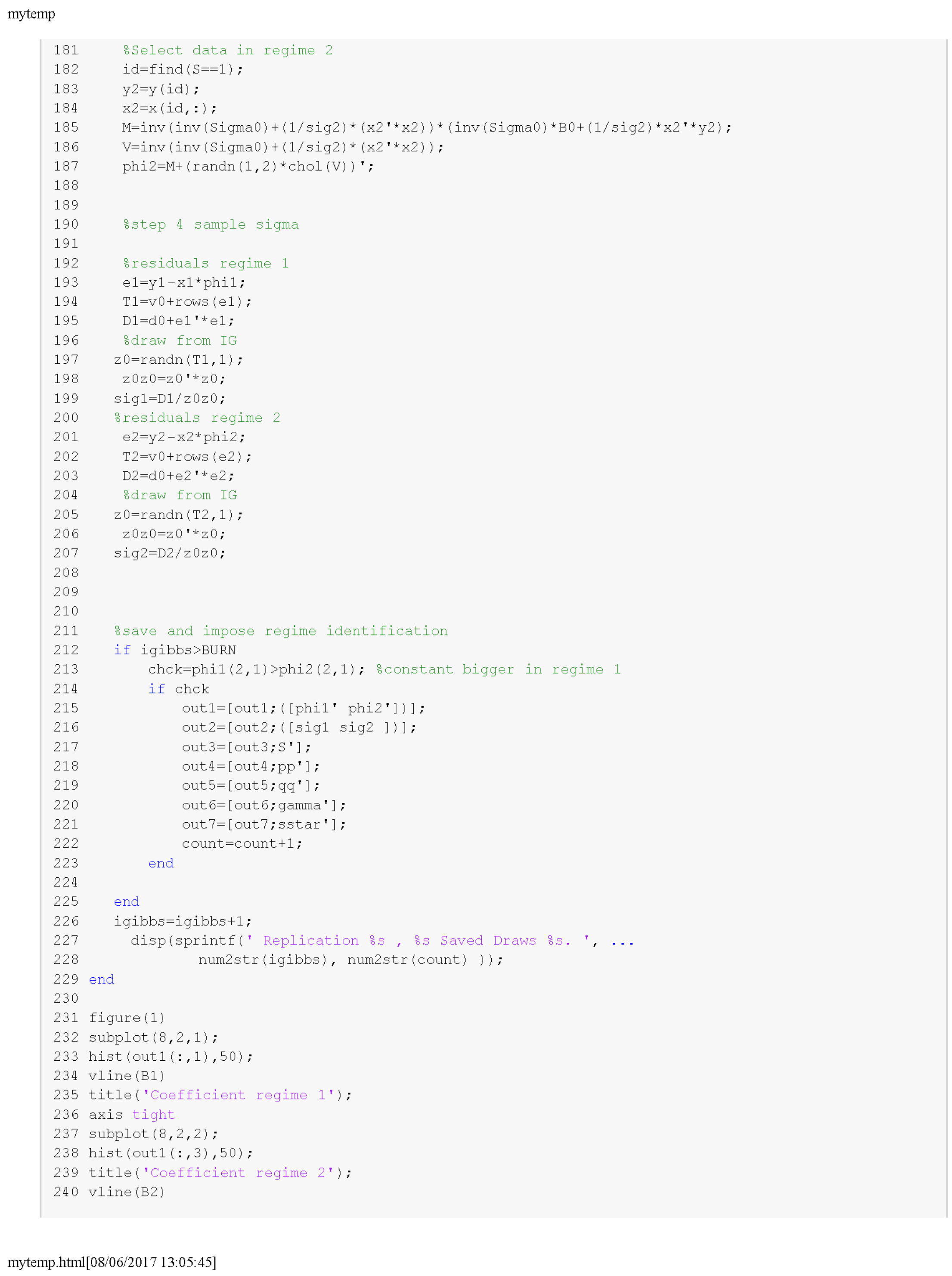
(3) $H\left(P|V_t, Q, S_t, b_{S_t}, \sigma^2_{V_t}\right)$. Given a Dirichlet prior, the columns of P are drawn as in section 3.1.

(4) $H\left(Q|V_t, P, S_t, b_{S_t}, \sigma^2_{V_t}\right)$. Given a Dirichlet prior, the first columns of Q is drawn as in section 3.1.

(5) $H\left(b_{S_t}|Q, V_t, P, S_t, \sigma^2_{V_t}\right)$. Define $\sigma_t = I\left[V_t = 0\right]\sigma_0 + I\left[V_t = 1\right]\sigma_1$ and write the regression as

$$\frac{y_t}{\sigma_t} = \frac{x_t}{\sigma_t} b_{S_t} + e_t, e_t \tilde{} N(0,1)$$

Given S_t, a regression with unit error variance applies when $S_t = 0$ and $S_t = 1$ and b_{S_t} is drawn easily from its (Normal) conditional posterior.



mytemp

```
181     %Select data in regime 2
182     id=find(S==1);
183     y2=y(id);
184     x2=x(id,:);
185     M=inv(inv(Sigma0)+(1/sig2)*(x2'*x2))*(inv(Sigma0)*B0+(1/sig2)*x2'*y2);
186     V=inv(inv(Sigma0)+(1/sig2)*(x2'*x2));
187     phi2=M+(randn(1,2)*chol(V))';
188 
189 
190     %step 4 sample sigma
191 
192     %residuals regime 1
193     e1=y1-x1*phi1;
194     T1=v0+rows(e1);
195     D1=d0+e1'*e1;
196     %draw from IG
197    z0=randn(T1,1);
198     z0z0=z0'*z0;
199    sig1=D1/z0z0;
200    %residuals regime 2
201     e2=y2-x2*phi2;
202     T2=v0+rows(e2);
203     D2=d0+e2'*e2;
204     %draw from IG
205    z0=randn(T2,1);
206     z0z0=z0'*z0;
207    sig2=D2/z0z0;
208 
209 
210 
211    %save and impose regime identification
212    if igibbs>BURN
213        chck=phi1(2,1)>phi2(2,1); %constant bigger in regime 1
214        if chck
215            out1=[out1;([phi1' phi2'])];
216            out2=[out2;([sig1 sig2 ])];
217            out3=[out3;S'];
218            out4=[out4;pp'];
219            out5=[out5;qq'];
220            out6=[out6;gamma'];
221            out7=[out7;sstar'];
222            count=count+1;
223        end
224 
225    end
226    igibbs=igibbs+1;
227      disp(sprintf(' Replication %s , %s Saved Draws %s. ', ...
228              num2str(igibbs), num2str(count) ));
229 end
230 
231 figure(1)
232 subplot(8,2,1);
233 hist(out1(:,1),50);
234 vline(B1)
235 title('Coefficient regime 1');
236 axis tight
237 subplot(8,2,2);
238 hist(out1(:,3),50);
239 title('Coefficient regime 2');
240 vline(B2)
```

mytemp.html[08/06/2017 13:05:45]

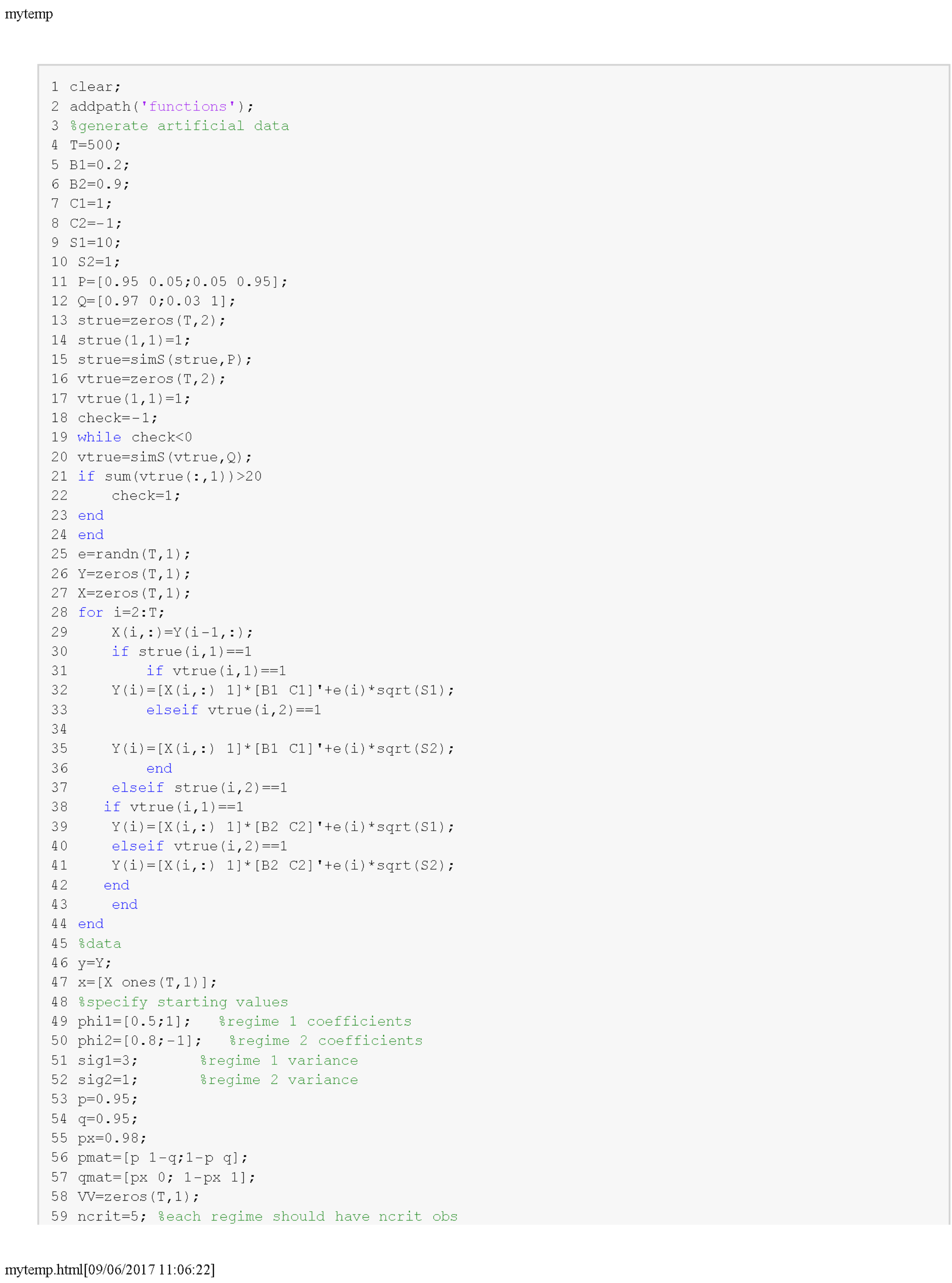
FIGURE 19. MS model with time-varying transition probabilties.

(6) $H\left(\sigma^2_{V_t}|b_{S_t}, Q, V_t, P, S_t\right)$. Define the residual as $\varepsilon_t = I\left[S_t = 0\right]\left[y_t - x_t b_{S_t}\right] + I\left[S_t = 1\right]\left[y_t - x_t b_{S_t}\right]$. This residual is split in to the two variance regimes using V_t with $\sigma^2_{V_t}$ drawn from the inverse Gamma distribution.

The code for this example is shown in figures 20 to 24. Key lines to note are lines 87 to 153 where S_t is drawn with the change in the conditional likelihoods in the Hamilton filter on lines 93 to 103. V_t is drawn on lines 158 to 226. The Hamilton filter is modified on lines 170 to 177. The transition probabilties are drawn on lines 232 to 254. To draw the regression coefficients σ_t is created on line 258. The sample is then split using S_t and b_{S_t} is drawn in the two sub-samples (lines 261 to 274). The residuals are calculated on lines 278. The remaining code draws the variances $\sigma^2_{V_t}$ when $V_t = 0$ and $V_t = 1$ using these residuals.

8. Further reading

- A classic paper on the Bayesian approach to MS models: Chib (1996).



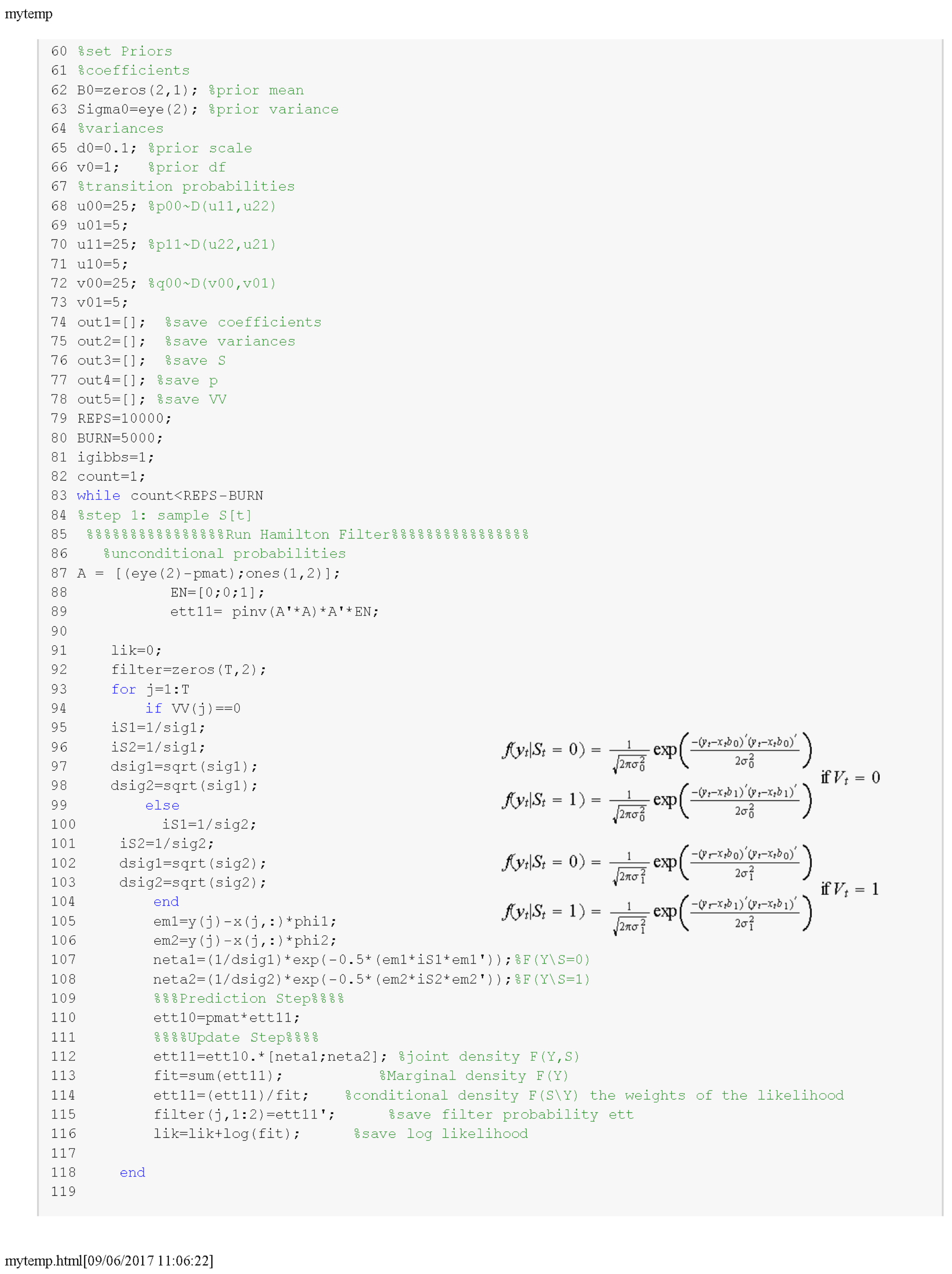
mytemp

```
clear;
addpath('functions');
%generate artificial data
T=500;
B1=0.2;
B2=0.9;
C1=1;
C2=-1;
S1=10;
S2=1;
P=[0.95 0.05;0.05 0.95];
Q=[0.97 0;0.03 1];
strue=zeros(T,2);
strue(1,1)=1;
strue=simS(strue,P);
vtrue=zeros(T,2);
vtrue(1,1)=1;
check=-1;
while check<0
vtrue=simS(vtrue,Q);
if sum(vtrue(:,1))>20
    check=1;
end
end
e=randn(T,1);
Y=zeros(T,1);
X=zeros(T,1);
for i=2:T;
    X(i,:)=Y(i-1,:);
    if strue(i,1)==1
        if vtrue(i,1)==1
    Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S1);
        elseif vtrue(i,2)==1

    Y(i)=[X(i,:) 1]*[B1 C1]'+e(i)*sqrt(S2);
        end
    elseif strue(i,2)==1
   if vtrue(i,1)==1
    Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S1);
    elseif vtrue(i,2)==1
    Y(i)=[X(i,:) 1]*[B2 C2]'+e(i)*sqrt(S2);
   end
    end
end
%data
y=Y;
x=[X ones(T,1)];
%specify starting values
phi1=[0.5;1];   %regime 1 coefficients
phi2=[0.8;-1];   %regime 2 coefficients
sig1=3;       %regime 1 variance
sig2=1;       %regime 2 variance
p=0.95;
q=0.95;
px=0.98;
pmat=[p 1-q;1-p q];
qmat=[px 0; 1-px 1];
VV=zeros(T,1);
ncrit=5; %each regime should have ncrit obs
```

mytemp.html[09/06/2017 11:06:22]

FIGURE 20. Independent switching and structural breaks

- Recent applications by Chris Sims and co-authors: Sims *et al.* (2008).
- Notes by James Hamilton: http://econweb.ucsd.edu/~jhamilto/Econ226_4_slides.pdf



mytemp

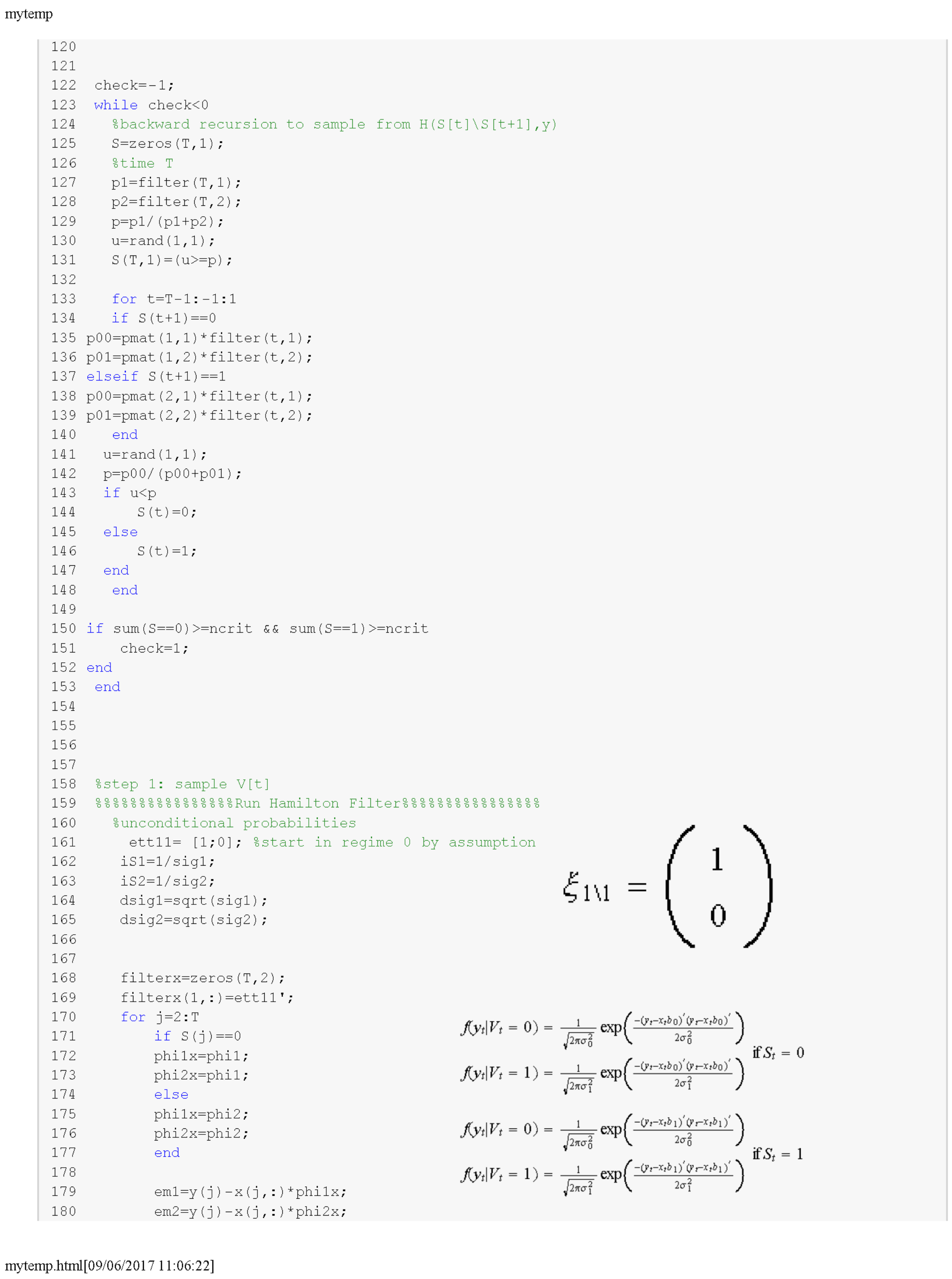
```
%set Priors
%coefficients
B0=zeros(2,1); %prior mean
Sigma0=eye(2); %prior variance
%variances
d0=0.1; %prior scale
v0=1;   %prior df
%transition probabilities
u00=25; %p00~D(u11,u22)
u01=5;
u11=25; %p11~D(u22,u21)
u10=5;
v00=25; %q00~D(v00,v01)
v01=5;
out1=[];  %save coefficients
out2=[];  %save variances
out3=[];  %save S
out4=[]; %save p
out5=[]; %save VV
REPS=10000;
BURN=5000;
igibbs=1;
count=1;
while count<REPS-BURN
%step 1: sample S[t]
 %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
   %unconditional probabilities
A = [(eye(2)-pmat);ones(1,2)];
           EN=[0;0;1];
           ett11= pinv(A'*A)*A'*EN;

    lik=0;
    filter=zeros(T,2);
    for j=1:T
        if VV(j)==0
    iS1=1/sig1;
    iS2=1/sig1;
    dsig1=sqrt(sig1);
    dsig2=sqrt(sig1);
        else
         iS1=1/sig2;
    iS2=1/sig2;
    dsig1=sqrt(sig2);
    dsig2=sqrt(sig2);
        end
        em1=y(j)-x(j,:)*phi1;
        em2=y(j)-x(j,:)*phi2;
        neta1=(1/dsig1)*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
        neta2=(1/dsig2)*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
        %%%Prediction Step%%%%
        ett10=pmat*ett11;
        %%%%Update Step%%%%
        ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
        fit=sum(ett11);           %Marginal density F(Y)
        ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
        filter(j,1:2)=ett11';      %save filter probability ett
        lik=lik+log(fit);      %save log likelihood

    end

```

$$\left.\begin{aligned} f(y_t|S_t = 0) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_0^2}\right) \\ f(y_t|S_t = 1) &= \frac{1}{\sqrt{2\pi\sigma_0^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_0^2}\right)\end{aligned}\right\} \text{if } V_t = 0$$

$$\left.\begin{aligned} f(y_t|S_t = 0) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_0)'(y_t-x_tb_0)'}{2\sigma_1^2}\right) \\ f(y_t|S_t = 1) &= \frac{1}{\sqrt{2\pi\sigma_1^2}}\exp\left(\frac{-(y_t-x_tb_1)'(y_t-x_tb_1)'}{2\sigma_1^2}\right)\end{aligned}\right\} \text{if } V_t = 1$$

mytemp.html[09/06/2017 11:06:22]

Figure 21. Independent switching and structural breaks



mytemp

```


 check=-1;
 while check<0
   %backward recursion to sample from H(S[t]\S[t+1],y)
   S=zeros(T,1);
   %time T
   p1=filter(T,1);
   p2=filter(T,2);
   p=p1/(p1+p2);
   u=rand(1,1);
   S(T,1)=(u>=p);

   for t=T-1:-1:1
   if S(t+1)==0
p00=pmat(1,1)*filter(t,1);
p01=pmat(1,2)*filter(t,2);
elseif S(t+1)==1
p00=pmat(2,1)*filter(t,1);
p01=pmat(2,2)*filter(t,2);
   end
  u=rand(1,1);
  p=p00/(p00+p01);
  if u<p
      S(t)=0;
  else
      S(t)=1;
  end
   end

if sum(S==0)>=ncrit && sum(S==1)>=ncrit
    check=1;
end
 end




 %step 1: sample V[t]
 %%%%%%%%%%%%%%%%Run Hamilton Filter%%%%%%%%%%%%%%%%
   %unconditional probabilities
     ett11= [1;0]; %start in regime 0 by assumption
    iS1=1/sig1;
    iS2=1/sig2;
    dsig1=sqrt(sig1);
    dsig2=sqrt(sig2);


    filterx=zeros(T,2);
    filterx(1,:)=ett11';
    for j=2:T
        if S(j)==0
        phi1x=phi1;
        phi2x=phi1;
        else
        phi1x=phi2;
        phi2x=phi2;
        end

        em1=y(j)-x(j,:)*phi1x;
        em2=y(j)-x(j,:)*phi2x;
```

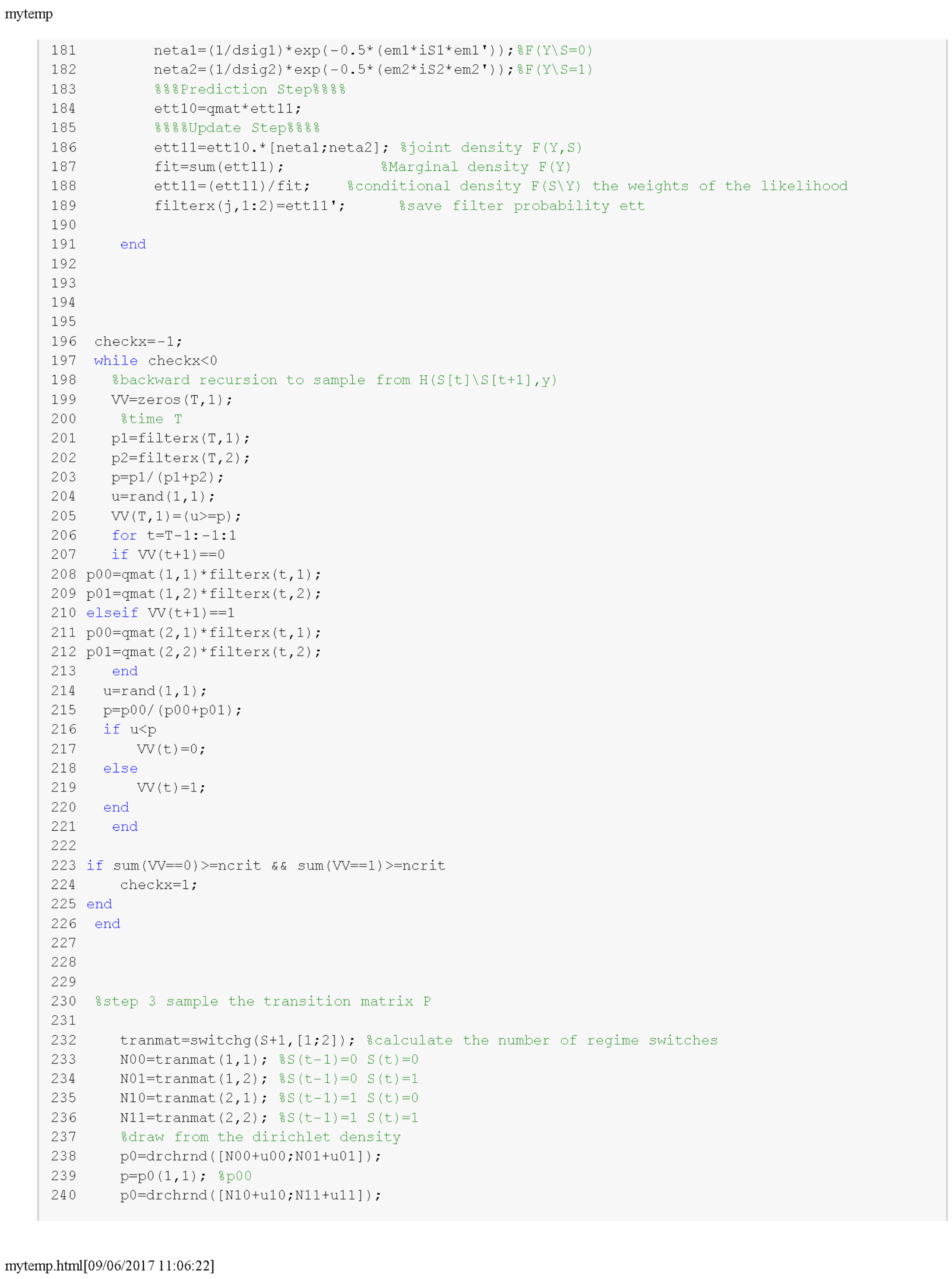
$$\xi_{1\backslash 1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\left.\begin{aligned} f(y_t|V_t = 0) &= \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-(y_t - x_t b_0)'(y_t - x_t b_0)'}{2\sigma_0^2}\right) \\ f(y_t|V_t = 1) &= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-(y_t - x_t b_0)'(y_t - x_t b_0)'}{2\sigma_1^2}\right) \end{aligned}\right\} \text{if } S_t = 0$$

$$\left.\begin{aligned} f(y_t|V_t = 0) &= \frac{1}{\sqrt{2\pi\sigma_0^2}} \exp\left(\frac{-(y_t - x_t b_1)'(y_t - x_t b_1)'}{2\sigma_0^2}\right) \\ f(y_t|V_t = 1) &= \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(\frac{-(y_t - x_t b_1)'(y_t - x_t b_1)'}{2\sigma_1^2}\right) \end{aligned}\right\} \text{if } S_t = 1$$

mytemp.html[09/06/2017 11:06:22]

Figure 22. Independent switching and structural breaks

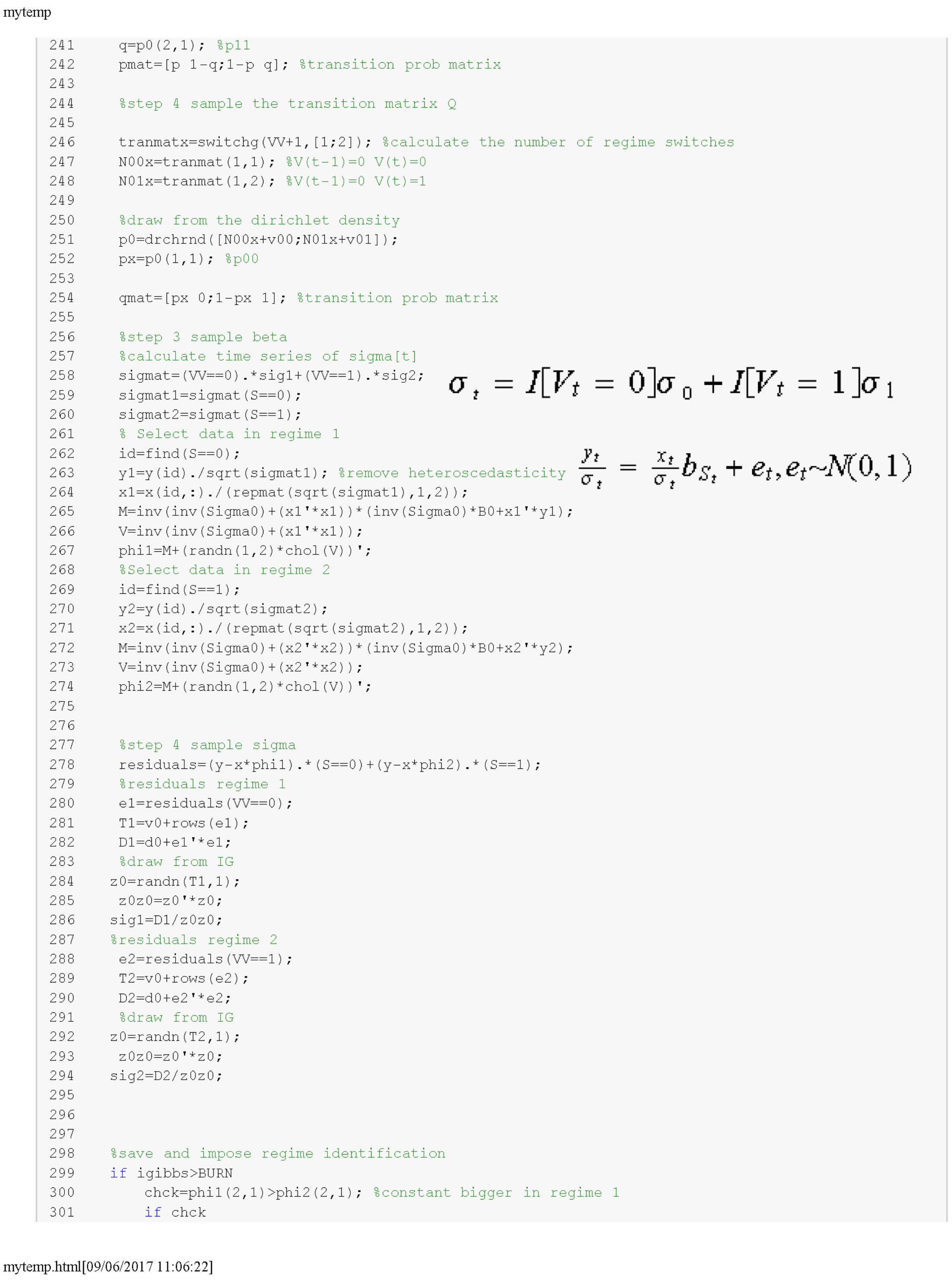


mytemp

```
        neta1=(1/dsig1)*exp(-0.5*(em1*iS1*em1'));%F(Y\S=0)
        neta2=(1/dsig2)*exp(-0.5*(em2*iS2*em2'));%F(Y\S=1)
        %%%Prediction Step%%%%
        ett10=qmat*ett11;
        %%%%Update Step%%%%
        ett11=ett10.*[neta1;neta2]; %joint density F(Y,S)
        fit=sum(ett11);           %Marginal density F(Y)
        ett11=(ett11)/fit;    %conditional density F(S\Y) the weights of the likelihood
        filterx(j,1:2)=ett11';      %save filter probability ett

    end



 checkx=-1;
 while checkx<0
   %backward recursion to sample from H(S[t]\S[t+1],y)
   VV=zeros(T,1);
    %time T
   p1=filterx(T,1);
   p2=filterx(T,2);
   p=p1/(p1+p2);
   u=rand(1,1);
   VV(T,1)=(u>=p);
   for t=T-1:-1:1
   if VV(t+1)==0
p00=qmat(1,1)*filterx(t,1);
p01=qmat(1,2)*filterx(t,2);
elseif VV(t+1)==1
p00=qmat(2,1)*filterx(t,1);
p01=qmat(2,2)*filterx(t,2);
   end
  u=rand(1,1);
  p=p00/(p00+p01);
  if u<p
      VV(t)=0;
  else
      VV(t)=1;
  end
   end

if sum(VV==0)>=ncrit && sum(VV==1)>=ncrit
    checkx=1;
end
 end



 %step 3 sample the transition matrix P

    tranmat=switchg(S+1,[1;2]); %calculate the number of regime switches
    N00=tranmat(1,1); %S(t-1)=0 S(t)=0
    N01=tranmat(1,2); %S(t-1)=0 S(t)=1
    N10=tranmat(2,1); %S(t-1)=1 S(t)=0
    N11=tranmat(2,2); %S(t-1)=1 S(t)=1
    %draw from the dirichlet density
    p0=drchrnd([N00+u00;N01+u01]);
    p=p0(1,1); %p00
    p0=drchrnd([N10+u10;N11+u11]);
```

mytemp.html[09/06/2017 11:06:22]

Figure 23. Independent switching and structural breaks



mytemp

```
 q=p0(2,1); %p11
 pmat=[p 1-q;1-p q]; %transition prob matrix

 %step 4 sample the transition matrix Q

 tranmatx=switchg(VV+1,[1;2]); %calculate the number of regime switches
 N00x=tranmat(1,1); %V(t-1)=0 V(t)=0
 N01x=tranmat(1,2); %V(t-1)=0 V(t)=1

 %draw from the dirichlet density
 p0=drchrnd([N00x+v00;N01x+v01]);
 px=p0(1,1); %p00

 qmat=[px 0;1-px 1]; %transition prob matrix

 %step 3 sample beta
 %calculate time series of sigma[t]
 sigmat=(VV==0).*sig1+(VV==1).*sig2;
```

$$\sigma_t = I[V_t = 0]\sigma_0 + I[V_t = 1]\sigma_1$$

```
 sigmat1=sigmat(S==0);
 sigmat2=sigmat(S==1);
 % Select data in regime 1
 id=find(S==0);
 y1=y(id)./sqrt(sigmat1); %remove heteroscedasticity
```

$$\frac{y_t}{\sigma_t} = \frac{x_t}{\sigma_t} b_{S_t} + e_t, e_t \sim N(0,1)$$

```
 x1=x(id,:)./(repmat(sqrt(sigmat1),1,2));
 M=inv(inv(Sigma0)+(x1'*x1))*(inv(Sigma0)*B0+x1'*y1);
 V=inv(inv(Sigma0)+(x1'*x1));
 phi1=M+(randn(1,2)*chol(V))';
 %Select data in regime 2
 id=find(S==1);
 y2=y(id)./sqrt(sigmat2);
 x2=x(id,:)./(repmat(sqrt(sigmat2),1,2));
 M=inv(inv(Sigma0)+(x2'*x2))*(inv(Sigma0)*B0+x2'*y2);
 V=inv(inv(Sigma0)+(x2'*x2));
 phi2=M+(randn(1,2)*chol(V))';


 %step 4 sample sigma
 residuals=(y-x*phi1).*(S==0)+(y-x*phi2).*(S==1);
 %residuals regime 1
 e1=residuals(VV==0);
 T1=v0+rows(e1);
 D1=d0+e1'*e1;
 %draw from IG
z0=randn(T1,1);
 z0z0=z0'*z0;
sig1=D1/z0z0;
%residuals regime 2
 e2=residuals(VV==1);
 T2=v0+rows(e2);
 D2=d0+e2'*e2;
 %draw from IG
z0=randn(T2,1);
 z0z0=z0'*z0;
sig2=D2/z0z0;



%save and impose regime identification
if igibbs>BURN
    chck=phi1(2,1)>phi2(2,1); %constant bigger in regime 1
    if chck
```

mytemp.html[09/06/2017 11:06:22]

Figure 24. Independent switching and structural breaks.

Part 2

The Metropolis Hastings algorithm

CHAPTER 5

An introduction to the the Metropolis Hastings Algorithm

1. Introduction

The Gibbs sampling algorithm relies on the availability of conditional distributions to be operational. In many cases (of practical relevance) conditional distributions are not available in closed form. An important example of such a situation is the estimation of Dynamic Stochastic General Equilibrium (DSGE) models where the conditional distribution of different parameter blocks is unavailable. In such cases an algorithm more general than the Gibbs sampler is required to approximate the posterior distribution. The Metropolis Hastings algorithm offers such an alternative. This chapter introduces this algorithm and discusses its implementation in Matlab for a number of important cases. The algorithm is applied to DSGE models in the next chapter.

2. The Metropolis Hastings algorithm

In this section we describe the Metropolis Hastings (MH) algorithm in a general setting. We follow that with a number of specific examples and Matlab code in the subsequent sections.

Suppose that we are interested in drawing samples from the following distribution (this is referred to as the target density below)

$$\pi\left(\Phi\right) \tag{2.1}$$

where Φ is a $K\times 1$ vector which represents a set of parameters. $\pi\left(\Phi\right)$ could be a posterior distribution where direct sampling is not possible and the Gibbs sampler is not operational as conditional distributions of different blocks of the parameters Φ are unknown. However, given a value for $\Phi = \Phi^*$ we are able to evaluate the density at Φ^* i.e. calculate $\pi\left(\Phi^*\right)$.

In this situation the MH algorithm can be used to take draws from $\pi\left(\Phi\right)$ using the following steps

Step 1 Specify a *candidate density* $q\left(\Phi^{G+1}|\Phi^{G}\right)$ where G indexes the draw of the parameters Φ. One must be able to draw samples from this density. We discuss the exact specification of $q\left(\Phi^{G+1}|\Phi^{G}\right)$ below.

Step 2 Draw a candidate value of the parameters $\Phi^{G+1}$from the candidate density $q\left(\Phi^{G+1}|\Phi^{G}\right)$.

Step 3 Compute the probability of accepting Φ^{G+1} (denoted by α) using the expression

$$\alpha = \min\left(\frac{\pi\left(\Phi^{G+1}\right)/q\left(\Phi^{G+1}|\Phi^{G}\right)}{\pi\left(\Phi^{G}\right)/q\left(\Phi^{G}|\Phi^{G+1}\right)},1\right) \tag{2.2}$$

The numerator of this expression is the target density evaluated at the new draw of the parameters $\pi\left(\Phi^{G+1}\right)$ divided by the candidate density evaluated at the new draw of the parameters $q\left(\Phi^{G+1}|\Phi^{G}\right)$. The denominator is the same expression evaluated at the previous draw of the parameters.

Step 4 If the acceptance probability α is large enough retain the new draw Φ^{G+1}, otherwise retain the old draw Φ^{G}. How do we decide if α is large enough in practice? We draw a number u from the standard uniform distribution. If $u<\alpha$. accept Φ^{G+1} otherwise keep Φ^{G}.[1].

Step 5 Repeat steps 2 to 4 M times and base inference on the last L draws. In other words, the empirical distribution using the last L draws is an approximation to target density. We discuss convergence of the MH algorithm below.

Note that one can think of the Gibbs sampler as a special case of the MH algorithm–i.e. a situation where the candidate density $q\left(\Phi^{G+1}|\Phi^{G}\right)$ coincides with the target density and the acceptance probability assigned to every draw equals 1.

3. The Random Walk Metropolis Hastings Algorithm

The random walk MH algorithm offers a simple way of specifying the candidate density $q\left(\Phi^{G+1}|\Phi^{G}\right)$ and is therefore widely used in applied work. As the name suggests, the random walk MH algorithm specifies the candidate generating density as a random walk

$$\Phi^{G+1} = \Phi^{G} + e \tag{3.1}$$

[1] This essentially means that we accept the draw with probability α if this experiment is repeated many times. For e.g if $\alpha = 0.1$, and if we 1000 replications we should expect 100 of the 1000 draws to have $u<\alpha$

where $e \sim N(0,\Sigma)$ is a $K \times 1$ vector. Note that $e_t = \Phi^{G+1} - \Phi^G$ is normally distributed. As the normal distribution is symmetric, the density $p\left(\Phi^{G+1} - \Phi^G\right)$ equals $p\left(\Phi^G - \Phi^{G+1}\right)$. In other words, $q\left(\Phi^{G+1}|\Phi^G\right) = q\left(\Phi^G|\Phi^{G+1}\right)$ under this random walk candidate density and the formula for the acceptance probability in equation 2.2 simplifies to

$$\alpha = \min\left(\frac{\pi\left(\Phi^{G+1}\right)}{\pi\left(\Phi^G\right)}, 1\right) \tag{3.2}$$

The random walk MH algorithm, therefore, works in the following steps:

Step 1 Specify a starting value for the parameters Φ denoted by Φ^0 and fix Σ the variance of shock to the random walk candidate generating density.

Step 2 Draw a new value for the parameters Φ^{New} using

$$\Phi^{New} = \Phi^{Old} + e \tag{3.3}$$

where $\Phi^{Old} = \Phi^0$ for the first draw

Step 3 Compute the acceptance probability

$$\alpha = \min\left(\frac{\pi\left(\Phi^{New}\right)}{\pi\left(\Phi^{Old}\right)}, 1\right) \tag{3.4}$$

If $\alpha > u \backsim U(0,1)$, then retain Φ^{New} and set $\Phi^{Old} = \Phi^{New}$, otherwise retain Φ^{Old}.

Step 4 Repeat steps 2 and 3 M times and use the last L draws for inference.

Note that Σ the variance of e_t is set by the researcher. A higher value for Σ could mean a lower rate of acceptances across the MH iterations (i.e. the acceptance rate is defined as the number of accepted MH draws divided by the total number of MH draws) but would mean that the algorithm explores a larger parameter space. In contrast, a lower value for Σ would mean a larger acceptance rate with the algorithm considering a smaller number of possible parameter values. The general recommendation is to choose Σ such that the acceptance rate is between 20% to 40%. We consider the choice of Σ in detail in the examples described below.[2]

3.1. Estimating a non-linear regression via the random walk MH algorithm. As an example, consider the estimation of the following non-linear regression model

$$Y_t = B_1\left(X_t^{B_2}\right) + v_t, v_t \sim N(0,\sigma^2) \tag{3.5}$$

and for the moment assume no prior information is used in estimating B_1, B_2 and σ^2 so the posterior distribution coincides with the likelihood function. Our aim is to draw samples from the marginal posterior distribution of the parameters. As the model is non linear, the results on the conditional distributions of the regression coefficients shown in Chapter 1 do not apply and the MH algorithm is needed. We proceed in the following steps:

Step 1 Set starting values for $\Phi = \{B_1, B_2, \sigma^2\}$. These starting values could be set, for e.g, by estimating a log linearised version of equation 3.5 via OLS. The variance of the candidate generating density Σ can be set as the OLS coefficient covariance matrix $\hat{\Omega}$ times a scaling factor λ i.e $\Sigma = \hat{\Omega} \times \lambda$.Note that $\hat{\Omega}$ provides a rough guess of how volatile each parameter is. The scaling factor lets the researcher control the acceptance rate (a higher value for λ would mean a lower acceptance rate). Note that in this simple model the choice of starting values may not be very important and the algorithm would probably converge rapidly to the posterior. However, in the more complicated (and realistic) models considered below this choice can be quite important.

Step 2 Draw a new set of parameters from the random walk candidate density

$$\Phi^{New} = \Phi^{Old} + e \tag{3.6}$$

Step 3 Compute the acceptance probability $\alpha = \min\left(\frac{\pi\left(\Phi^{New}\right)}{\pi\left(\Phi^{Old}\right)}, 1\right)$. Note that the target density $\pi(.)$ is the likelihood function in this example. The log likelihood function for this regression model is given by

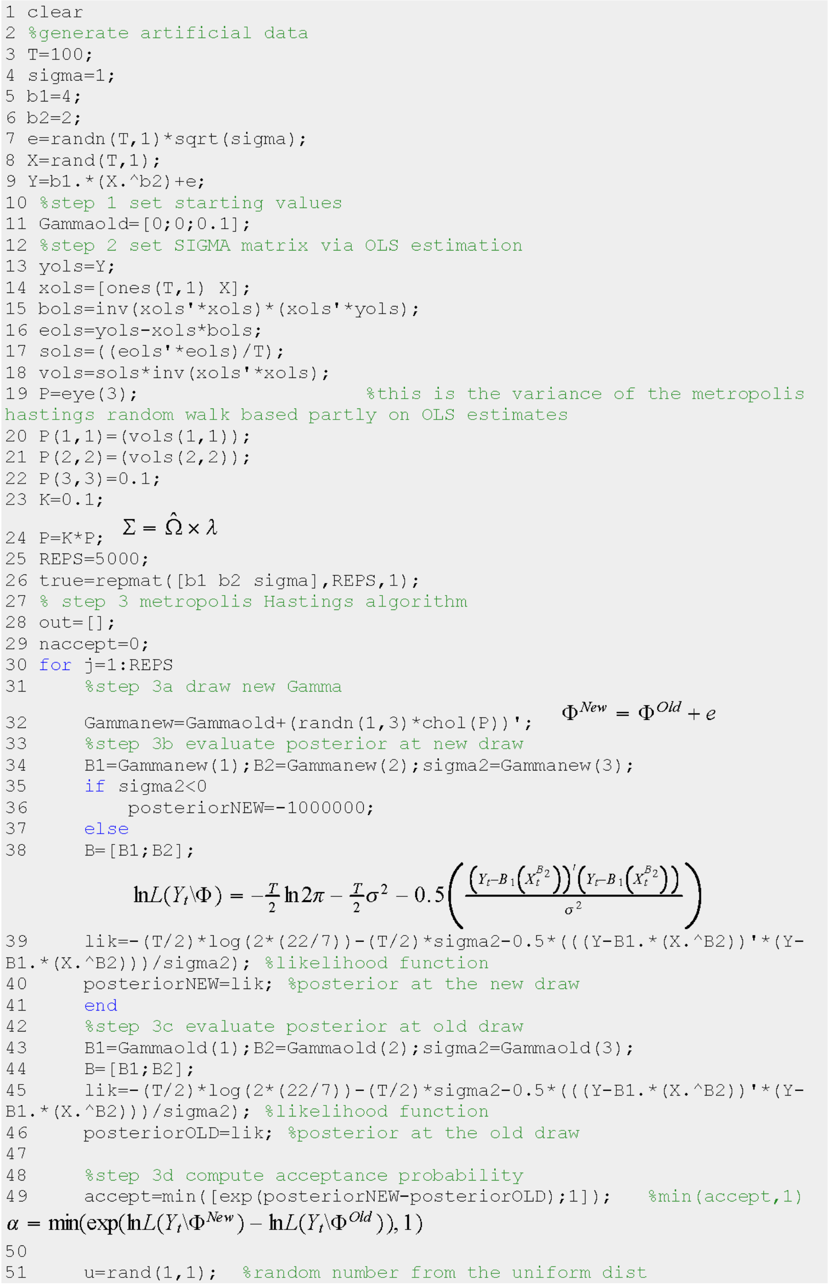
$$\ln L\left(Y_t|\Phi\right) = -\frac{T}{2}\ln 2\pi - \frac{T}{2}\sigma^2 - 0.5\left(\frac{\left(Y_t - B_1\left(X_t^{B_2}\right)\right)'\left(Y_t - B_1\left(X_t^{B_2}\right)\right)}{\sigma^2}\right) \tag{3.7}$$

Therefore the acceptance probability is simply the likelihood ratio

$$\alpha = \min\left(\exp\left(\ln L\left(Y_t|\Phi^{New}\right) - \ln L\left(Y_t|\Phi^{Old}\right)\right), 1\right) \tag{3.8}$$

where $\ln L\left(Y_t|\Phi^{New}\right)$ is the log likelihood evaluated at the new draw of B_1, B_2, σ^2 and $\ln L\left(Y_t|\Phi^{Old}\right)$ is the log likelihood at the old draw. If $\alpha < u \sim U(0,1)$ we retain the new draw and set $\Phi^{Old} = \Phi^{New}$.

Step 4 Repeat steps 2 and 3 M times. The last L draws of B_1, B_2, σ^2 provide an approximation to the marginal posterior distributions.



```
1 clear
2 %generate artificial data
3 T=100;
4 sigma=1;
5 b1=4;
6 b2=2;
7 e=randn(T,1)*sqrt(sigma);
8 X=rand(T,1);
9 Y=b1.*(X.^b2)+e;
10 %step 1 set starting values
11 Gammaold=[0;0;0.1];
12 %step 2 set SIGMA matrix via OLS estimation
13 yols=Y;
14 xols=[ones(T,1) X];
15 bols=inv(xols'*xols)*(xols'*yols);
16 eols=yols-xols*bols;
17 sols=((eols'*eols)/T);
18 vols=sols*inv(xols'*xols);
19 P=eye(3);                    %this is the variance of the metropolis
hastings random walk based partly on OLS estimates
20 P(1,1)=(vols(1,1));
21 P(2,2)=(vols(2,2));
22 P(3,3)=0.1;
23 K=0.1;
```

$$\Sigma = \hat{\Omega} \times \lambda$$

```
24 P=K*P;
25 REPS=5000;
26 true=repmat([b1 b2 sigma],REPS,1);
27 % step 3 metropolis Hastings algorithm
28 out=[];
29 naccept=0;
30 for j=1:REPS
31     %step 3a draw new Gamma
```

$$\Phi^{New} = \Phi^{Old} + e$$

```
32     Gammanew=Gammaold+(randn(1,3)*chol(P))';
33     %step 3b evaluate posterior at new draw
34     B1=Gammanew(1);B2=Gammanew(2);sigma2=Gammanew(3);
35     if sigma2<0
36         posteriorNEW=-1000000;
37     else
38     B=[B1;B2];
```

$$\ln L(Y_t \backslash \Phi) = -\frac{T}{2}\ln 2\pi - \frac{T}{2}\sigma^2 - 0.5\left(\frac{\left(Y_t - B_1\left(X_t^{B_2}\right)\right)'\left(Y_t - B_1\left(X_t^{B_2}\right)\right)}{\sigma^2}\right)$$

```
39     lik=-(T/2)*log(2*(22/7))-(T/2)*sigma2-0.5*(((Y-B1.*(X.^B2))'*(Y-
B1.*(X.^B2)))/sigma2); %likelihood function
40     posteriorNEW=lik; %posterior at the new draw
41     end
42     %step 3c evaluate posterior at old draw
43     B1=Gammaold(1);B2=Gammaold(2);sigma2=Gammaold(3);
44     B=[B1;B2];
45     lik=-(T/2)*log(2*(22/7))-(T/2)*sigma2-0.5*(((Y-B1.*(X.^B2))'*(Y-
B1.*(X.^B2)))/sigma2); %likelihood function
46     posteriorOLD=lik; %posterior at the old draw
47
48     %step 3d compute acceptance probability
49     accept=min([exp(posteriorNEW-posteriorOLD);1]);   %min(accept,1)
```

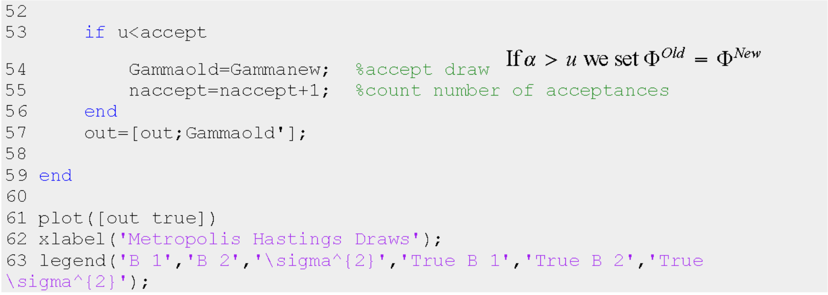
$$\alpha = \min(\exp(\ln L(Y_t \backslash \Phi^{New}) - \ln L(Y_t \backslash \Phi^{Old})), 1)$$

```
50
51     u=rand(1,1);  %random number from the uniform dist
```

FIGURE 1. Matlab code for example1

Figures 1 and 2 show the matlab code for this example (example1.m). Lines 3 to 9 generate artificial data for the non-linear regression model assuming that $B_1 = 4, B_2 = 2, \sigma^2 = 1$. Line 11 sets the starting values for these parameters arbitrarily. Lines 20 to 22 set the variance of the random walk candidate density as $\Sigma = \begin{pmatrix} \sigma^{B_1} & 0 & 0 \\ 0 & \sigma^{B_2} & 0 \\ 0 & 0 & 0.1 \end{pmatrix}$ ×scaling factor where σ^{B_1} and σ^{B_2} are OLS estimates of the variance of B_1 and B_2. Line 29 sets the variable naccept which will count the number of accepted draws. Hence the acceptance rate is naccept/REPS. Line 30 starts the loop for the

[2] See Chib and Ramamurthy (2010) for a more efficient version of the basic algorithm described above.



```
52
53     if u<accept
54         Gammaold=Gammanew;  %accept draw
55         naccept=naccept+1;  %count number of acceptances
56     end
57     out=[out;Gammaold'];
58
59 end
60
61 plot([out true])
62 xlabel('Metropolis Hastings Draws');
63 legend('B 1','B 2','\sigma^{2}','True B 1','True B 2','True
\sigma^{2}');
```

If $\alpha > u$ we set $\Phi^{Old} = \Phi^{New}$

Published with MATLAB® 7.9

Figure 2. Matlab code for example1 (continued)

MH algorithm. Line 32 draws the new value of the parameters from the random walk candidate density. Note that there is nothing intrinsic in this step that stops the new value of σ^2 from being less than zero. Therefore lines 35 to 37 set the value of the log likelihood to a very small number if a negative σ^2 is drawn thus ensuring that this draw is not going to be accepted. Alternatively one can set the acceptance probability to 0 when a negative value for σ^2 is drawn. Lines 44 to 46 calculate the log likelihood at the old draw. Line 49 calculates the acceptance probability. Line 53 checks if the acceptance probability is bigger than u a number from the standard uniform distribution. If this is the case we retain the new draw of the parameters.Figure 3 shows all the draws of the model parameters. The algorithm is close to the true values of these parameters after a few hundred draws.

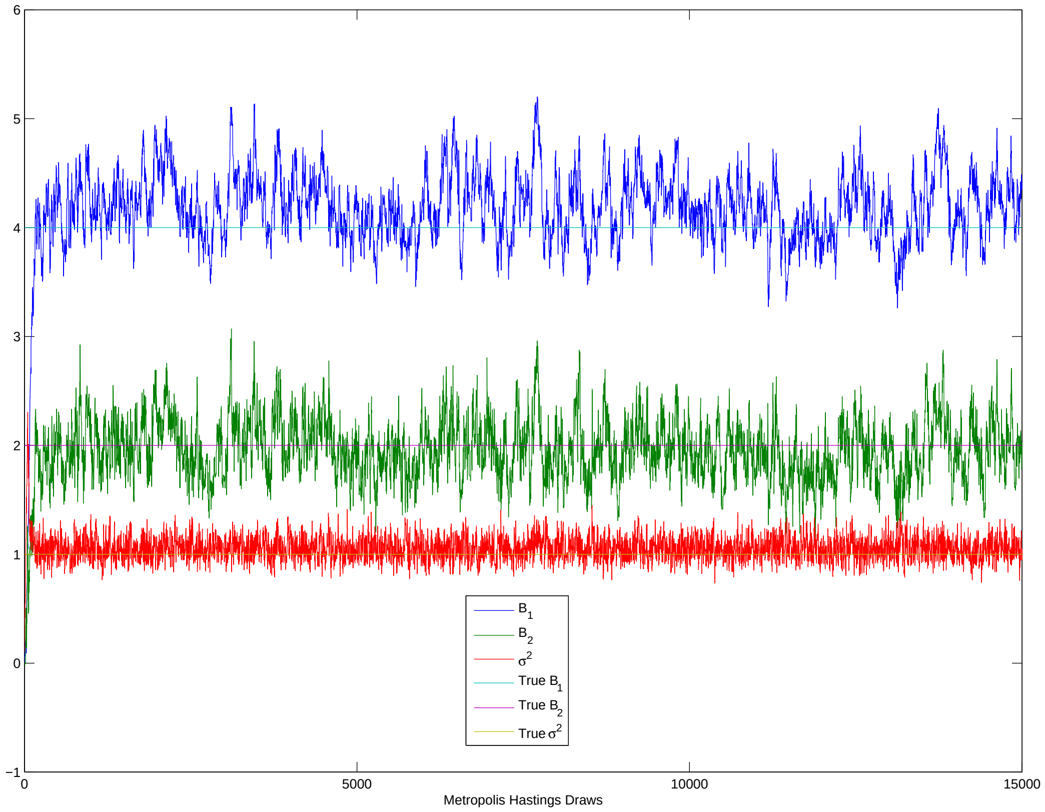


FIGURE 3. Draws of B_1, B_2, σ^2 using the MH algorithm in example 1

3.2. Estimating a non-linear regression via the random walk MH algorithm (incorporating prior distributions). We consider the same non-linear regression model examined in the previous section but now incorporate prior distribution for the regression parameters. We assume that the regression coefficients $B = \{B_1, B_2\}$ have a normal prior $p(B) \sim N(B_0, \Sigma_0)$. For convenience, we set a prior for the precision, the reciprocal of the variance. The Gamma prior $p\left(1/\sigma^2\right)$ with a prior scale parameter $\frac{\sigma^0}{2}$ and degrees of freedom $\frac{V^0}{2}$. The random walk MH algorithm now consists of the following steps:

algorithm is needed. We proceed in the following steps:

Step 1 Set the parameters of the prior distributions $p(B)$ and $p\left(1/\sigma^2\right)$. Set starting values for $\Phi = \{B_1, B_2, \sigma^2\}$. Finally set the variance of the candidate generating density Σ.

Step 2 Draw a new set of parameters from the random walk candidate density

$$\Phi^{New} = \Phi^{Old} + e \tag{3.9}$$

Step 3 Compute the acceptance probability $\alpha = \min\left(\frac{\pi(\Phi^{New})}{\pi(\Phi^{Old})}, 1\right)$. Note that the target density $\pi(.)$ is the *posterior density* in this example as we have prior distributions for our parameters. Recall from chapter 1 that the Bayes law states that he posterior distribution is proportional to the likelihood times the prior. Therefore we need to evaluate the likelihood and the prior distributions at the drawn value of the parameters and multiply them together. The log likelihood function for this regression model is given by

$$\ln L(Y_t|\Phi) = -\frac{T}{2}\ln 2\pi - \frac{T}{2}\sigma^2 - 0.5\left(\frac{\left(Y_t - B_1\left(X_t^{B_2}\right)\right)'\left(Y_t - B_1\left(X_t^{B_2}\right)\right)}{\sigma^2}\right) \tag{3.10}$$

The prior density for the regression coefficients is just a normal density given by

$$P(B) = (2\pi)^{-K/2}|\Sigma_0|^{-\frac{1}{2}}\exp\left[-0.5(B - B_0)'\Sigma_0^{-1}(B - B_0)\right] \tag{3.11}$$

Note that this is evaluated at the new draw of the regression coefficients. If the new draw is very far from the prior mean B_0 and the prior is tight (the diagonal elements of Σ_0 are small) then $P(B)$ will evaluate to a small number.

Similarly, the log prior density for $1/\sigma^2$ is a Gamma distribution with a density function given by

$$P\left(1/\sigma^2\right) = C^* {\frac{1}{\sigma^2}}^{\frac{V_0}{2}-1} \exp\left(\frac{-\sigma_0}{2\sigma^2}\right) \tag{3.12}$$

where $C^* = \frac{1}{\Gamma\left(\frac{V_0}{2}\right)\left(\frac{2}{\sigma^0}\right)^{\frac{V_0}{2}}}$ and $\Gamma\left(.\right)$ denotes the Gamma function. The log posterior is given by

$$\ln H\left(\Phi|Y_t\right) \propto \ln L\left(Y_t|\Phi\right) + \ln P\left(B\right) + \ln P\left(1/\sigma^2\right)$$

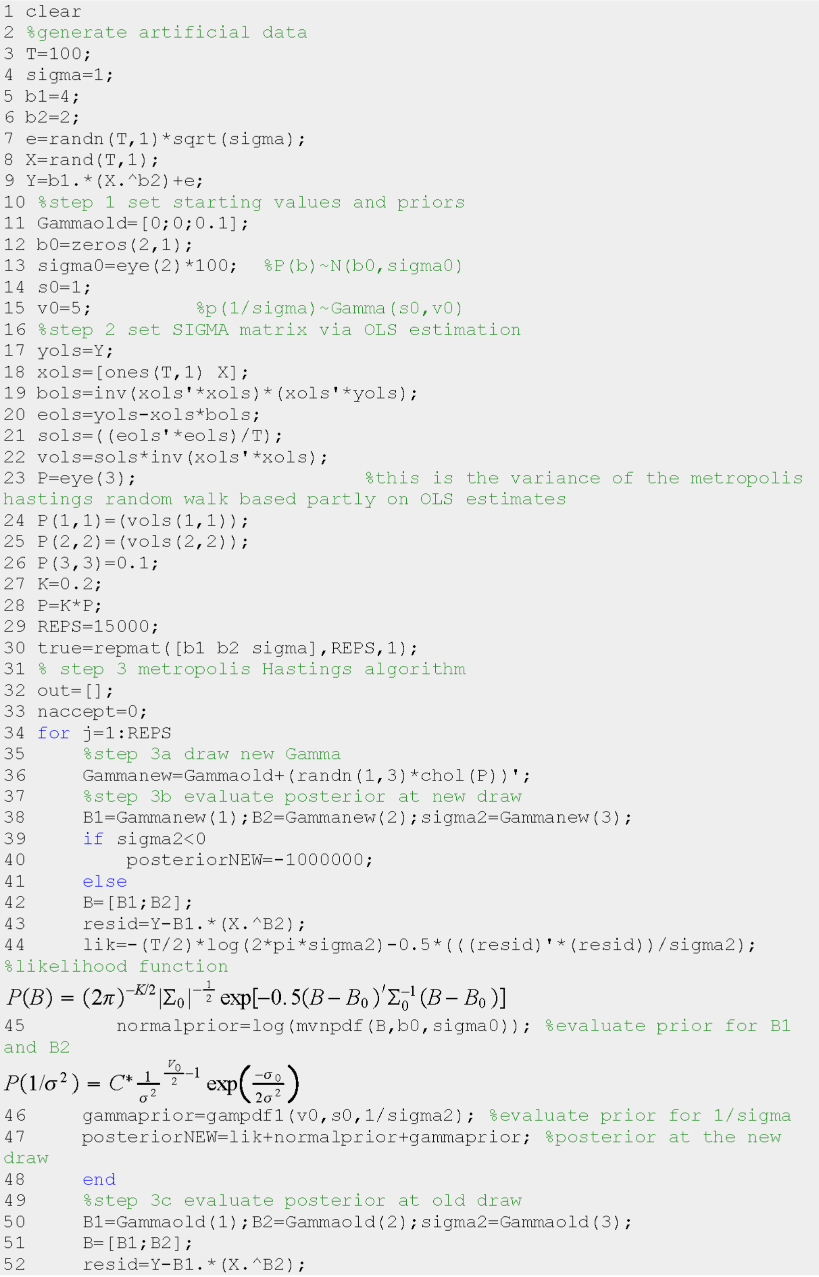
Therefore the acceptance probability is simply the likelihood ratio

$$\alpha = \min\left(\exp\left(\ln H\left(\Phi^{New}|Y_t\right) - \ln H\left(\Phi^{Old}|Y_t\right)\right), 1\right) \tag{3.13}$$

where $\ln H\left(\Phi^{New}|Y_t\right)$ is the log posterior evaluated at the new draw of B_1, B_2, σ^2 and $\ln H\left(\Phi^{Old}|Y_t\right)$ is the log posterior evaluated at the old draw. If $\alpha < u \sim U(0,1)$ we retain the new draw and set $\Phi^{Old} = \Phi^{New}$.

Step 4 Repeat steps 2 and 3 M times. The last L draws of B_1, B_2, σ^2 provide an approximation to the marginal posterior distributions.

Figures 4 and 5 show the code for this example (example2.m). Relative to the previous example there are only two changes. First on lines 12 to 15 we set the parameters of the prior distributions for B and $1/\sigma^2$. Second, line 45 evaluates the log prior density for B at the new draw. Similarly, line 46 evaluates the log prior density for $1/\sigma^2$ at the new draw. The log posterior at the new draw is calculated on line 47. Lines 50 to 55 calculate the log posterior at the old draw of the parameters. The remaining code is identical to the previous example.



```
clear
%generate artificial data
T=100;
sigma=1;
b1=4;
b2=2;
e=randn(T,1)*sqrt(sigma);
X=rand(T,1);
Y=b1.*(X.^b2)+e;
%step 1 set starting values and priors
Gammaold=[0;0;0.1];
b0=zeros(2,1);
sigma0=eye(2)*100;  %P(b)~N(b0,sigma0)
s0=1;
v0=5;         %p(1/sigma)~Gamma(s0,v0)
%step 2 set SIGMA matrix via OLS estimation
yols=Y;
xols=[ones(T,1) X];
bols=inv(xols'*xols)*(xols'*yols);
eols=yols-xols*bols;
sols=((eols'*eols)/T);
vols=sols*inv(xols'*xols);
P=eye(3);                    %this is the variance of the metropolis hastings random walk based partly on OLS estimates
P(1,1)=(vols(1,1));
P(2,2)=(vols(2,2));
P(3,3)=0.1;
K=0.2;
P=K*P;
REPS=15000;
true=repmat([b1 b2 sigma],REPS,1);
% step 3 metropolis Hastings algorithm
out=[];
naccept=0;
for j=1:REPS
    %step 3a draw new Gamma
    Gammanew=Gammaold+(randn(1,3)*chol(P))';
    %step 3b evaluate posterior at new draw
    B1=Gammanew(1);B2=Gammanew(2);sigma2=Gammanew(3);
    if sigma2<0
        posteriorNEW=-1000000;
    else
    B=[B1;B2];
    resid=Y-B1.*(X.^B2);
    lik=-(T/2)*log(2*pi*sigma2)-0.5*(((resid)'*(resid))/sigma2); %likelihood function
```

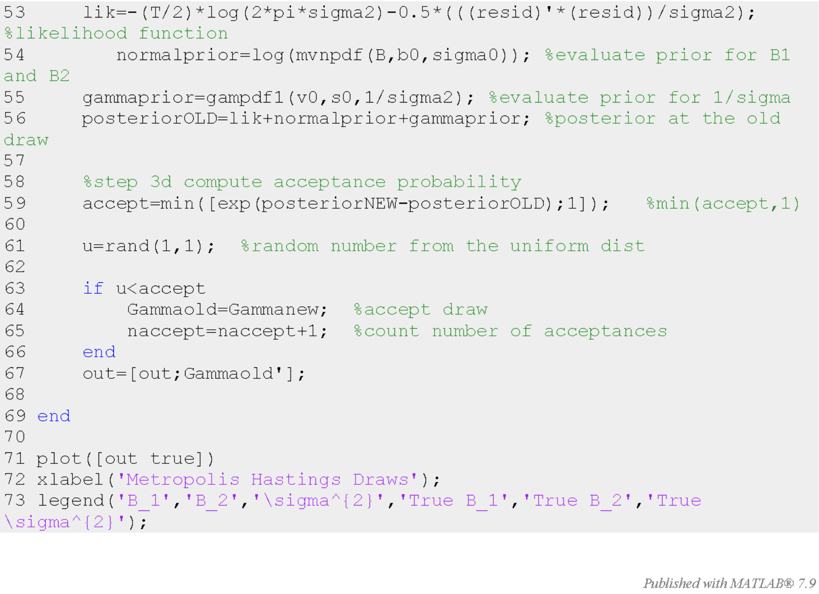
$$P(B) = (2\pi)^{-K/2}|\Sigma_0|^{-\frac{1}{2}}\exp[-0.5(B-B_0)'\Sigma_0^{-1}(B-B_0)]$$

```
       normalprior=log(mvnpdf(B,b0,sigma0)); %evaluate prior for B1 and B2
```

$$P(1/\sigma^2) = C^*\frac{1}{\sigma^2}^{\frac{V_0}{2}-1}\exp\left(\frac{-\sigma_0}{2\sigma^2}\right)$$

```
    gammaprior=gampdf1(v0,s0,1/sigma2); %evaluate prior for 1/sigma
    posteriorNEW=lik+normalprior+gammaprior; %posterior at the new draw
    end
    %step 3c evaluate posterior at old draw
    B1=Gammaold(1);B2=Gammaold(2);sigma2=Gammaold(3);
    B=[B1;B2];
    resid=Y-B1.*(X.^B2);
```

FIGURE 4. Matlab code for example 2



```
    lik=-(T/2)*log(2*pi*sigma2)-0.5*(((resid)'*(resid))/sigma2); %likelihood function
       normalprior=log(mvnpdf(B,b0,sigma0)); %evaluate prior for B1 and B2
    gammaprior=gampdf1(v0,s0,1/sigma2); %evaluate prior for 1/sigma
    posteriorOLD=lik+normalprior+gammaprior; %posterior at the old draw

    %step 3d compute acceptance probability
    accept=min([exp(posteriorNEW-posteriorOLD);1]);   %min(accept,1)

    u=rand(1,1);  %random number from the uniform dist

    if u<accept
        Gammaold=Gammanew;  %accept draw
        naccept=naccept+1;  %count number of acceptances
    end
    out=[out;Gammaold'];

end

plot([out true])
xlabel('Metropolis Hastings Draws');
legend('B_1','B_2','\sigma^{2}','True B_1','True B_2','True \sigma^{2}');
```

Published with MATLAB® 7.9

FIGURE 5. Matlab code for example 2 (continued)

3.3. The random walk MH algorithm for a state space model. In this section we consider the estimation of a regression with time-varying parameters using the MH algorithm. Note that Gibbs sampling is feasible for this model. Our reason for using the MH algorithm is related to the fact the steps involved in dealing with this model are very similar to those required when estimating a DSGE model. In particular, the choice of starting values is no longer trivial.

The model we consider has the following state space representation

$$\begin{aligned} y_t &= c_t + b_t x_t + v_t, v_t \tilde{} N(0,R) \qquad (3.14) \\ \begin{pmatrix} c_t \\ b_t \end{pmatrix} &= \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} + \begin{pmatrix} F_1 & 0 \\ 0 & F_2 \end{pmatrix} \begin{pmatrix} c_{t-1} \\ b_{t-1} \end{pmatrix} + \begin{pmatrix} e_{1t} \\ e_{2t} \end{pmatrix}, \\ & \begin{pmatrix} e_{1t} \\ e_{2t} \end{pmatrix} \tilde{} N\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix} \right) \end{aligned}$$

The random walk MH algorithm for this model works exactly as before. At each iteration we calculate the log posterior for the model at the old and the new draw of the parameters $\Phi = \{\mu_1, \mu_2, F_1, F_2, R, Q_1, Q_2\}$. Calculation of the posterior involves evaluating the prior distributions and the log likelihood of the model. Note that the likelihood function of this state space model is evaluated using the Kalman filter. As discussed in Hamilton (1994) (page 385) if the shocks to the state space model (v_t, e_{1t}, e_{2t}) are distributed normally, then the density of the data $f(y_t|x_t)$ is given as

$$f(y_t|x_t) = (2\pi)^{-1/2} \left|f_{t|t-1}\right|^{-1/2} \times \exp\left(-0.5\eta'_{t|t-1} f^{-1}_{t|t-1} \eta_{t|t-1}\right) \qquad (3.15)$$

for $t = 1....T$ with the log likelihood of the model given by

$$\ln f(y_t|\Phi) = \sum_{t=1}^{T} \ln f(y_t|x_t) \qquad (3.16)$$

Here $\eta_{t|t-1}$ is the prediction error from the prediction step of the Kalman filter and $f_{t|t-1}$ is the variance of the prediction error (see Chapter 3).

Figures 6 and 7 show a matlab function (likelihoodTVP.m) which uses the Kalman filter to calculate the likelihood for this model and will be used in the main code discussed below. Line 4 checks if the variances (stored as the last three elements of theta) are positive and $F_1$and F_2 do not sum to a number greater than 1 (this is a rough way to check for stability). Lines 5 to 7 form the matrix $\begin{pmatrix} F_1 & 0 \\ 0 & F_2 \end{pmatrix}$ while lines 8 to 10 form the matrix $\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$. Line 13 specifies the matrix R while lines 14 to 16 specify the matrix $\begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}$. The Kalman filter recursions on lines 20 to 39 are as described in Chapter 3. Line 40 uses the prediction error and the variance of the prediction error to calculate $f(y_t|x_t) = (2\pi)^{-1/2} \left|f_{t|t-1}\right|^{-1/2} \times \exp\left(-0.5\eta'_{t|t-1} f^{-1}_{t|t-1} \eta_{t|t-1}\right)$. Line 42 adds this for each observation (if there are no numerical problems). Line 47 returns the negative of the likelihood function (we are going to minimise this below).

The MH algorithm for this model is given by the following steps:

Step 1 Set priors for the coefficients and variances of the state space model. We assume that μ_1, μ_2, F_1, F_2 have a normal prior while the reciprocal of R, Q_1, Q_2 have a Gamma prior.

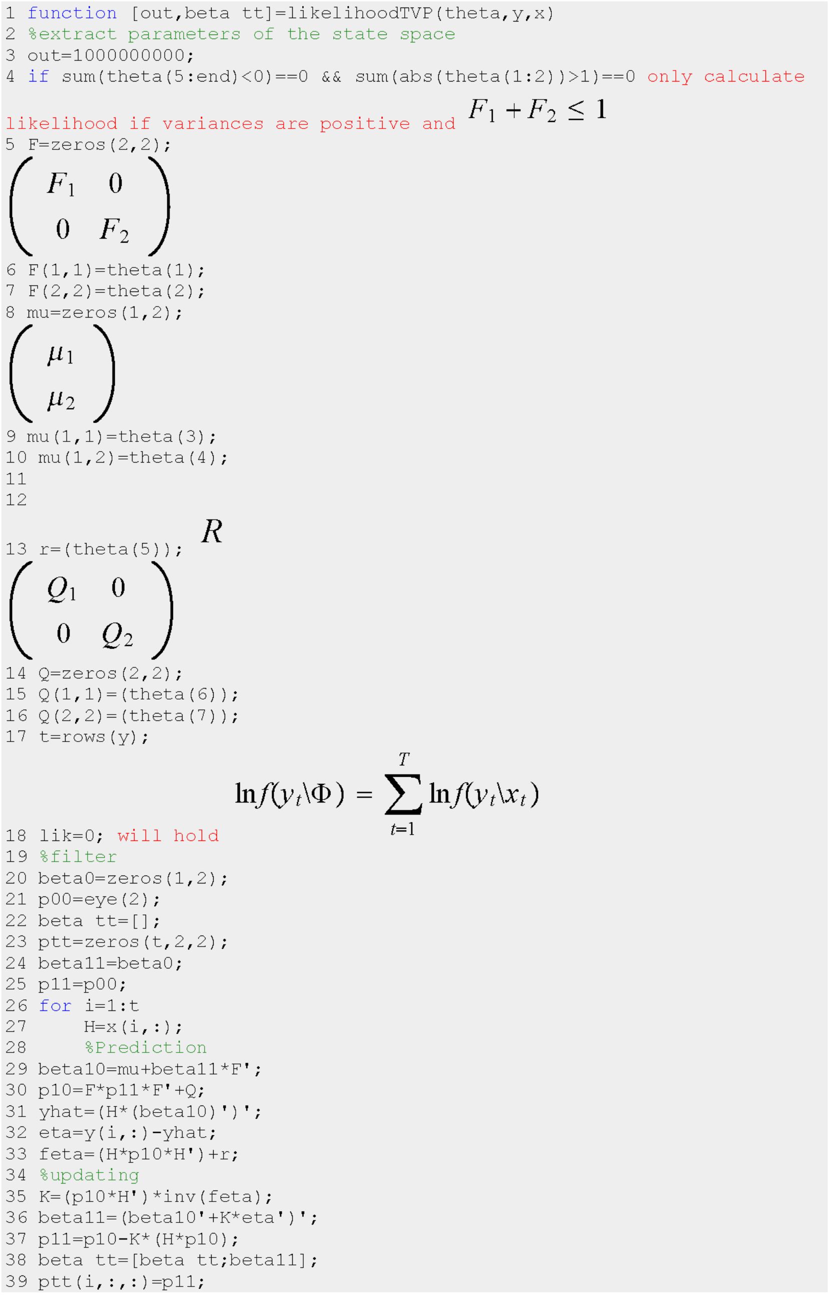
Step 2 Set a starting value for the parameters $\Phi = \{\mu_1, \mu_2, F_1, F_2, R, Q_1, Q_2\}$ and the variance of the shock to the random walk candidate generating density. We set the starting value for Φ as the estimate Φ^{ML} by numerically maximising the log posterior. The mode of the posterior provides a reasonable point to the start the MH algorithm and implies that fewer iterations may be required for the algorithm to converge.[3]The estimate of the covariance of Φ^{ML} can be used to set the variance of the random walk candidate density. Note that the covariance of Φ^{ML} is given by the inverse of the hessian of the log posterior with respect to the model parameters. Denoting this estimated variance by $\hat{\Omega}$, the variance of the shock to the candidate generating density is set as $\Sigma = \hat{\Omega} \times \lambda$ where λ is a scaling factor chosen by the researchers such that the acceptance rate is between 20% and 40%.

Step 3 Draw a new set of parameters from the random walk candidate density

$$\Phi^{New} = \Phi^{Old} + e \qquad (3.17)$$

Step 4 Compute the acceptance probability $\alpha = \min\left(\frac{\pi(\Phi^{New})}{\pi(\Phi^{Old})}, 1\right)$. As in the previous example the target density is the posterior distribution. The log of the posterior distribution is calculated as the sum of the log likelihood and the sum of the log priors. As described above, the log likelihood is calculated by using the Kalman filter. If $\alpha > u \sim U(0,1)$ then we keep Φ^{New} otherwise we retain the old draw.

[3] Note also that if the posterior is multi-modal (which may be the case for complicated models) the numerical maximum will be a rough approximation to the posterior mode.



```
function [out,beta tt]=likelihoodTVP(theta,y,x)
%extract parameters of the state space
out=1000000000;
if sum(theta(5:end)<0)==0 && sum(abs(theta(1:2))>1)==0 only calculate
likelihood if variances are positive and
```

$F_1 + F_2 \leq 1$

```
F=zeros(2,2);
```

$$\begin{pmatrix} F_1 & 0 \\ 0 & F_2 \end{pmatrix}$$

```
F(1,1)=theta(1);
F(2,2)=theta(2);
mu=zeros(1,2);
```

$$\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$$

```
mu(1,1)=theta(3);
mu(1,2)=theta(4);


r=(theta(5));
```

R

$$\begin{pmatrix} Q_1 & 0 \\ 0 & Q_2 \end{pmatrix}$$

```
Q=zeros(2,2);
Q(1,1)=(theta(6));
Q(2,2)=(theta(7));
t=rows(y);
```

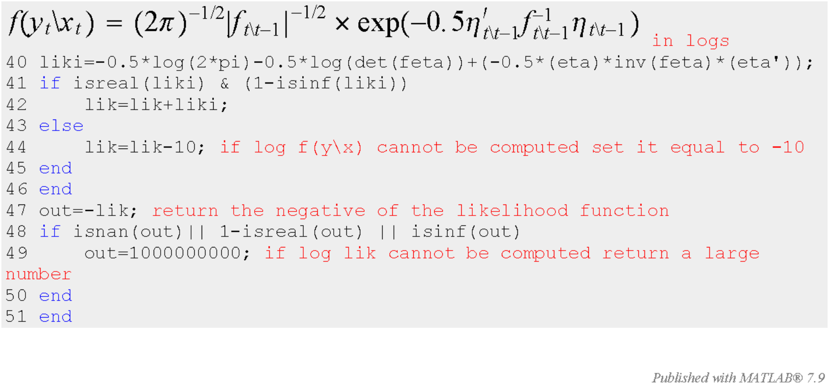
$$\ln f(y_t \backslash \Phi) = \sum_{t=1}^{T} \ln f(y_t \backslash x_t)$$

```
lik=0; will hold
%filter
beta0=zeros(1,2);
p00=eye(2);
beta tt=[];
ptt=zeros(t,2,2);
beta11=beta0;
p11=p00;
for i=1:t
    H=x(i,:);
    %Prediction
beta10=mu+beta11*F';
p10=F*p11*F'+Q;
yhat=(H*(beta10)')';
eta=y(i,:)-yhat;
feta=(H*p10*H')+r;
%updating
K=(p10*H')*inv(feta);
beta11=(beta10'+K*eta')';
p11=p10-K*(H*p10);
beta tt=[beta tt;beta11];
ptt(i,:,:)=p11;
```

FIGURE 6. The log likelihood for the time-varying parameter model in Matlab

Step 5 Repeat steps 3 and 4 M times. The last L draws of Φ provide an approximation to the marginal posterior distributions.

Figures 8, 9 and 10 show the code for the MH algorithm for this model. Line 2 of the code adds the optimization software *csminwel* written by Chris Sims and freely available at http://sims.princeton.edu/yftp/optimize/mfiles/. This matlab function minimises a user supplied function. Lines 5 to 23 create artificial data for the state space model assuming that $\mu_1 = 0.1, \mu_2 = -0.1, F_1 = 0.95, F_2 = 0.95, R = 2, Q_1 = 0.1, Q_2 = 0.1$. Lines 25 to 36 set the parameters for the prior distributions. Lines 37 to 39 maximise the log posterior of the model using *csminwel.* Line 39 called csminwel using the code:



$f(y_t\backslash x_t) = (2\pi)^{-1/2}|f_{t\backslash t-1}|^{-1/2} \times \exp(-0.5\eta'_{t\backslash t-1}f^{-1}_{t\backslash t-1}\eta_{t\backslash t-1})$ in logs

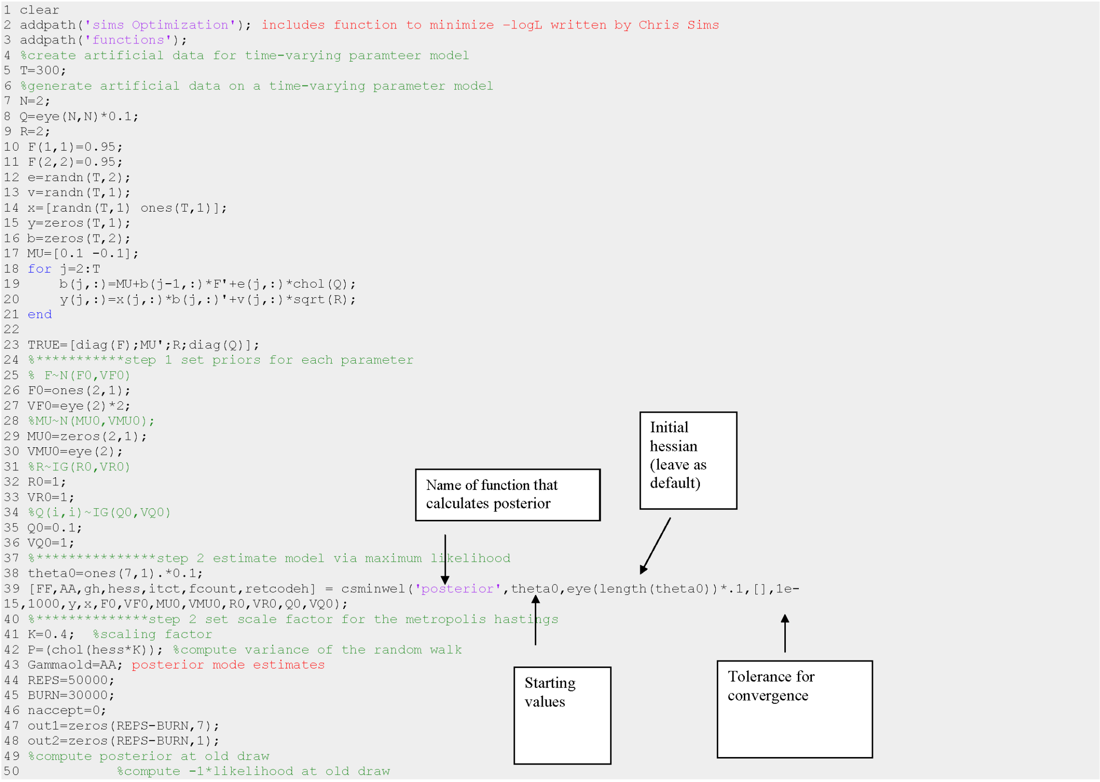
```
liki=-0.5*log(2*pi)-0.5*log(det(feta))+(-0.5*(eta)*inv(feta)*(eta'));
if isreal(liki) & (1-isinf(liki))
    lik=lik+liki;
else
    lik=lik-10; if log f(y\x) cannot be computed set it equal to -10
end
end
out=-lik; return the negative of the likelihood function
if isnan(out)|| 1-isreal(out) || isinf(out)
    out=1000000000; if log lik cannot be computed return a large
number
end
end
```

Published with MATLAB® 7.9

FIGURE 7. The log likelihood for the time-varying parameter model in Matlab (continued)

[FF,AA,gh,hess,itct,fcount,retcodeh] =
csminwel(‘posterior’,theta0,eye(length(theta0))*.1,[],1e-15,1000,y,x,F0,VF0,MU0,VMU0,R0,VR0,Q0,VQ0);

The inputs to the function are (1) the name of the function that calculates the log posterior. This is called posterior.m in our example. Note that this example evaluates the log likelihood using likelihoodTVP.m. The function then evaluates the log prior for each parameter. The sum of these is the log joint prior. The sum of the log joint prior and the log likelihood is the log posterior. Note that posterior.m returns the negative of the log posterior. Therefore csminwel minimises the minimum of the negative log posterior which is equivalent to maximising the log posterior. (2) the initial values of the model parameters theta0. (3) the initial hessian matrix which can be left as default. (4) a



```
clear
addpath('sims Optimization'); includes function to minimize -logL written by Chris Sims
addpath('functions');
%create artificial data for time-varying paramteer model
T=300;
%generate artificial data on a time-varying parameter model
N=2;
Q=eye(N,N)*0.1;
R=2;
F(1,1)=0.95;
F(2,2)=0.95;
e=randn(T,2);
v=randn(T,1);
x=[randn(T,1) ones(T,1)];
y=zeros(T,1);
b=zeros(T,2);
MU=[0.1 -0.1];
for j=2:T
    b(j,:)=MU+b(j-1,:)*F'+e(j,:)*chol(Q);
    y(j,:)=x(j,:)*b(j,:)'+v(j,:)*sqrt(R);
end

TRUE=[diag(F);MU';R;diag(Q)];
%***********step 1 set priors for each parameter
%  F~N(F0,VF0)
F0=ones(2,1);
VF0=eye(2)*2;
%MU~N(MU0,VMU0);
MU0=zeros(2,1);
VMU0=eye(2);
%R~IG(R0,VR0)
R0=1;
VR0=1;
%Q(i,i)~IG(Q0,VQ0)
Q0=0.1;
VQ0=1;
%***************step 2 estimate model via maximum likelihood
theta0=ones(7,1).*0.1;
[FF,AA,gh,hess,itct,fcount,retcodeh] = csminwel('posterior',theta0,eye(length(theta0))*.1,[],1e-15,1000,y,x,F0,VF0,MU0,VMU0,R0,VR0,Q0,VQ0);
%**************step 2 set scale factor for the metropolis hastings
K=0.4;  %scaling factor
P=(chol(hess*K)); %compute variance of the random walk
Gammaold=AA; posterior mode estimates
REPS=50000;
BURN=30000;
naccept=0;
out1=zeros(REPS-BURN,7);
out2=zeros(REPS-BURN,1);
%compute posterior at old draw
           %compute -1*likelihood at old draw
```

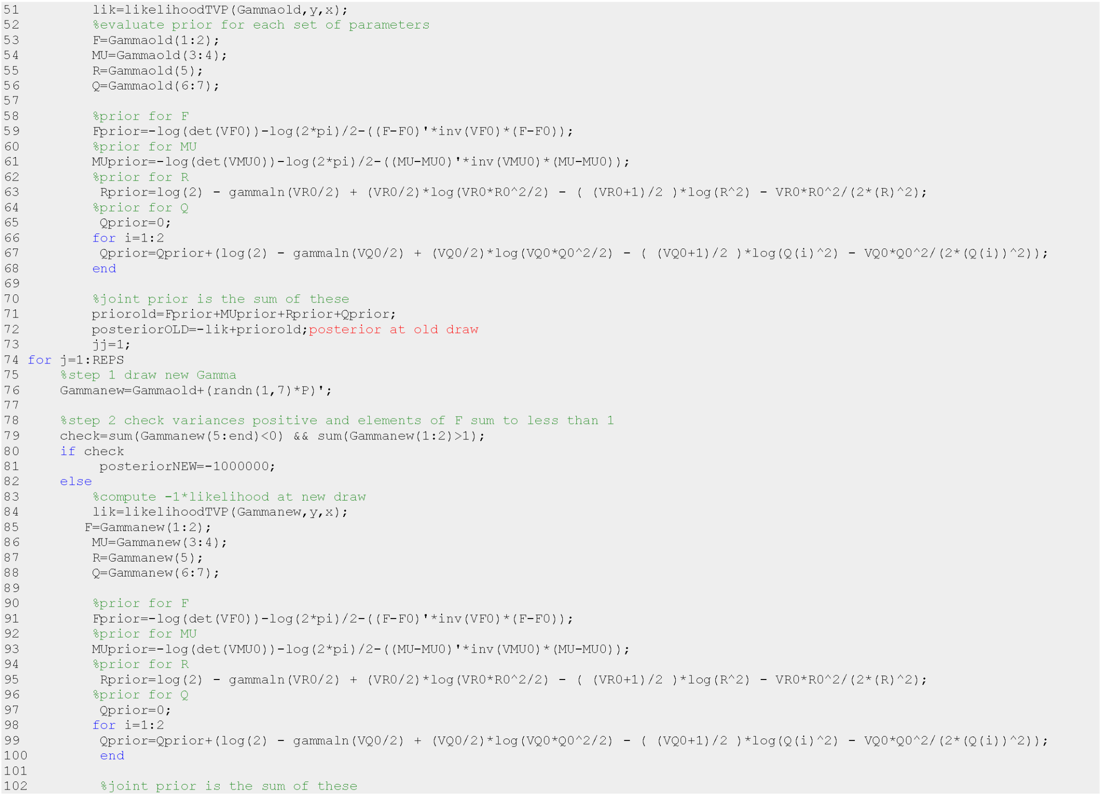
Name of function that calculates posterior

Initial hessian (leave as default)

Starting values

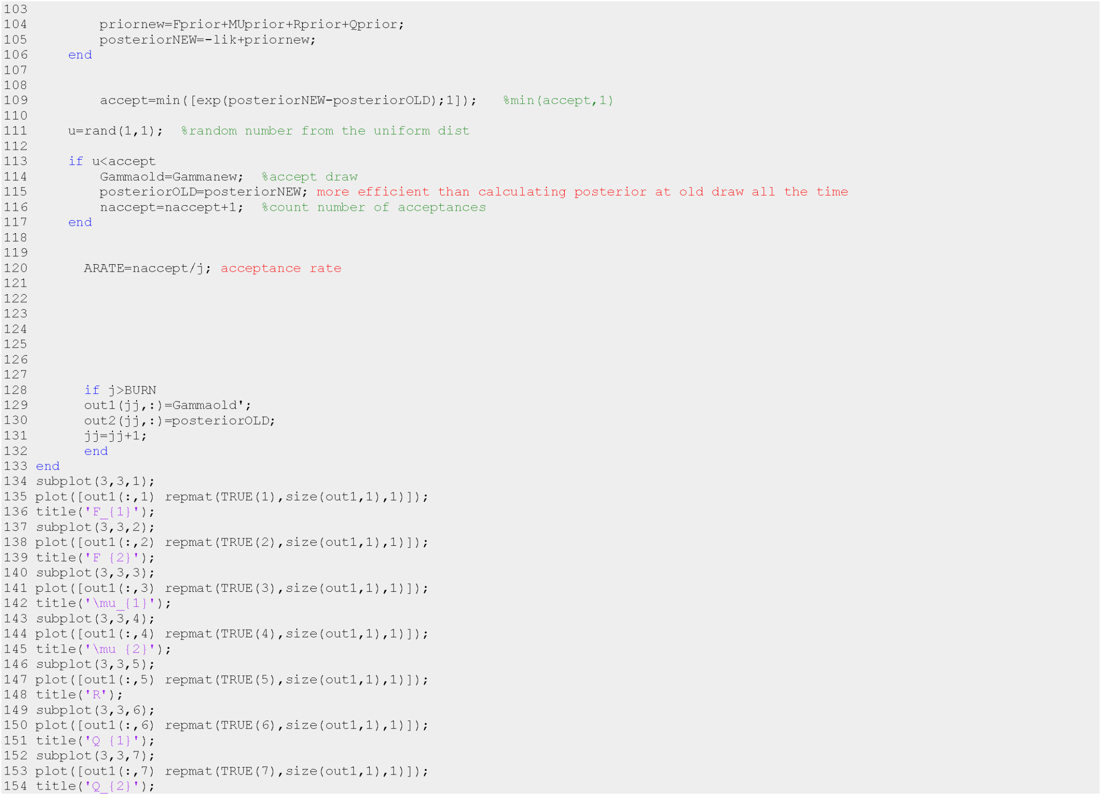
Tolerance for convergence

FIGURE 8. Matlab code for the TVP model



```
        lik=likelihoodTVP(Gammaold,y,x);
        %evaluate prior for each set of parameters
        F=Gammaold(1:2);
        MU=Gammaold(3:4);
        R=Gammaold(5);
        Q=Gammaold(6:7);

        %prior for F
        Fprior=-log(det(VF0))-log(2*pi)/2-((F-F0)'*inv(VF0)*(F-F0));
        %prior for MU
        MUprior=-log(det(VMU0))-log(2*pi)/2-((MU-MU0)'*inv(VMU0)*(MU-MU0));
        %prior for R
         Rprior=log(2) - gammaln(VR0/2) + (VR0/2)*log(VR0*R0^2/2) - ( (VR0+1)/2 )*log(R^2) - VR0*R0^2/(2*(R)^2);
        %prior for Q
         Qprior=0;
        for i=1:2
         Qprior=Qprior+(log(2) - gammaln(VQ0/2) + (VQ0/2)*log(VQ0*Q0^2/2) - ( (VQ0+1)/2 )*log(Q(i)^2) - VQ0*Q0^2/(2*(Q(i))^2));
        end

        %joint prior is the sum of these
        priorold=Fprior+MUprior+Rprior+Qprior;
        posteriorOLD=-lik+priorold;posterior at old draw
        jj=1;
for j=1:REPS
    %step 1 draw new Gamma
    Gammanew=Gammaold+(randn(1,7)*P)';

    %step 2 check variances positive and elements of F sum to less than 1
    check=sum(Gammanew(5:end)<0) && sum(Gammanew(1:2)>1);
    if check
         posteriorNEW=-1000000;
    else
        %compute -1*likelihood at new draw
        lik=likelihoodTVP(Gammanew,y,x);
       F=Gammanew(1:2);
        MU=Gammanew(3:4);
        R=Gammanew(5);
        Q=Gammanew(6:7);

        %prior for F
        Fprior=-log(det(VF0))-log(2*pi)/2-((F-F0)'*inv(VF0)*(F-F0));
        %prior for MU
        MUprior=-log(det(VMU0))-log(2*pi)/2-((MU-MU0)'*inv(VMU0)*(MU-MU0));
        %prior for R
         Rprior=log(2) - gammaln(VR0/2) + (VR0/2)*log(VR0*R0^2/2) - ( (VR0+1)/2 )*log(R^2) - VR0*R0^2/(2*(R)^2);
        %prior for Q
         Qprior=0;
        for i=1:2
         Qprior=Qprior+(log(2) - gammaln(VQ0/2) + (VQ0/2)*log(VQ0*Q0^2/2) - ( (VQ0+1)/2 )*log(Q(i)^2) - VQ0*Q0^2/(2*(Q(i))^2));
         end

         %joint prior is the sum of these
```

FIGURE 9. Matlab code for the TVP model (continued)



```
103
104         priornew=Fprior+MUprior+Rprior+Qprior;
105         posteriorNEW=-lik+priornew;
106     end
107
108
109         accept=min([exp(posteriorNEW-posteriorOLD);1]);   %min(accept,1)
110
111     u=rand(1,1);  %random number from the uniform dist
112
113     if u<accept
114         Gammaold=Gammanew;  %accept draw
115         posteriorOLD=posteriorNEW; more efficient than calculating posterior at old draw all the time
116         naccept=naccept+1;  %count number of acceptances
117     end
118
119
120       ARATE=naccept/j; acceptance rate
121
122
123
124
125
126
127
128       if j>BURN
129       out1(jj,:)=Gammaold';
130       out2(jj,:)=posteriorOLD;
131       jj=jj+1;
132       end
133 end
134 subplot(3,3,1);
135 plot([out1(:,1) repmat(TRUE(1),size(out1,1),1)]);
136 title('F_{1}');
137 subplot(3,3,2);
138 plot([out1(:,2) repmat(TRUE(2),size(out1,1),1)]);
139 title('F {2}');
140 subplot(3,3,3);
141 plot([out1(:,3) repmat(TRUE(3),size(out1,1),1)]);
142 title('\mu_{1}');
143 subplot(3,3,4);
144 plot([out1(:,4) repmat(TRUE(4),size(out1,1),1)]);
145 title('\mu {2}');
146 subplot(3,3,5);
147 plot([out1(:,5) repmat(TRUE(5),size(out1,1),1)]);
148 title('R');
149 subplot(3,3,6);
150 plot([out1(:,6) repmat(TRUE(6),size(out1,1),1)]);
151 title('Q {1}');
152 subplot(3,3,7);
153 plot([out1(:,7) repmat(TRUE(7),size(out1,1),1)]);
154 title('Q_{2}');
```

Figure 10. Matlab code for TVP model continued

function for calculating analytical derivitives. If this is unavailable then we enter an empty matrix [] as done above. (5) The tolerance level to stop the iterative procedure. This should be left as default. (6) The maximum number of iterations set to a 1000 in the example above. All the remaining arguments (y,x,F0,VF0,MU0,VMU0,R0,VR0,Q0,VQ0) are passed directly to the function posterior.m and are inputs for that function. The function returns (1) FF the value of the function at the minimum. (2) AA the value of the parameters at the minimum and (3) hess the inverse hessian of the function being minimised

Line 42 of the code sets the variance of the random walk candidate generating density as a scalar times the parameter variance obtained from the optimisation using csminwel. Line 43 sets the initial value of the parameters as the posterior mode estimates.

Line 51 calculates the log likelihood at the initial value of the parameters. Lines 59 to 68 evaluate the log prior distributions for the parameters of the state space model. Line 69 calculates the log joint prior as the sum of these prior distributions. Line 70 calculates the log posterior as the sum of the log likelihood and the log joint prior.

Line 74 draws the new value of the parameters from the random walk candidate generating density. Line 82 calculates the log likelihood at the new draw (assuming that the drawn variances are positive and the elements of F sum to less than 1). Lines 83 to 100 evaluate the log joint prior at the new draw and line 101 calculates the posterior. Line 109 calculates the acceptance probability. If this is bigger than a number from the standard uniform distribution then the new draw of the parameters is accepted. In this case Line 115 also sets posteriorOLD to posteriorNEW– it automatically updates the value of the posterior at the old draw eliminating the need to compute the posterior at the old draw at every iteration (as we have done in the examples above).

Line 120 computes the acceptance rate (the ratio of the number of accepted draws and the total draws). Once past the burn-in stage we save the draws of the model parameters. Figure 11 shows the retained draws of the parameters along with the true values.

3.4. The random walk MH algorithm used in a Threshold VAR model. In this section, we consider how the MH algorithm is used in the estimation of a Threshold VAR model (TVAR). The TVAR is defined as

$$
\begin{aligned}
Y_t &= c_1 + \sum_{j=1}^{P} \beta_1 Y_{j,t-j} + v_t, VAR(v_t) = \Omega_1 \text{ if } S_t \leq Y^* \\
Y_t &= c_2 + \sum_{j=1}^{P} \beta_2 Y_{j,t-j} + v_t, VAR(v_t) = \Omega_2 \text{ if } S_t > Y^*
\end{aligned}
$$

where Y_t is a matrix of endogenous variables, $S_t = Y_{j,t-d}$ (i.e. a lag of one of the endogenous variables) is the threshold variable and Y^* is the threshold level. Note that if Y^* and d are known, then the TVAR is simply two

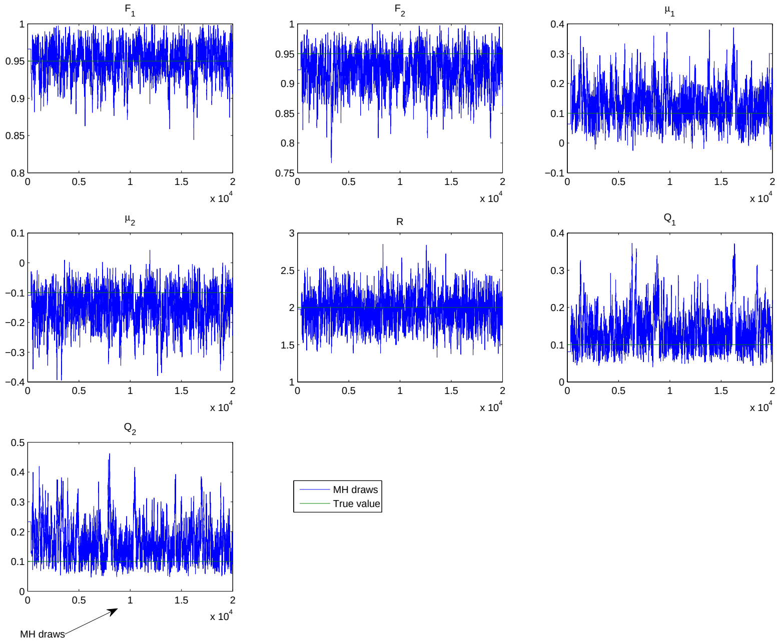


FIGURE 11. MH draws for the TVP model

VAR models defined over the appropriate data samples using $S_t \leq Y^*$ and $S_t > Y^*$. This observation allows us to devise a Gibbs algorithm (with a MH step). In what follows below we assume the delay parameter d to be known. See Chen and Lee (1995) for the extension of the algorithm to the case where d is estimated.

Step 1 Set Priors. In the application below, we assume $p(Y^*) \tilde{} N(\bar{Y}^*, \sigma_{Y^*})$. We set a natural conjugate prior for the VAR parameters in both regimes using dummy observations. See the prior used in section 6. Set an initial value for Y^*. One way to do this is to use the mean or median of $Y_{i,t-d}$.

Step 2 Seperate the data into two regimes. The first regime includes all observations such that $S_t \leq Y^*$. Call this sample $Y_{1,t}$. The second regime includes all observations such that $S_t > Y^*$. Call this sample $Y_{2,t}$.

Step 3 Sample the VAR parameters $b_i = \{c_i, \beta_i\}$ and Ω_i in each regime $i = 1, 2$. Let X denote the right hand side variables of the VAR. The conditional distribution is exactly as defined in chapter 2 above and is given by

$$H(b_i|\Omega_i, Y_t, Y^*) \tilde{} N(vec(B_i^*), \Omega_i \otimes (X_i^{*\prime}X_i^*)^{-1}) \qquad (3.18)$$
$$H(\Omega_i|b_i, Y_t, , Y^*) \tilde{} IW(S_i^*, T_i^*)$$

where

$$B_i^* = (X_i^{*\prime}X_i^*)^{-1}(X_i^{*\prime}y_i^*)$$
$$S_i^* = (y_i^* - X_i^* b)'(y_i^* - X_i^* b_i)$$

where $y_i^* = [Y_{i,t}; Y_D]$ and $X_i^* = [X_{i,t}; X_D]$ with Y_D, X_D the dummy observations that define the prior for the left and the right hand side of the VAR respectively.

Step 4 Use a MH step to sample Y^*. Draw a new value of the threshold from the random walk

$$Y_{new}^* = Y_{old}^* + e, e\tilde{}N(0, \Sigma)$$

Then compute the acceptance probability

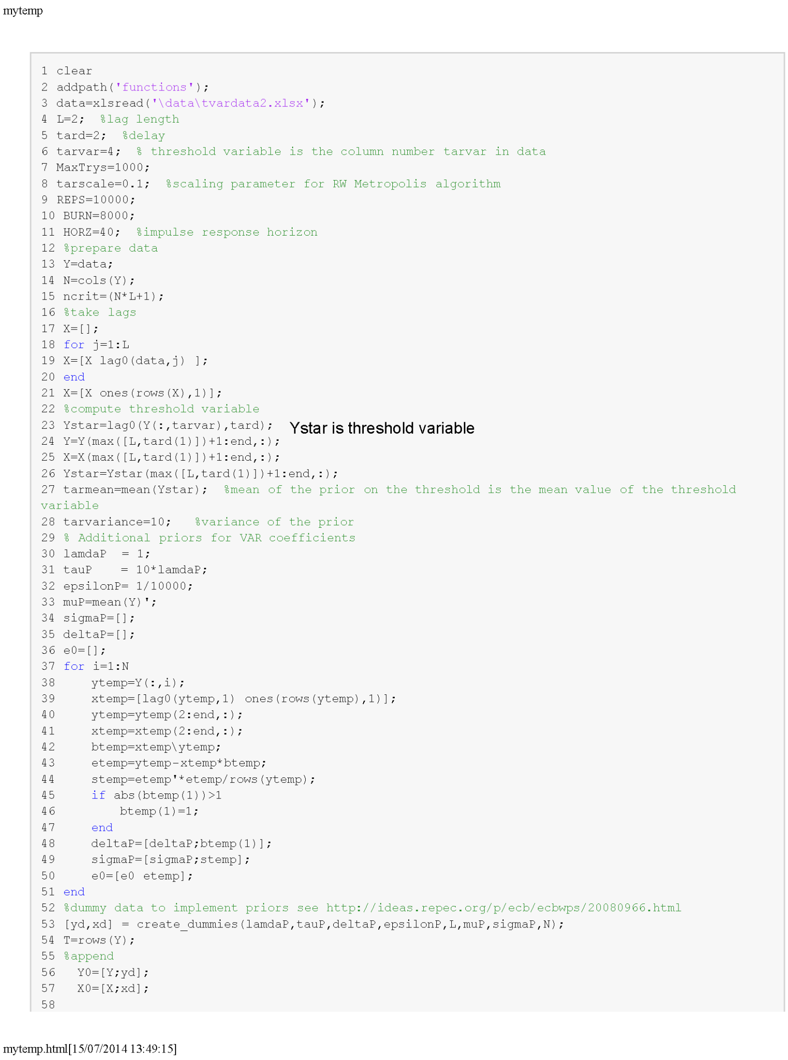
$$\alpha = \frac{F(Y|b_i, \Omega_i, Y_{new}^*)\, p(Y_{new}^*)}{F(Y|b_i, \Omega_i, Y_{old}^*)\, p(Y_{old}^*)}$$

where $F(Y|b_i, \Omega_i, Y_{new}^*)$ is the likelihood of the VAR computed as the product of the likelihoods in the two regimes. The log likelihood in each regime (ignoring constants) is

$$\left(\frac{T}{2}\right)\log|\Omega_i^{-1}| - 0.5\sum_{t=1}^{T}\left[\left(Y_{i,t} - X_{i,t}\tilde{b}_i\right)'\Omega_i^{-1}\left(Y_{i,t} - X_{i,t}\tilde{b}_i\right)\right]$$

with $\tilde{b}_i$ equivalent to b_i reshaped to be conformable with $X_{i,t}$. Then draw $u\tilde{}U(0,1)$. If $u < \alpha$ accept Y_{new}^* else retain Y_{old}^*. The scale Σ can be tuned to ensure an acceptance rate between 20% and 40%.

As an example we consider a TVAR where Y contains US data on GDP growth, CPI inflation, a short term interest rate and a financial conditions index (FCI) calculated by the Chicago Fed. The threshold variable is assumed to be the second lag of FCI and examine the impulse response of the variables to a unit increase in FCI (a deterioration



mytemp

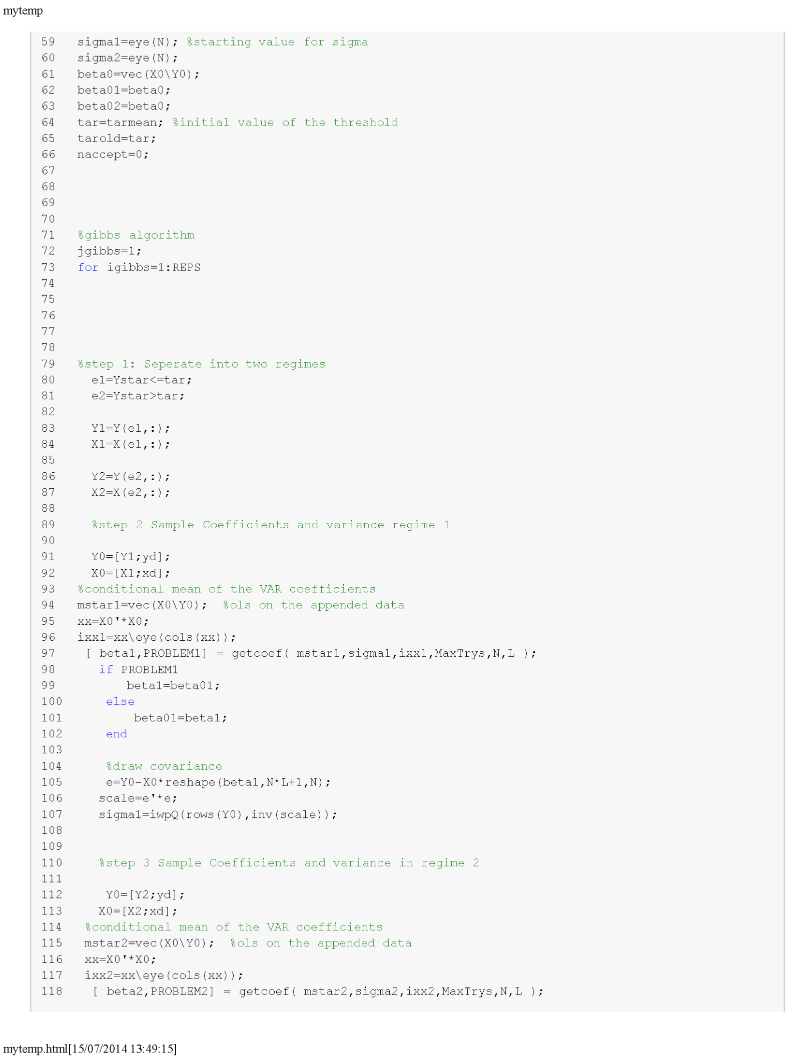
```
1 clear
2 addpath('functions');
3 data=xlsread('\data\tvardata2.xlsx');
4 L=2;  %lag length
5 tard=2;  %delay
6 tarvar=4;  % threshold variable is the column number tarvar in data
7 MaxTrys=1000;
8 tarscale=0.1;  %scaling parameter for RW Metropolis algorithm
9 REPS=10000;
10 BURN=8000;
11 HORZ=40;  %impulse response horizon
12 %prepare data
13 Y=data;
14 N=cols(Y);
15 ncrit=(N*L+1);
16 %take lags
17 X=[];
18 for j=1:L
19 X=[X lag0(data,j) ];
20 end
21 X=[X ones(rows(X),1)];
22 %compute threshold variable
23 Ystar=lag0(Y(:,tarvar),tard);
24 Y=Y(max([L,tard(1)])+1:end,:);
25 X=X(max([L,tard(1)])+1:end,:);
26 Ystar=Ystar(max([L,tard(1)])+1:end,:);
27 tarmean=mean(Ystar);  %mean of the prior on the threshold is the mean value of the threshold
variable
28 tarvariance=10;   %variance of the prior
29 % Additional priors for VAR coefficients
30 lamdaP  = 1;
31 tauP    = 10*lamdaP;
32 epsilonP= 1/10000;
33 muP=mean(Y)';
34 sigmaP=[];
35 deltaP=[];
36 e0=[];
37 for i=1:N
38     ytemp=Y(:,i);
39     xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
40     ytemp=ytemp(2:end,:);
41     xtemp=xtemp(2:end,:);
42     btemp=xtemp\ytemp;
43     etemp=ytemp-xtemp*btemp;
44     stemp=etemp'*etemp/rows(ytemp);
45     if abs(btemp(1))>1
46         btemp(1)=1;
47     end
48     deltaP=[deltaP;btemp(1)];
49     sigmaP=[sigmaP;stemp];
50     e0=[e0 etemp];
51 end
52 %dummy data to implement priors see http://ideas.repec.org/p/ecb/ecbwps/20080966.html
53 [yd,xd] = create_dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
54 T=rows(Y);
55 %append
56   Y0=[Y;yd];
57   X0=[X;xd];
58
```

Ystar is threshold variable

mytemp.html[15/07/2014 13:49:15]

FIGURE 12. Matlab code for TVAR model

of financial conditions) in the two regimes. The matlab code is in the file named thresholdvarNFCI.m and displayed in figures 12, 13 and 14. In this example the prior $p(Y^*)\,\tilde{}\,N(\bar{Y}^*, \sigma_{Y^*})$ is set by using the mean of NFCI as $\bar{Y}^*$ and $\sigma_{Y^*} = 10$ (line 28). Lines 30 to 53 set the natural conjugate prior for the VAR parameters. Lines 80 to 87 seperate the sample into two regimes. Lines 89 to 128 draw the VAR coefficients and covariance in each regime. The MH step to draw the threshold variable starts on line 134 with a draw from the random walk candidate density. Then the log posterior $\ln\left(F\left(Y|b_i, \Omega_i, Y^*_{new}\right)p\left(Y^*_{new}\right)\right)$ is computed on line 136 while $\ln\left(F\left(Y|b_i, \Omega_i, Y^*_{old}\right)p\left(Y^*_{old}\right)\right)$ is computed on line 137. The acceptance probability is computed on line 138.

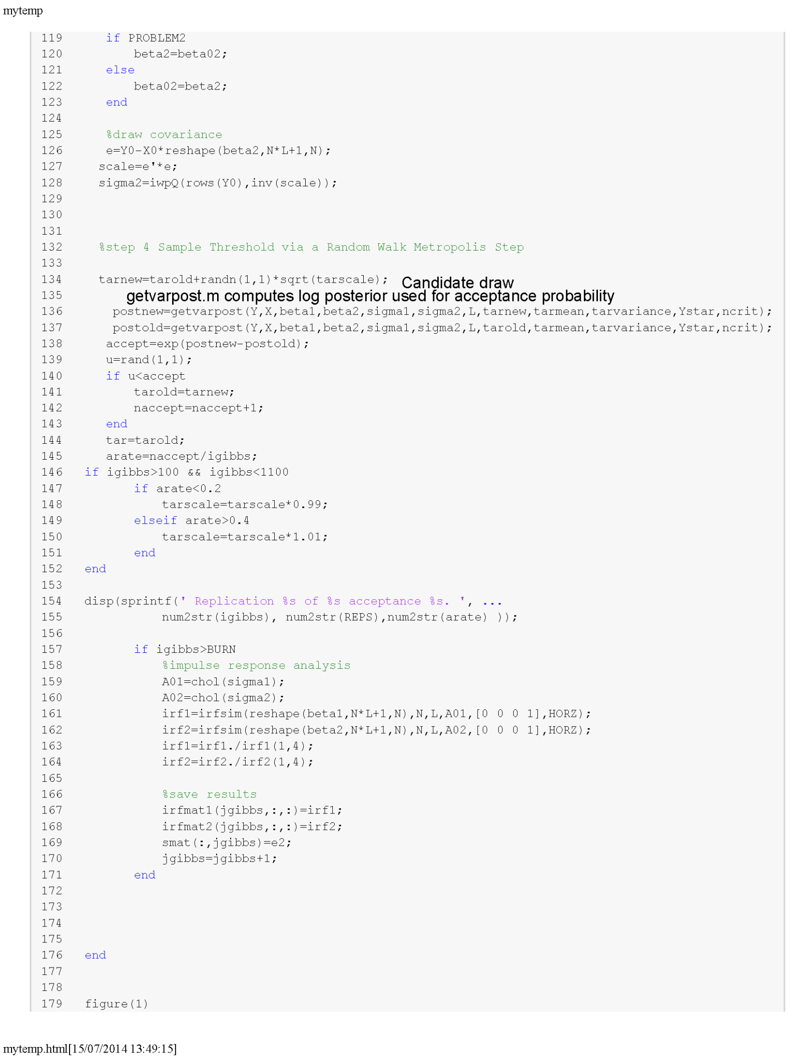


mytemp

```
sigma1=eye(N); %starting value for sigma
sigma2=eye(N);
beta0=vec(X0\Y0);
beta01=beta0;
beta02=beta0;
tar=tarmean; %initial value of the threshold
tarold=tar;
naccept=0;




%gibbs algorithm
jgibbs=1;
for igibbs=1:REPS





%step 1: Seperate into two regimes
  e1=Ystar<=tar;
  e2=Ystar>tar;

  Y1=Y(e1,:);
  X1=X(e1,:);

  Y2=Y(e2,:);
  X2=X(e2,:);

  %step 2 Sample Coefficients and variance regime 1

  Y0=[Y1;yd];
  X0=[X1;xd];
%conditional mean of the VAR coefficients
mstar1=vec(X0\Y0);  %ols on the appended data
xx=X0'*X0;
ixx1=xx\eye(cols(xx));
 [ beta1,PROBLEM1] = getcoef( mstar1,sigma1,ixx1,MaxTrys,N,L );
   if PROBLEM1
       beta1=beta01;
    else
        beta01=beta1;
    end

    %draw covariance
    e=Y0-X0*reshape(beta1,N*L+1,N);
   scale=e'*e;
   sigma1=iwpQ(rows(Y0),inv(scale));


   %step 3 Sample Coefficients and variance in regime 2

    Y0=[Y2;yd];
   X0=[X2;xd];
 %conditional mean of the VAR coefficients
 mstar2=vec(X0\Y0);  %ols on the appended data
 xx=X0'*X0;
 ixx2=xx\eye(cols(xx));
  [ beta2,PROBLEM2] = getcoef( mstar2,sigma2,ixx2,MaxTrys,N,L );
```

mytemp.html[15/07/2014 13:49:15]

Figure 13. Matlab code for TVAR model



mytemp

```
   if PROBLEM2
       beta2=beta02;
   else
       beta02=beta2;
   end

   %draw covariance
   e=Y0-X0*reshape(beta2,N*L+1,N);
  scale=e'*e;
  sigma2=iwpQ(rows(Y0),inv(scale));



  %step 4 Sample Threshold via a Random Walk Metropolis Step

  tarnew=tarold+randn(1,1)*sqrt(tarscale);

    postnew=getvarpost(Y,X,beta1,beta2,sigma1,sigma2,L,tarnew,tarmean,tarvariance,Ystar,ncrit);
    postold=getvarpost(Y,X,beta1,beta2,sigma1,sigma2,L,tarold,tarmean,tarvariance,Ystar,ncrit);
   accept=exp(postnew-postold);
   u=rand(1,1);
   if u<accept
       tarold=tarnew;
       naccept=naccept+1;
   end
   tar=tarold;
   arate=naccept/igibbs;
if igibbs>100 && igibbs<1100
       if arate<0.2
           tarscale=tarscale*0.99;
       elseif arate>0.4
           tarscale=tarscale*1.01;
       end
end

disp(sprintf(' Replication %s of %s acceptance %s. ', ...
           num2str(igibbs), num2str(REPS),num2str(arate) ));

       if igibbs>BURN
           %impulse response analysis
           A01=chol(sigma1);
           A02=chol(sigma2);
           irf1=irfsim(reshape(beta1,N*L+1,N),N,L,A01,[0 0 0 1],HORZ);
           irf2=irfsim(reshape(beta2,N*L+1,N),N,L,A02,[0 0 0 1],HORZ);
           irf1=irf1./irf1(1,4);
           irf2=irf2./irf2(1,4);

           %save results
           irfmat1(jgibbs,:,:)=irf1;
           irfmat2(jgibbs,:,:)=irf2;
           smat(:,jgibbs)=e2;
           jgibbs=jgibbs+1;
       end




end


figure(1)
```

Candidate draw

getvarpost.m computes log posterior used for acceptance probability

mytemp.html[15/07/2014 13:49:15]

Figure 14. Matlab code for TVAR model

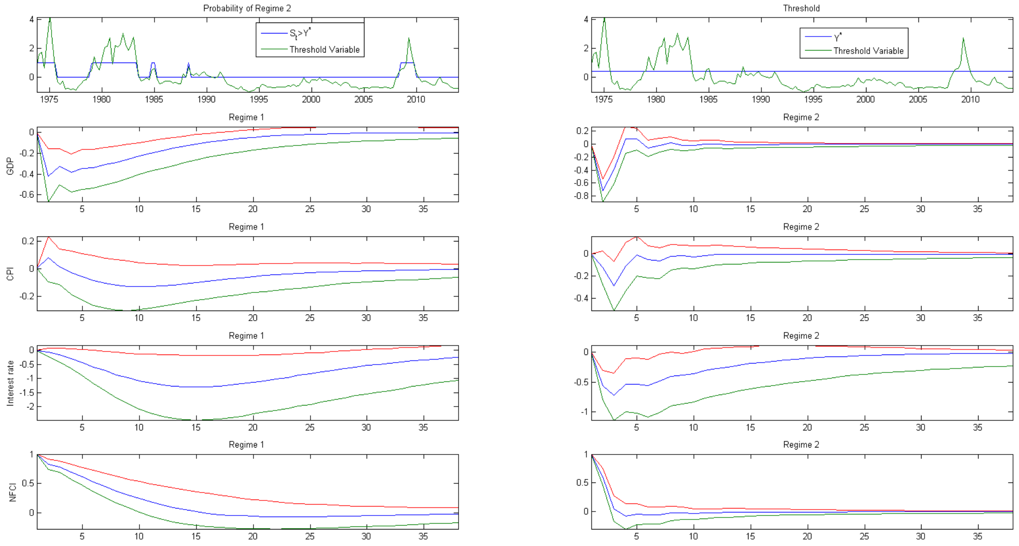


FIGURE 15. Results from the TVAR model for the US

The top right panel of figure 15 plots the estimated threshold and the threshold variable and shows that regime 2 persisted in the 1980s, the early 1990s and then during the recent recession. There is some evidence that the negative impact of an FCI shock on GDP growth is larger in regime 2.

3.5. The random walk MH algorithm used in a STVAR model. A related model is the smooth transition (ST)VAR given by:

$$\begin{aligned} Y_t &= (1-G(\gamma,Y^*,S_t))\left(c_1+\sum_{j=1}^{P}\beta_{1,j}Y_{t-j}\right)+ \\ &\quad G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{P}\beta_{2,j}Y_{t-j}\right)+v_t, \\ var(v_t) &= \Omega \end{aligned}$$

where $G(\gamma,Y^*,S_t)=\frac{1}{1+\exp(-\gamma(S_t-Y^*))}$. Here Y_t is a matrix of endogenous variables, $S_t=Y_{j,t-d}$ (i.e. a lag of one of the endogenous variables) is the threshold variable and Y^* is the threshold level and $\gamma>0$ is the smoothness parameter that determines the smoothness of the regime shifts. The Gibbs algorithm proceeds in the following steps (see Lopes and Salazar (2006))

(1) Set Priors. We assume $p(Y^*)\,\tilde{}\,N(\bar{Y}^*,\sigma_{Y^*})$. We assume a Gamma prior for $\gamma: p(\gamma)\,\tilde{}\,\Gamma(\gamma_0,v_0)$. We set a natural conjugate prior for the VAR parameters $B_i=\{c_i,\beta_{i,j}\}$ for $i=1,2.$ in both regimes using dummy observations. See the prior used in section 6. Set an initial value for Y^*. One way to do this is to use the mean or median of $Y_{i,t-d}$.

(2) Sample from $H(B_1|B_2,\Omega,\gamma,Y^*)$. Write the VAR model as

$$Y_t-G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{P}\beta_{2,j}Y_{t-j}\right)=G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{P}\beta_{2,j}Y_{t-j}\right)+v_t$$

or

$$\bar{Y}_t=B_1\bar{X}_t+v_t$$

where $\bar{Y}_t=Y_t-G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{P}\beta_{2,j}Y_{t-j}\right)$

and $\bar{X}_t=\{G(\gamma,Y^*,S_t),G(\gamma,Y^*,S_t)(Y_{t-1}),...,G(\gamma,Y^*,S_t)(Y_{t-j})\}$. This is a standard VAR model and the conditional posterior is as described for the coefficients of the Threshold VAR above:

$$H(B_1|B_2,\Omega,\gamma,Y^*)\,\tilde{}\,N(vec(B_i^*),\Omega\otimes(X_t^{*\prime}X_t^*)^{-1}) \tag{3.19}$$

where $X_t^*=[\bar{X}_t;X_D]$ with Y_D,X_D the dummy observations that define the prior for the left and the right hand side of the VAR respectively.

(3) Sample from $H(B_2|B_1,\Omega,\gamma,Y^*)$.
We proceed exactly as in step 2 by defining $\bar{Y}_t=Y_t-(1-G(\gamma,Y^*,S_t))\left(c_1+\sum_{j=1}^{P}\beta_{1,j}Y_{t-j}\right)$.

(4) Sample from $H\left(\Omega|B_2,B_1,\Omega,\gamma,Y^*\right)$. As in the previous example this conditional posterior is $IW(S^*,T^*)$ where $S^*=\left(y_t-B_1X_t^*\right)'\left(y_t-B_1X_t^*\right)$ with $y_t=[\bar{Y}_t;Y_D]$.
(5) Sample from $H\left(\gamma,Y^*|B_1,B_2,\Omega\right)$. We use a random walk MH step to sample $\Xi=\gamma,Y^*$. Draw a new value from $\Xi^{new}=\Xi^{old}+e,e\tilde{}N(0,\Sigma)$. The acceptance probability is

$$\alpha=\frac{F\left(Y|B_i,\Omega,\Xi^{new}\right)p\left(\Xi^{new}\right)}{F\left(Y|B_i,\Omega,\Xi^{old}\right)p\left(\Xi^{old}\right)}$$

where $F\left(Y|B_i,\Omega,\Xi^{new}\right)$ is the likelihood of the VAR:

$$\left(\frac{T}{2}\right)\log\left|\Omega^{-1}\right|-0.5\sum_{t=1}^{T}\left[\left(v_t\right)'\Omega^{-1}\left(v_t\right)\right]$$

where v_t are the VAR residuals. Then draw $u\tilde{}U(0,1)$. If $u<\alpha$ accept Y^*_{new} else retain Y^*_{old}. The scale Σ can be tuned to ensure an acceptance rate between 20% and 40%.
(6) Repeat 2 to 5 until convergence.

The code for this model is shown in figures 16 to 18. Here we use artificial data generated from a STVAR to test this algorithm. The Gibbs sampler begins on line 88. The transition function $G\left(\gamma,Y^*,S_t\right)=\frac{1}{1+\exp(-\gamma(S_t-Y^*))}$ is evaluated on line 94. The VAR coefficients in the two regimes are drawn on lines 101 to 127 and VAR error covariance is drawn 129 to 131. The MH step to draw $\Xi=\gamma,Y^*$ begins on line 137 with a draw from the candidate density. Lines 139 to 140 calculate the posterior at the new and old draw of Ξ using the function getvarpostx. Finally the acceptance probability is calculated on line 142.

3.6. The random walk metropolis algorithm for structural VAR model.

Consider an SVAR model

$$Y_t=BX_t+v_t$$

where $X_t=\{c,Y_{t-1},Y_{t-2},...,Y_{t-P}\}$ and $var\left(v_t\right)=\Omega=A^{-1}A^{-1'}$. Here A^{-1} is the contemporaneous impact matrix. In Chapter 2, we proceeded via Gibbs sampling where the draws of Ω were used to calculate the contemporaneous impact matrix. As discussed in Sims and Zha (1999), when the model is over identified (the number of distinct elements in Ω exceed the number of free parameters in A^{-1}), this naive approach of obtaining A indirectly from the draws of Ω may fail to provide a good approximation of the posterior of A. Instead, the correct approach is to sample from the posterior of A directly. Sims and Zha (1999) shows that the posterior distribution is given as:

$$H(A)\propto\det\left(A\right)^{T-K}\exp\left(-0.5trace\left(AS(\hat{B})A'\right)\right) \tag{3.20}$$

where $S(\hat{B})=\left(Y_t-\hat{B}X_t\right)'\left(Y_t-\hat{B}X_t\right)$ with $\hat{B}$ the OLS estimates of the VAR coefficients. Conditional on A the distribution of B is normal and (assuming a flat prior) given by:

$$H(B|A)\tilde{}N(vec(\hat{B}),\Omega\otimes\left(X_t'X_t\right)^{-1})$$

Therefore an MCMC algorithm can proceed via sampling from $H(A)$ and $H(B|A)$. As $H(A)$ is an unknown density, we use a random walk MH step to sample from it. The steps are as follows:

(1) Maximise $\log H(A)$ with respect to α the free elements of A to obtain the estimates at the posterior mode α^{ML} and the covariance V^{ML}.
(2) Draw from $P\left(A\right)$: Draw a candidate draw $\alpha^{new}=\alpha^{old}+e,e\tilde{}N(0,\left(V^{ML}\right)^{1/2}\times c)$. Compute the acceptance probability $a=\frac{H(A(\alpha^{new}))}{H(A(\alpha^{old}))}$ and accept the draw if $a>U(0,1)$.
(3) Draw from $H(B|A)$: Calculate $\Omega=A^{-1}A^{-1'}$ where A is based on the accepted draw of α in the previous step. Draw from $N(vec(\hat{B}),\Omega\otimes\left(X_t'X_t\right)^{-1})$.
(4) Repeat steps 2 and 3 until convergence. Adjust the scaling c to ensure that the acceptance rate is between 20% and 40%.

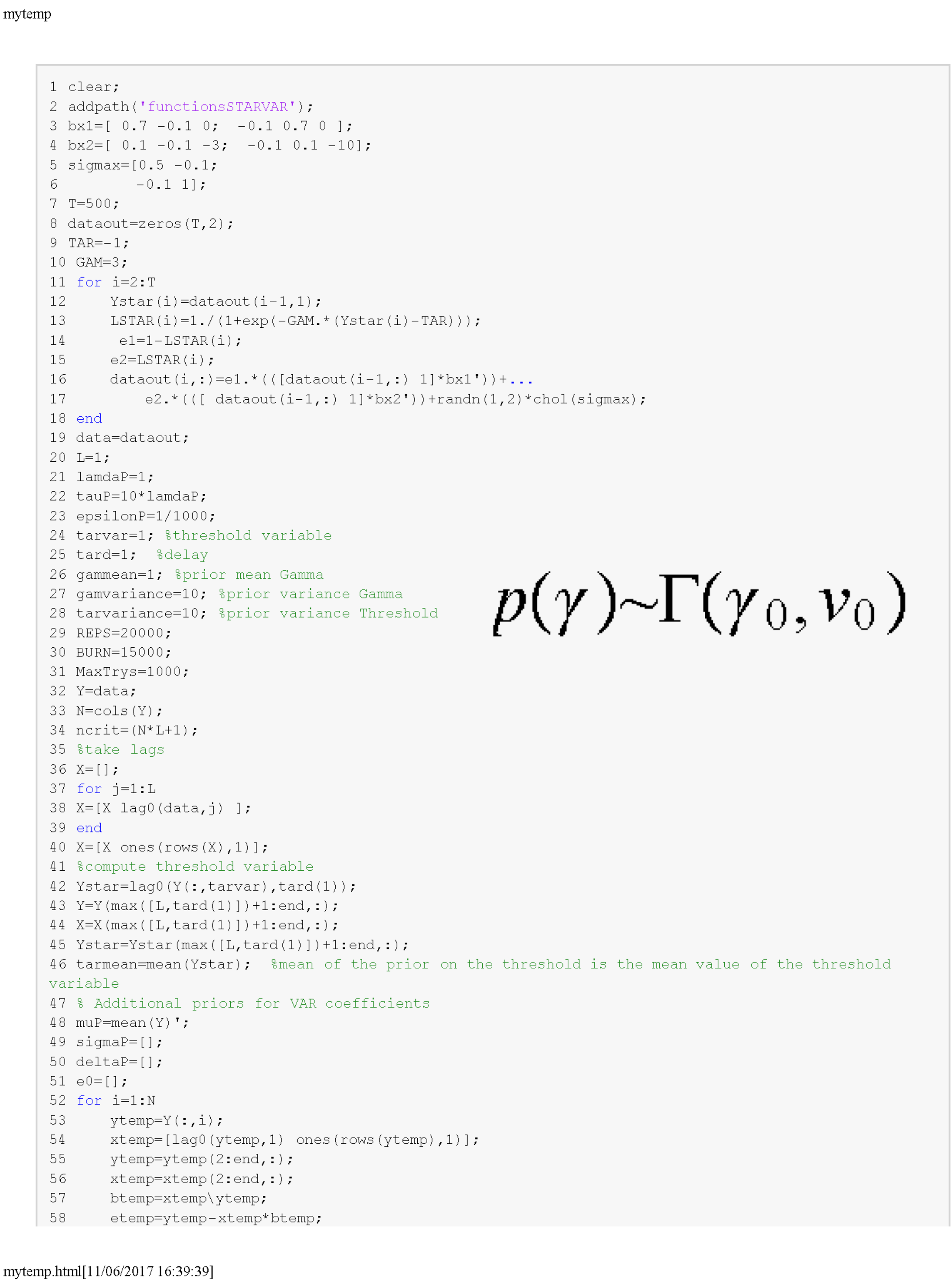
One important issue regarding step 2 needs to be highlighted. The sign of the columns of A can be switched without changing the likelihood function. Therefore a normalisation is required. A simple way to proceed is to switch the sign of the elements of a column if the corresponding diagonal element is negative. Note, however, that Waggoner and Zha (1997) show that this normalisation method may inflate the standard errors of the impulse responses and suggest an alternative approach to normalisation that preserves the shape of the likelihood.

To demonstrate the algorithm we generate data from a 3 variable VAR with the following A matrix:

$$A=\begin{pmatrix} x & 0 & 0 \\ x & x & 0 \\ 0 & x & x \end{pmatrix}$$

where x denotes parameters that need to be estimated. This model is overidentified with 5 parameters and 6 distinct elements in Ω.

The code for this example is shown in figures 19 and 20. This example is based on artificial data generated from the SVAR model on lines 1 to 17. The data is used to estimate the VAR coefficients and the sum of squares $S\left(\hat{B}\right)$



mytemp

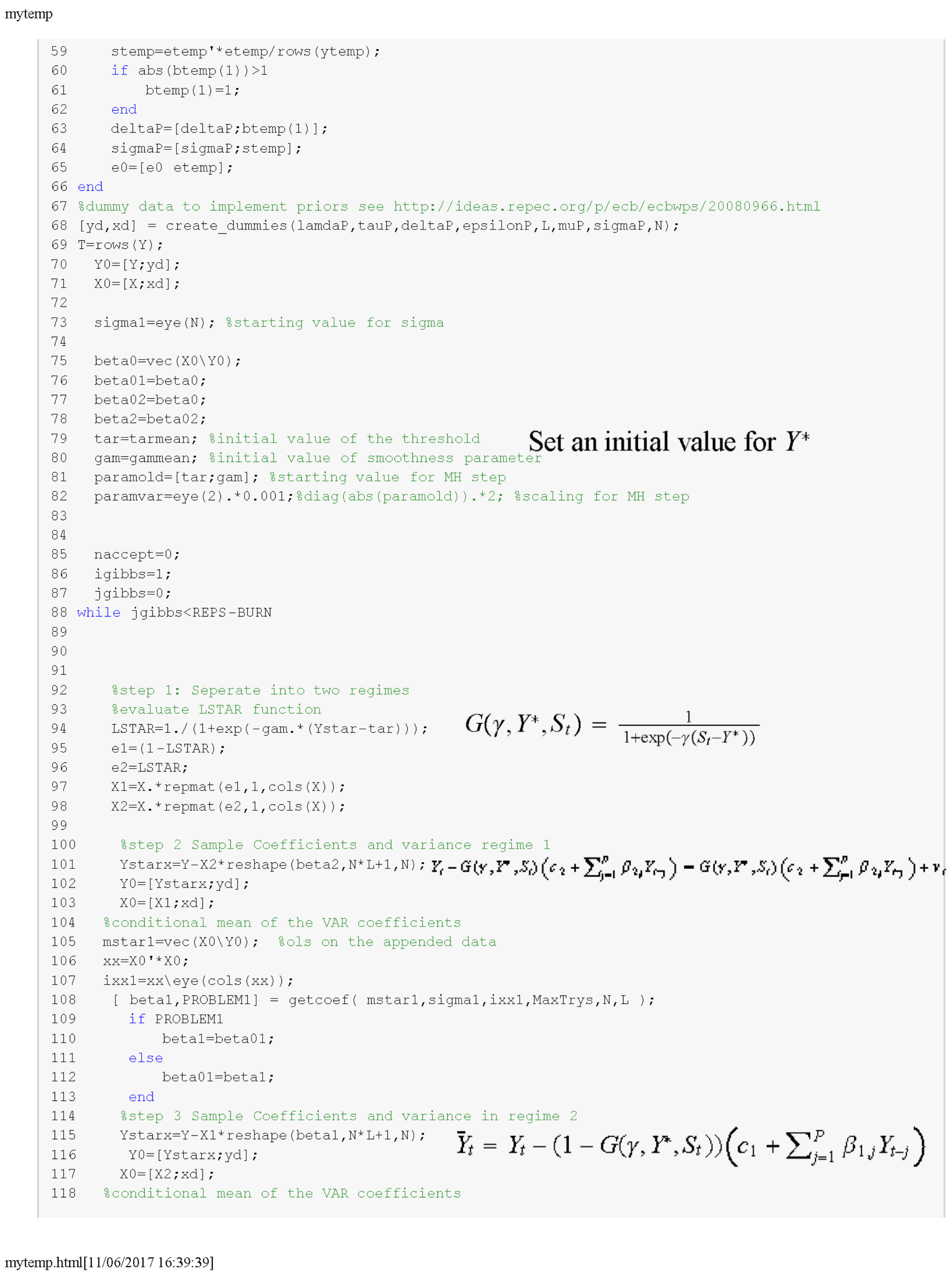
```
1 clear;
2 addpath('functionsSTARVAR');
3 bx1=[ 0.7 -0.1 0;  -0.1 0.7 0 ];
4 bx2=[ 0.1 -0.1 -3;  -0.1 0.1 -10];
5 sigmax=[0.5 -0.1;
6         -0.1 1];
7 T=500;
8 dataout=zeros(T,2);
9 TAR=-1;
10 GAM=3;
11 for i=2:T
12     Ystar(i)=dataout(i-1,1);
13     LSTAR(i)=1./(1+exp(-GAM.*(Ystar(i)-TAR)));
14      e1=1-LSTAR(i);
15     e2=LSTAR(i);
16     dataout(i,:)=e1.*(([dataout(i-1,:) 1]*bx1'))+...
17         e2.*(([ dataout(i-1,:) 1]*bx2'))+randn(1,2)*chol(sigmax);
18 end
19 data=dataout;
20 L=1;
21 lamdaP=1;
22 tauP=10*lamdaP;
23 epsilonP=1/1000;
24 tarvar=1; %threshold variable
25 tard=1;  %delay
26 gammean=1; %prior mean Gamma
27 gamvariance=10; %prior variance Gamma
28 tarvariance=10; %prior variance Threshold
29 REPS=20000;
30 BURN=15000;
31 MaxTrys=1000;
32 Y=data;
33 N=cols(Y);
34 ncrit=(N*L+1);
35 %take lags
36 X=[];
37 for j=1:L
38 X=[X lag0(data,j) ];
39 end
40 X=[X ones(rows(X),1)];
41 %compute threshold variable
42 Ystar=lag0(Y(:,tarvar),tard(1));
43 Y=Y(max([L,tard(1)])+1:end,:);
44 X=X(max([L,tard(1)])+1:end,:);
45 Ystar=Ystar(max([L,tard(1)])+1:end,:);
46 tarmean=mean(Ystar);  %mean of the prior on the threshold is the mean value of the threshold
variable
47 % Additional priors for VAR coefficients
48 muP=mean(Y)';
49 sigmaP=[];
50 deltaP=[];
51 e0=[];
52 for i=1:N
53     ytemp=Y(:,i);
54     xtemp=[lag0(ytemp,1) ones(rows(ytemp),1)];
55     ytemp=ytemp(2:end,:);
56     xtemp=xtemp(2:end,:);
57     btemp=xtemp\ytemp;
58     etemp=ytemp-xtemp*btemp;
```

$p(\gamma)\sim\Gamma(\gamma_0,v_0)$

mytemp.html[11/06/2017 16:39:39]

FIGURE 16. Code for the STVAR model

via OLS (Lines 28 to 31). The log posterior $(T-K)\ln\det(A)-0.5trace\left(AS(\hat{B})A'\right)$ is then maximised. Note that given a starting value of the 5 free parameters, the log posterior is evaluated by the function getML which returns the negative of this function. This negative posterior is then minimised first via Simplex on lines 41 using 500 iterations in the Matlab function fminsearch. This refines the starting values to be input into the main minimisation routine CSMINWEL which was introduced above. The values of α at the mode of the log posterior (minimum of minus log posterior) are given Theta2 and the covariance by hess. The former is used as the initial values in the metropolis step while the latter is used to calibrate the variance of the candidate distribution. The MCMC algorithm begins on line 58. Line 60 draws α from the candidate density. Lines 63 and 64 evaluate the posterior at the new and old draws with the acceptance probability calculated on line 66. The draw of A is normalised 75 to 77. The commented lines 78 to 80 show the normalisation rule proposed in Waggoner and Zha (2003). Finally, the VAR coefficients are drawn



mytemp

```
59     stemp=etemp'*etemp/rows(ytemp);
60     if abs(btemp(1))>1
61         btemp(1)=1;
62     end
63     deltaP=[deltaP;btemp(1)];
64     sigmaP=[sigmaP;stemp];
65     e0=[e0 etemp];
66 end
67 %dummy data to implement priors see http://ideas.repec.org/p/ecb/ecbwps/20080966.html
68 [yd,xd] = create_dummies(lamdaP,tauP,deltaP,epsilonP,L,muP,sigmaP,N);
69 T=rows(Y);
70   Y0=[Y;yd];
71   X0=[X;xd];
72
73   sigma1=eye(N); %starting value for sigma
74
75   beta0=vec(X0\Y0);
76   beta01=beta0;
77   beta02=beta0;
78   beta2=beta02;
79   tar=tarmean; %initial value of the threshold
80   gam=gammean; %initial value of smoothness parameter
81   paramold=[tar;gam]; %starting value for MH step
82   paramvar=eye(2).*0.001;%diag(abs(paramold)).*2; %scaling for MH step
83
84
85   naccept=0;
86   igibbs=1;
87   jgibbs=0;
88 while jgibbs<REPS-BURN
89
90
91
92     %step 1: Seperate into two regimes
93     %evaluate LSTAR function
94     LSTAR=1./(1+exp(-gam.*(Ystar-tar)));
95     e1=(1-LSTAR);
96     e2=LSTAR;
97     X1=X.*repmat(e1,1,cols(X));
98     X2=X.*repmat(e2,1,cols(X));
99
100     %step 2 Sample Coefficients and variance regime 1
101     Ystarx=Y-X2*reshape(beta2,N*L+1,N);
102     Y0=[Ystarx;yd];
103     X0=[X1;xd];
104   %conditional mean of the VAR coefficients
105   mstar1=vec(X0\Y0);  %ols on the appended data
106   xx=X0'*X0;
107   ixx1=xx\eye(cols(xx));
108    [ beta1,PROBLEM1] = getcoef( mstar1,sigma1,ixx1,MaxTrys,N,L );
109      if PROBLEM1
110          beta1=beta01;
111      else
112          beta01=beta1;
113      end
114     %step 3 Sample Coefficients and variance in regime 2
115     Ystarx=Y-X1*reshape(beta1,N*L+1,N);
116      Y0=[Ystarx;yd];
117     X0=[X2;xd];
118   %conditional mean of the VAR coefficients
```

Set an initial value for Y^*

$G(\gamma, Y^*, S_t) = \frac{1}{1+\exp(-\gamma(S_t - Y^*))}$

$Y_t - G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{p}\beta_{2j}Y_{t-j}\right) - G(\gamma,Y^*,S_t)\left(c_2+\sum_{j=1}^{p}\beta_{2j}Y_{t-j}\right)+v$

$\tilde{Y}_t = Y_t - (1 - G(\gamma, Y^*, S_t))\left(c_1 + \sum_{j=1}^{P} \beta_{1j} Y_{t-j}\right)$

mytemp.html[11/06/2017 16:39:39]

FIGURE 17. Code for the STVAR model

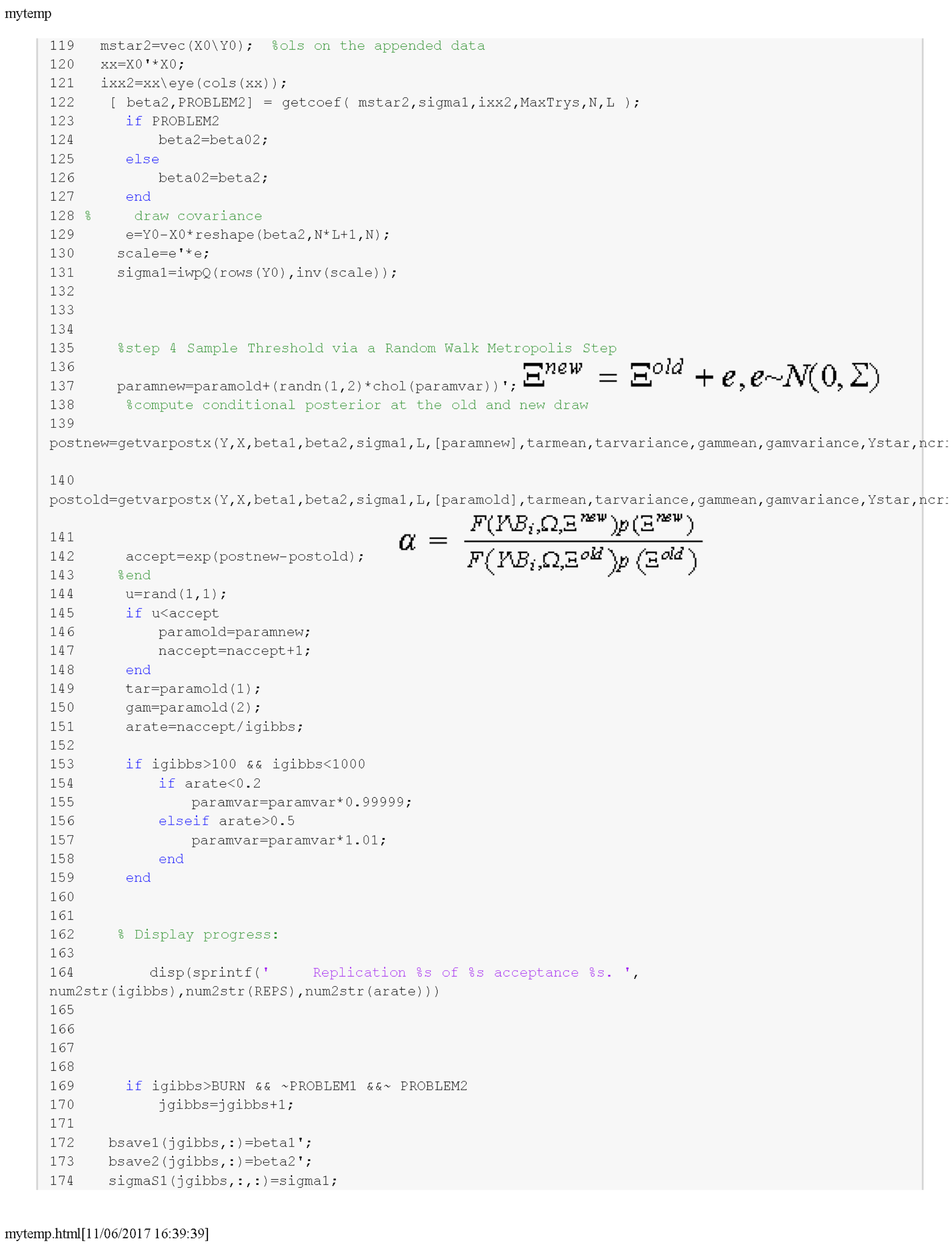
on 83 to 85. Running the code provides a comparison of the true impulse responses and the estimated posterior distribution.

4. The independence Metropolis Hastings algorithm

The independence MH algorithm differs from the random walk MH algorithm in that the candidate generating density is not specified as a random walk. Therefore, the new draw of the parameters does not depend directly on the previous draw. The candidate density is specified as

$$q\left(\Phi^{G+1}|\Phi^{G}\right) = q\left(\Phi^{G+1}\right) \tag{4.1}$$

Note that now, in general, the formula for the acceptance probability does not simplify and is given by



mytemp

```
mstar2=vec(X0\Y0);  %ols on the appended data
xx=X0'*X0;
ixx2=xx\eye(cols(xx));
 [ beta2,PROBLEM2] = getcoef( mstar2,sigma1,ixx2,MaxTrys,N,L );
   if PROBLEM2
       beta2=beta02;
   else
       beta02=beta2;
   end
%     draw covariance
   e=Y0-X0*reshape(beta2,N*L+1,N);
  scale=e'*e;
  sigma1=iwpQ(rows(Y0),inv(scale));



  %step 4 Sample Threshold via a Random Walk Metropolis Step

  paramnew=paramold+(randn(1,2)*chol(paramvar))';
   %compute conditional posterior at the old and new draw

postnew=getvarpostx(Y,X,beta1,beta2,sigma1,L,[paramnew],tarmean,tarvariance,gammean,gamvariance,Ystar,ncr:

postold=getvarpostx(Y,X,beta1,beta2,sigma1,L,[paramold],tarmean,tarvariance,gammean,gamvariance,Ystar,ncr:

   accept=exp(postnew-postold);
  %end
   u=rand(1,1);
   if u<accept
       paramold=paramnew;
       naccept=naccept+1;
   end
   tar=paramold(1);
   gam=paramold(2);
   arate=naccept/igibbs;

   if igibbs>100 && igibbs<1000
       if arate<0.2
           paramvar=paramvar*0.99999;
       elseif arate>0.5
           paramvar=paramvar*1.01;
       end
   end


  % Display progress:

      disp(sprintf('     Replication %s of %s acceptance %s. ',
num2str(igibbs),num2str(REPS),num2str(arate)))




   if igibbs>BURN && ~PROBLEM1 &&~ PROBLEM2
       jgibbs=jgibbs+1;

 bsave1(jgibbs,:)=beta1';
 bsave2(jgibbs,:)=beta2';
 sigmaS1(jgibbs,:,:)=sigma1;
```

$$\Xi^{new} = \Xi^{old} + e, e\sim N(0,\Sigma)$$

$$\alpha = \frac{F(Y\backslash B_i,\Omega,\Xi^{new})p(\Xi^{new})}{F(Y\backslash B_i,\Omega,\Xi^{old})p(\Xi^{old})}$$

mytemp.html[11/06/2017 16:39:39]

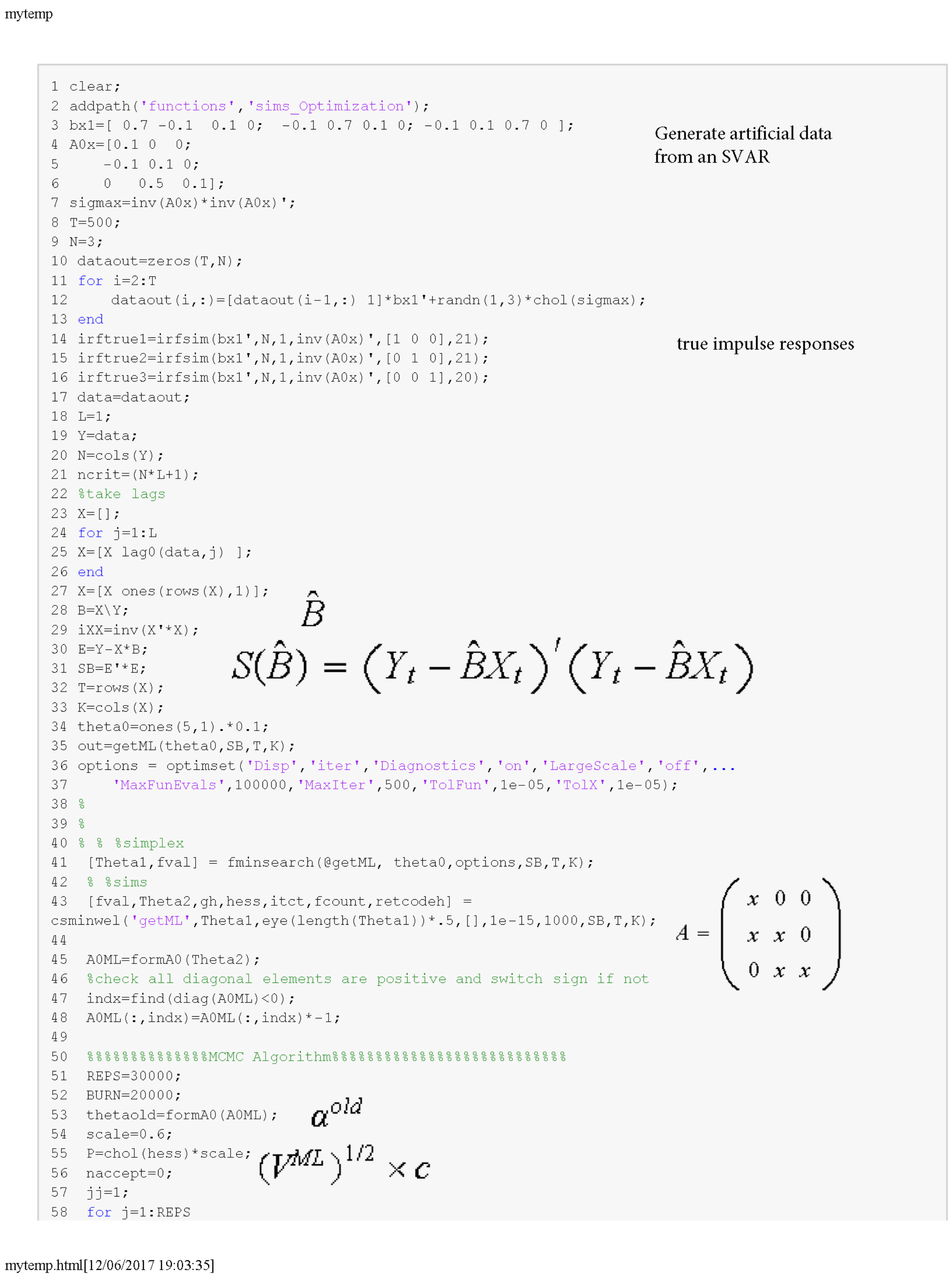
FIGURE 18. Code for the STVAR model

$$\alpha = \min\left(\frac{\pi\left(\Phi^{G+1}\right)/q\left(\Phi^{G+1}|\Phi^{G}\right)}{\pi\left(\Phi^{G}\right)/q\left(\Phi^{G}|\Phi^{G+1}\right)},1\right) \tag{4.2}$$

The independence MH algorithm is therefore more general than the random walk MH algorithm. Unlike the random walk MH algorithm, the candidate generating density in the independence MH algorithm has to be tailored to the particular problem at hand. We examine an application to stochastic volatility models below.

Apart from the change in the form of the candidate generating density the steps of the algorithm remain the same:

Step 1 Set starting values for the model parameters.
Step 2 Draw a candidate value of the parameters $\Phi^{G+1}$from the candidate generating density $q\left(\Phi^{G+1}\right)$



mytemp

```
clear;
addpath('functions','sims_Optimization');
bx1=[ 0.7 -0.1  0.1 0;  -0.1 0.7 0.1 0; -0.1 0.1 0.7 0 ];
A0x=[0.1 0   0;
    -0.1 0.1 0;
    0   0.5  0.1];
sigmax=inv(A0x)*inv(A0x)';
T=500;
N=3;
dataout=zeros(T,N);
for i=2:T
    dataout(i,:)=[dataout(i-1,:) 1]*bx1'+randn(1,3)*chol(sigmax);
end
irftrue1=irfsim(bx1',N,1,inv(A0x)',[1 0 0],21);
irftrue2=irfsim(bx1',N,1,inv(A0x)',[0 1 0],21);
irftrue3=irfsim(bx1',N,1,inv(A0x)',[0 0 1],20);
data=dataout;
L=1;
Y=data;
N=cols(Y);
ncrit=(N*L+1);
%take lags
X=[];
for j=1:L
X=[X lag0(data,j) ];
end
X=[X ones(rows(X),1)];
B=X\Y;
iXX=inv(X'*X);
E=Y-X*B;
SB=E'*E;
T=rows(X);
K=cols(X);
theta0=ones(5,1).*0.1;
out=getML(theta0,SB,T,K);
options = optimset('Disp','iter','Diagnostics','on','LargeScale','off',...
    'MaxFunEvals',100000,'MaxIter',500,'TolFun',1e-05,'TolX',1e-05);
%
%
% % %simplex
 [Theta1,fval] = fminsearch(@getML, theta0,options,SB,T,K);
 % %sims
 [fval,Theta2,gh,hess,itct,fcount,retcodeh] =
csminwel('getML',Theta1,eye(length(Theta1))*.5,[],1e-15,1000,SB,T,K);

 A0ML=formA0(Theta2);
 %check all diagonal elements are positive and switch sign if not
 indx=find(diag(A0ML)<0);
 A0ML(:,indx)=A0ML(:,indx)*-1;

 %%%%%%%%%%%%%%MCMC Algorithm%%%%%%%%%%%%%%%%%%%%%%%%%%%
 REPS=30000;
 BURN=20000;
 thetaold=formA0(A0ML);
 scale=0.6;
 P=chol(hess)*scale;
 naccept=0;
 jj=1;
 for j=1:REPS
```

Generate artificial data from an SVAR

true impulse responses

$\hat{B}$

$$S(\hat{B}) = \left(Y_t - \hat{B}X_t\right)'\left(Y_t - \hat{B}X_t\right)$$

$$A = \begin{pmatrix} x & 0 & 0 \\ x & x & 0 \\ 0 & x & x \end{pmatrix}$$

α^{old}

$\left(V^{ML}\right)^{1/2} \times c$

mytemp.html[12/06/2017 19:03:35]

FIGURE 19. Code for SVAR model

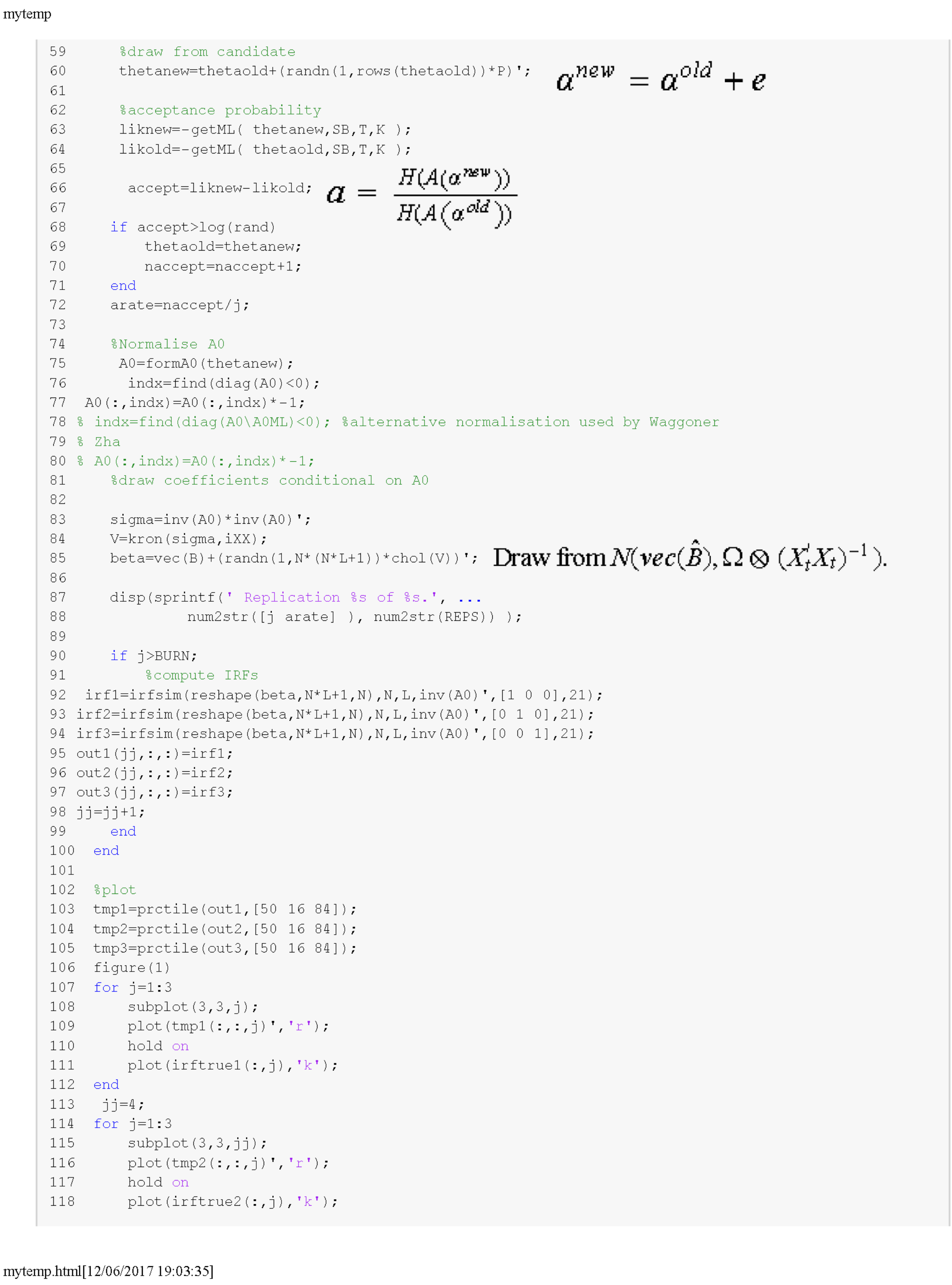
Step 3 Compute the acceptance probability

$$\alpha = \min\left(\frac{\pi\left(\Phi^{G+1}\right)/q\left(\Phi^{G+1}/\Phi^{G}\right)}{\pi\left(\Phi^{G}\right)/q\left(\Phi^{G}/\Phi^{G+1}\right)}, 1\right) \tag{4.3}$$

Step 4 If $u \sim U(0,1)$ is less than α retain Φ^{G+1}. Otherwise retain the old draw.

Step 5 Repeat steps 2 to 4 M times and base inference on the last L draws. In other words, the empirical distribution using the last L draws is an approximation to target density.

4.1. Estimation of stochastic volatility models via the independence MH algorithm. A simple stochastic volatility model for a $T \times 1$ data series y_t is given by



mytemp

```
     %draw from candidate
     thetanew=thetaold+(randn(1,rows(thetaold))*P)';

     %acceptance probability
     liknew=-getML( thetanew,SB,T,K );
     likold=-getML( thetaold,SB,T,K );

      accept=liknew-likold;

    if accept>log(rand)
        thetaold=thetanew;
        naccept=naccept+1;
    end
    arate=naccept/j;

    %Normalise A0
     A0=formA0(thetanew);
      indx=find(diag(A0)<0);
 A0(:,indx)=A0(:,indx)*-1;
% indx=find(diag(A0\A0ML)<0); %alternative normalisation used by Waggoner
% Zha
% A0(:,indx)=A0(:,indx)*-1;
    %draw coefficients conditional on A0

    sigma=inv(A0)*inv(A0)';
    V=kron(sigma,iXX);
    beta=vec(B)+(randn(1,N*(N*L+1))*chol(V))';

    disp(sprintf(' Replication %s of %s.', ...
             num2str([j arate] ), num2str(REPS)) );

    if j>BURN;
        %compute IRFs
 irf1=irfsim(reshape(beta,N*L+1,N),N,L,inv(A0)',[1 0 0],21);
irf2=irfsim(reshape(beta,N*L+1,N),N,L,inv(A0)',[0 1 0],21);
irf3=irfsim(reshape(beta,N*L+1,N),N,L,inv(A0)',[0 0 1],21);
out1(jj,:,:)=irf1;
out2(jj,:,:)=irf2;
out3(jj,:,:)=irf3;
jj=jj+1;
    end
end

%plot
tmp1=prctile(out1,[50 16 84]);
tmp2=prctile(out2,[50 16 84]);
tmp3=prctile(out3,[50 16 84]);
figure(1)
for j=1:3
    subplot(3,3,j);
    plot(tmp1(:,:,j)','r');
    hold on
    plot(irftrue1(:,j),'k');
end
 jj=4;
for j=1:3
    subplot(3,3,jj);
    plot(tmp2(:,:,j)','r');
    hold on
    plot(irftrue2(:,j),'k');
```

$$\alpha^{new} = \alpha^{old} + e$$

$$\alpha = \frac{H(A(\alpha^{new}))}{H(A(\alpha^{old}))}$$

Draw from $N(vec(\hat{B}), \Omega \otimes (X_t'X_t)^{-1})$.

mytemp.html[12/06/2017 19:03:35]

FIGURE 20. Code for the SVAR model

$$\begin{aligned} y_t &= \varepsilon_t \sqrt{\exp(\ln h_t)} \\ \ln h_t &= \ln h_{t-1} + v_t \\ & v_t \tilde{} N(0, g) \end{aligned} \tag{4.4}$$

where h_t is time-varying variance. Note that this is a state space model where the observation equation is non-linear in the state variable h_t and therefore the Carter and Kohn algorithm does not apply. Jacquier *et al.* (2004) instead suggest applying an independence MH algorithm at each point in time to sample from the conditional distribution of h_t which is given by $f(h_t|h_{-t}, y_t)$ where the subscript $-t$ denotes all other dates than t.Jacquier *et al.* (2004) argue that because the transition equation of the model is a random walk, the knowledge of h_{t+1} and h_{t-1} contains all relavant information about h_t. Therefore, the conditional distribution of h_t can be simplified as

$$f\left(h_t|h_{-t}, y_t\right) = f\left(h_t|h_{t-1}, h_{t+1}, y_t\right) \tag{4.5}$$

Jacquier *et al.* (2004) show that this density has the following form

$$f\left(h_t|h_{t-1}, h_{t+1}, y_t\right) = h_t^{-0.5} \exp\left(\frac{-y_t^2}{2h_t}\right) \times h_t^{-1} \exp\left(\frac{-\left(\ln h_t - \mu\right)^2}{2\sigma_h}\right) \tag{4.6}$$

with

$$\mu = \frac{\left(\ln h_{t+1} + \ln h_{t-1}\right)}{2} \tag{4.7}$$

$$\sigma_h = \frac{g}{2} \tag{4.8}$$

That is $f\left(h_t|h_{t-1}, h_{t+1}, y_t\right)$ is a product of a normal density $h_t^{-0.5} \exp\left(\frac{-y_t^2}{2h_t}\right)$ and a log normal density $h_t^{-1} \exp\left(\frac{-(\ln h_t - \mu)^2}{2\sigma_h}\right)$.

To sample from $f\left(h_t|h_{t-1}, h_{t+1}, y_t\right)$, Jacquier *et al.* (2004) suggest a date by date application of the independence MH algorithm with the candidate density defined as the second term in equation 4.6

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-\left(\ln h_t - \mu\right)^2}{2\sigma_h}\right) \tag{4.9}$$

The acceptance probability in this case is given by

$$\alpha = \min\left(\frac{\pi\left(\Phi^{G+1}\right)/q\left(\Phi^{G+1}/\Phi^{G}\right)}{\pi\left(\Phi^{G}\right)/q\left(\Phi^{G}/\Phi^{G+1}\right)}, 1\right) \tag{4.10}$$

$\rightarrow$

$$\alpha = \frac{\left[h_{t,new}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,new}}\right) \times h_{t,new}^{-1} \exp\left(\frac{-(\ln h_{t,new} - \mu)^2}{2\sigma_h}\right)\right] / h_{t,new}^{-1} \exp\left(\frac{-(\ln h_{t,new} - \mu)^2}{2\sigma_h}\right)}{\left[h_{t,old}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,old}}\right) \times h_{t,old}^{-1} \exp\left(\frac{-(\ln h_{t,old} - \mu)^2}{2\sigma_h}\right)\right] / h_{t,old}^{-1} \exp\left(\frac{-(\ln h_{t,old} - \mu)^2}{2\sigma_h}\right)} \tag{4.11}$$

where the subscript *new* denotes the new draw and the subscript *old* denotes the old draw. Equation 4.11 simplifies to give

$$\alpha = \frac{h_{t,new}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,old}}\right)} \tag{4.12}$$

Therefore, for each t one generates a value of h_t using the candidate density in equation 4.9 and then calculates the acceptance probability using equation 4.12. Note however that this algorithm is not operational for the first and the last date in the sample as the calculation of $\mu = \frac{(\ln h_{t+1} + \ln h_{t-1})}{2}$ requires knowledge of h_{t+1} and h_{t-1}.

Jacquier *et al.* (2004) suggest sampling the initial value of h_t denoted by h_0 using the following procedure. Starting with the following prior for $\ln h_0\tilde{}N\left(\bar{\mu}, \bar{\sigma}\right)$ Jacquier *et al.* (2004) show that the posterior for $\ln h_0$ is given by

$$f\left(h_0|h_1\right) = h_0^{-1} \exp\left(\frac{-\left(\ln h_0 - \mu_0\right)^2}{2\sigma_0}\right) \tag{4.13}$$

where

$$\begin{aligned} \sigma_0 &= \frac{\bar{\sigma} g}{\bar{\sigma} + g} \\ \mu_0 &= \sigma_0\left(\frac{\bar{\mu}}{\bar{\sigma}} + \frac{\ln h_1}{g}\right) \end{aligned}$$

Therefore the algorithm starts by sampling h_0 from equation 4.13 and accepting the draw (as the data for this observation y_0 is not defined).

Jacquier *et al.* (2004) suggest sampling the final value of h_t (with $t = T$)using the following modified candidate generating density

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-\left(\ln h_t - \mu\right)^2}{2\sigma_h}\right) \tag{4.14}$$

where

$$\begin{aligned} \mu &= \ln h_{t-1} \qquad (4.15) \\ \sigma_h &= g \end{aligned}$$

The algorithm for the stochastic volatility model consists of the following steps[4]:

[4] Note that the Jacquier *et al.* (2004) algorithm is a single-move algorithm– the stochastic volatility is drawn one period at a time. This may mean that this algorithm requires a large number of draws before convergence occurs. Kim *et al.* (1998) develop an algorithm to sample the entire time-series of the stochastic volatility jointly and show that this multi-move algorithm is more efficient.

Step 1 Obtain a starting value for $h_t, t = 0....T$ as $\hat{\varepsilon}_t^2$ and set the prior $\bar{\mu}, \bar{\sigma}$ (e.g $\bar{\mu}$ could be the log of OLS estimate of the variance of ε_t and $\bar{\sigma}$ could be set to a big number to reflect the uncertainty in this initial guess). Set an inverse Gamma prior for g i.e. $p(g) \sim IG\left(\frac{g_0}{2}, \frac{v_0}{2}\right)$ Set a starting value for g.

Step 2 Time 0 Sample the initial value of h_t denoted by h_0 from the log normal density

$$f(h_0|h_1) = h_0^{-1} \exp\left(\frac{-(\ln h_0 - \mu_0)^2}{2\sigma_0}\right)$$

where the mean $\mu_0 = \sigma_0\left(\frac{\bar{\mu}}{\bar{\sigma}} + \frac{\ln h_1}{g}\right)$ and $\sigma_0 = \frac{\bar{\sigma}g}{\bar{\sigma}+g}$.

Algorithm 5. *To sample from the log normal density $z \sim \log normal(\mu, \sigma)$ sample z_0 from the normal density $N(\mu, \sigma)$. Then $z = \exp(z_0)$.*

ep 2 Time 1 to T-1 For *each date* t=1 to T-1 draw a new value for h_t from the candidate density (call the draw $h_{t,new}$)

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-(\ln h_t - \mu)^2}{2\sigma_h}\right)$$

where $\mu = \frac{(\ln h_{t+1} + \ln h_{t-1})}{2}$ and $\sigma_h = \frac{g}{2}$. Compute the acceptance probability

$$\alpha = \min\left(\frac{h_{t,new}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,old}}\right)}, 1\right)$$

Draw $u\tilde{}U(0,1)$. If $u < \alpha$ set $\boldsymbol{h}_t = h_{t,new}$. Otherwise retain the old draw.

Step 2 Time T For the last time period $t = T$ compute $\mu = \ln h_{t-1}$ and $\sigma_h = g$ and draw $h_{t,new}$ from the candidate density

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-(\ln h_t - \mu)^2}{2\sigma_h}\right)$$

Compute the acceptance probability

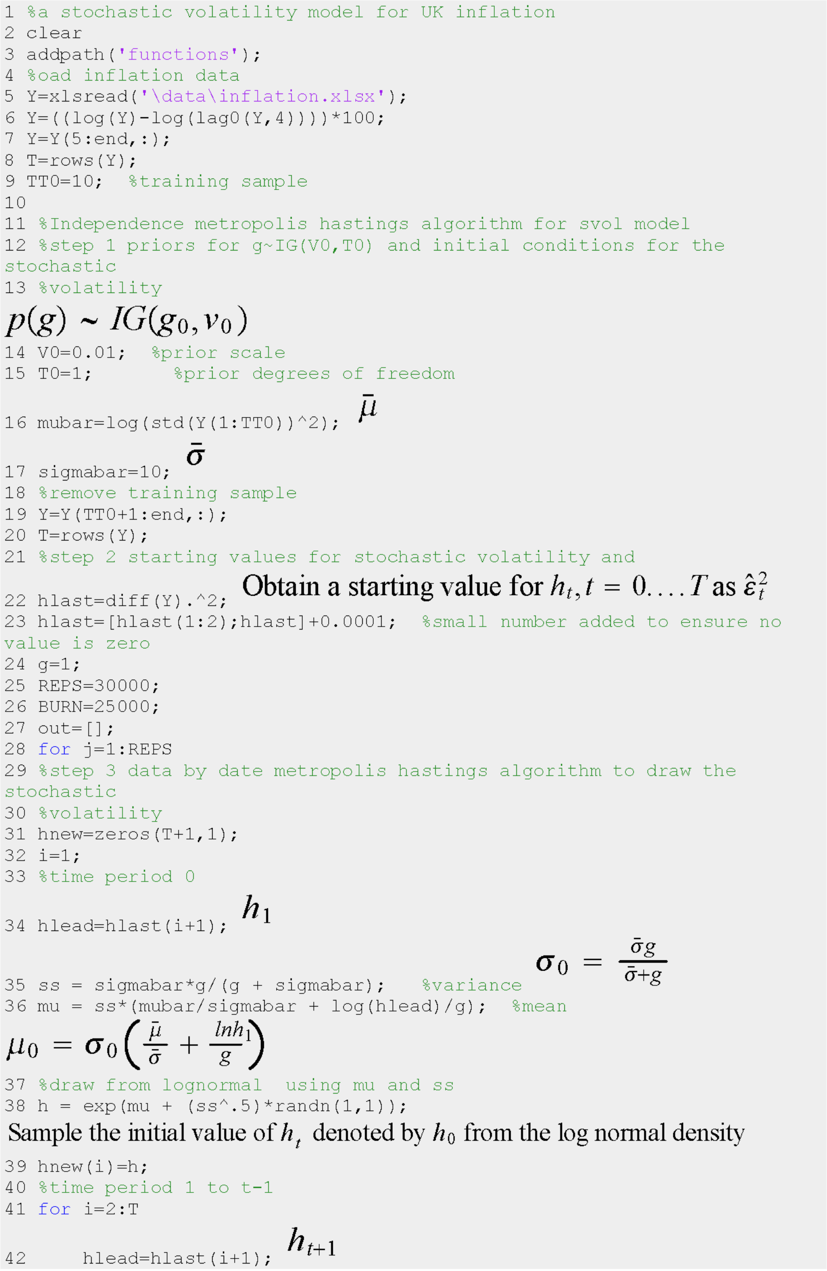
$$\alpha = \min\left(\frac{h_{t,new}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,old}}\right)}, 1\right)$$

Draw $u\tilde{}U(0,1)$. If $u < \alpha$ set $\boldsymbol{h}_t = h_{t,new}$. Otherwise retain the old draw.

Step 3 Given a draw for h_t compute the residuals of the transition equation $v_t = \ln h_t - \ln h_{t-1}$. Draw g from the inverse Gamma distribution with scale parameter $\frac{v_t' v_t + g_0}{2}$ and degrees of freedom $\frac{T+v_0}{2}$. Note that this is an example of a combination of Metropolis and Gibbs sampling algorithms.

Step 4 Repeat steps 2 and 3 M times. The last L draws of h_t and g provide an approximation to the marginal posterior distributions.

Figures 21, 22 and 23 present the Matlab code for the stochastic volatility model applied to annual UK inflation over the period 1914q1 to 2011q4 (example4.m). Lines 14 and 15 of the code set the prior for g. Lines 16 and 17 set the prior $\ln h_0\tilde{}N(\bar{\mu}, \bar{\sigma})$ where $\bar{\mu}$ is set equal to the log of the variance of the first 10 observations in the sample. Line 23 calculates a rough starting value for h_t as the square of the first difference of y_t. Lines 35 and 36 calculate σ_0 and μ_0 and line 38 draws h_0 from the log normal density. Line 41 starts a loop from period 1 to T-1. Note that line 42 selects h_{t+1} as the lead value of h_t using the last draw of h_t. Line 47 and 48 calculate the mean and variance of the candidate density and line 49 draws the candidiate value of h_t. Line 54 calculates the acceptance probability in logs. Lines 68 to 84 repeat this for the final observation in the sample period. Line 84 calculate the residuals of the transition equation as $v_t = \ln h_t - \ln h_{t-1}$. Line 85 draws g from the inverse Gamma distribution.



```
%a stochastic volatility model for UK inflation
clear
addpath('functions');
%oad inflation data
Y=xlsread('\data\inflation.xlsx');
Y=((log(Y)-log(lag0(Y,4))))*100;
Y=Y(5:end,:);
T=rows(Y);
TT0=10;  %training sample

%Independence metropolis hastings algorithm for svol model
%step 1 priors for g~IG(V0,T0) and initial conditions for the stochastic
%volatility
```

$$p(g) \sim IG(g_0, v_0)$$

```
V0=0.01;  %prior scale
T0=1;       %prior degrees of freedom
mubar=log(std(Y(1:TT0))^2);
```

$$\bar{\mu}$$

```
sigmabar=10;
```

$$\bar{\sigma}$$

```
%remove training sample
Y=Y(TT0+1:end,:);
T=rows(Y);
%step 2 starting values for stochastic volatility and
hlast=diff(Y).^2;
```

Obtain a starting value for $h_t, t = 0 \ldots T$ as $\hat{\varepsilon}_t^2$

```
hlast=[hlast(1:2);hlast]+0.0001;  %small number added to ensure no value is zero
g=1;
REPS=30000;
BURN=25000;
out=[];
for j=1:REPS
%step 3 data by date metropolis hastings algorithm to draw the stochastic
%volatility
hnew=zeros(T+1,1);
i=1;
%time period 0
hlead=hlast(i+1);
```

$$h_1$$

```
ss = sigmabar*g/(g + sigmabar);   %variance
```

$$\sigma_0 = \frac{\bar{\sigma} g}{\bar{\sigma} + g}$$

```
mu = ss*(mubar/sigmabar + log(hlead)/g);  %mean
```

$$\mu_0 = \sigma_0 \left(\frac{\bar{\mu}}{\bar{\sigma}} + \frac{\ln h_1}{g} \right)$$

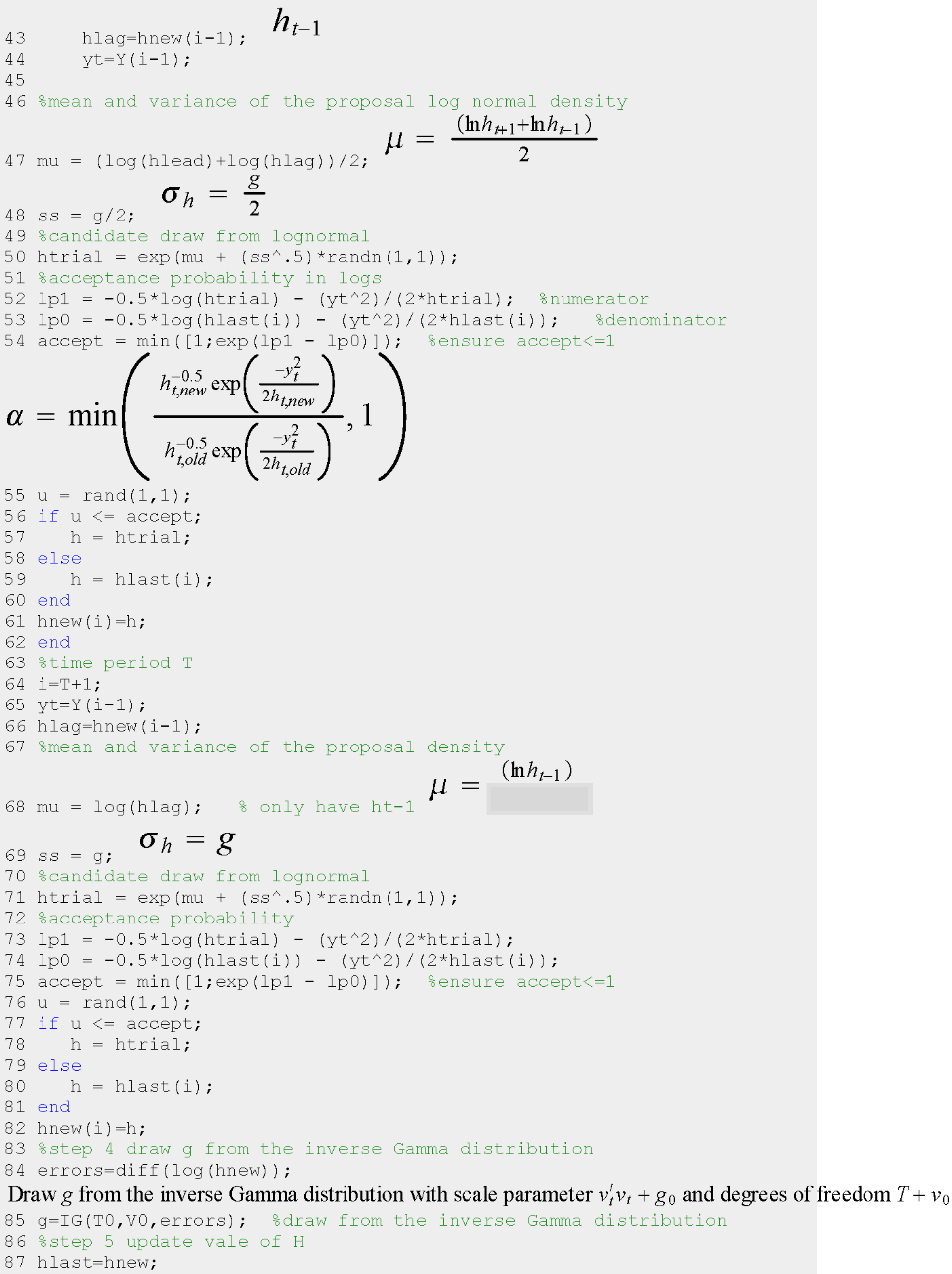
```
%draw from lognormal  using mu and ss
h = exp(mu + (ss^.5)*randn(1,1));
```

Sample the initial value of h_t denoted by h_0 from the log normal density

```
hnew(i)=h;
%time period 1 to t-1
for i=2:T
    hlead=hlast(i+1);
```

$$h_{t+1}$$

Figure 21. Matlab code for the stochastic volatility model



$$h_{t-1}$$

```
    hlag=hnew(i-1);
    yt=Y(i-1);

%mean and variance of the proposal log normal density
```

$$\mu = \frac{(\ln h_{t+1} + \ln h_{t-1})}{2}$$

```
mu = (log(hlead)+log(hlag))/2;
```

$$\sigma_h = \frac{g}{2}$$

```
ss = g/2;
%candidate draw from lognormal
htrial = exp(mu + (ss^.5)*randn(1,1));
%acceptance probability in logs
lp1 = -0.5*log(htrial) - (yt^2)/(2*htrial);  %numerator
lp0 = -0.5*log(hlast(i)) - (yt^2)/(2*hlast(i));   %denominator
accept = min([1;exp(lp1 - lp0)]);  %ensure accept<=1
```

$$\alpha = \min\left(\frac{h_{t,new}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5} \exp\left(\frac{-y_t^2}{2h_{t,old}}\right)}, 1 \right)$$

```
u = rand(1,1);
if u <= accept;
   h = htrial;
else
   h = hlast(i);
end
hnew(i)=h;
end
%time period T
i=T+1;
yt=Y(i-1);
hlag=hnew(i-1);
%mean and variance of the proposal density
```

$$\mu = (\ln h_{t-1})$$

```
mu = log(hlag);   % only have ht-1
```

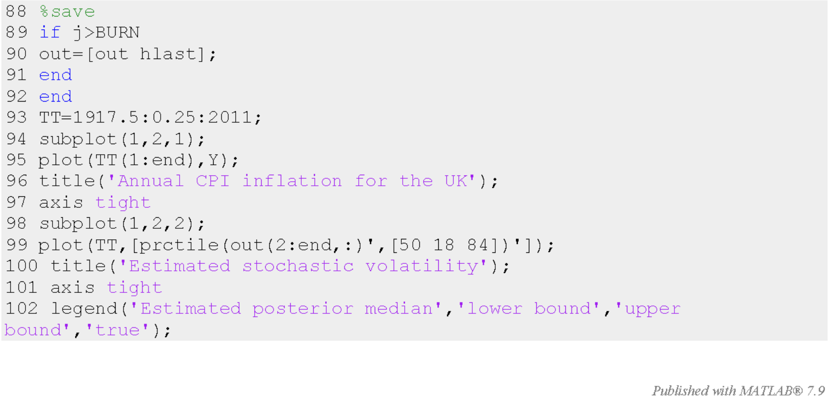
$$\sigma_h = g$$

```
ss = g;
%candidate draw from lognormal
htrial = exp(mu + (ss^.5)*randn(1,1));
%acceptance probability
lp1 = -0.5*log(htrial) - (yt^2)/(2*htrial);
lp0 = -0.5*log(hlast(i)) - (yt^2)/(2*hlast(i));
accept = min([1;exp(lp1 - lp0)]);  %ensure accept<=1
u = rand(1,1);
if u <= accept;
   h = htrial;
else
   h = hlast(i);
end
hnew(i)=h;
%step 4 draw g from the inverse Gamma distribution
errors=diff(log(hnew));
```

Draw g from the inverse Gamma distribution with scale parameter $v_t'v_t + g_0$ and degrees of freedom $T + v_0$

```
g=IG(T0,V0,errors);  %draw from the inverse Gamma distribution
%step 5 update vale of H
hlast=hnew;
```

FIGURE 22. Matlab code for the stochastic volatility model continued



```
%save
if j>BURN
out=[out hlast];
end
end
TT=1917.5:0.25:2011;
subplot(1,2,1);
plot(TT(1:end),Y);
title('Annual CPI inflation for the UK');
axis tight
subplot(1,2,2);
plot(TT,[prctile(out(2:end,:)',[50 18 84])']);
title('Estimated stochastic volatility');
axis tight
legend('Estimated posterior median','lower bound','upper bound','true');
```

Published with MATLAB® 7.9

Figure 23. Matlab code for the stochastic volatility model continued

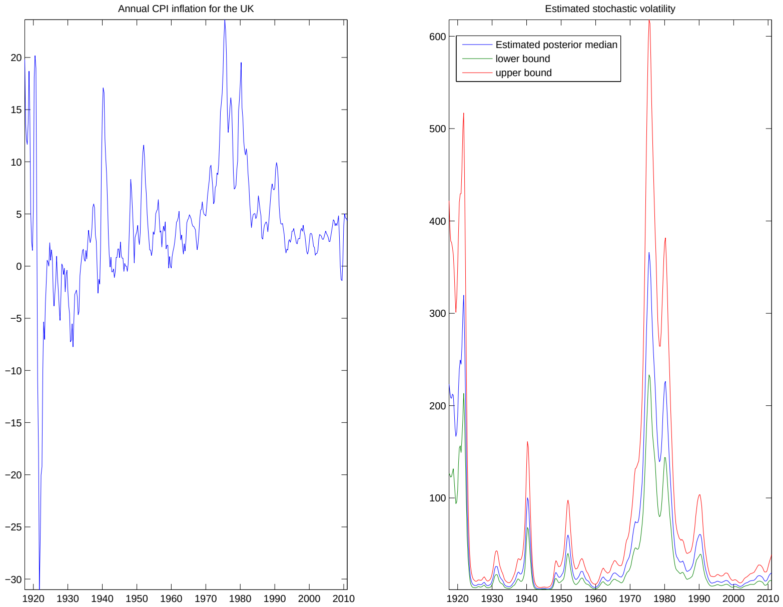


Figure 24. Estimated stochastic volatility of UK inflation

The right panel of figure 24 plots the estimated stochastic volatility of UK inflation.

We now a consider an extended version of this stochastic volatility model for inflation. The model now assumes a time-varying AR(1) specification for inflation with stochastic volatility in the error term. This model is given as

$$y_t = c_t + b_t y_{t-1} + \varepsilon_t \sqrt{\exp\left(\ln h_t\right)} \tag{4.16}$$

Letting $B = \{c, b\}$ the coefficients in the regression evolve as

$$B_t = B_{t-1} + e_t \tag{4.17}$$

where $e_t \sim N(0, Q)$. As before, the variance of the error term h_t evolves as

$$\begin{aligned} \ln h_t &= \ln h_{t-1} + v_t \\ & v_t\tilde{}N(0, g) \end{aligned} \tag{4.18}$$

This model can be easily estimated by combining the Carter and Kohn algorithm with the Metropolis algorithm described above. The steps are as follows:

Step 1 Set a inverse Wishart prior for Q. The prior scale matrix can be set as $Q_0 = k \times Q_{ols} \times T_0$ where T_0 is the length of training sample, Q_{ols} is the variance covariance matrix of B obtained via OLS using the traning sample and k is a scaling factor set to a small number. Obtain a starting value for $h_t, t = 0....T$ as $\hat{\varepsilon}_t^2$ and set the prior $\bar{\mu}, \bar{\sigma}$ (e.g $\bar{\mu}$ could be the log of OLS estimate of the variance of ε_t using the training sample and $\bar{\sigma}$ could be set to a big number to reflect the uncertainty in this initial guess). Set an inverse Gamma prior for g i.e. $p(g) \sim IG\left(g_0, v_0\right)$ Set a starting value for g and Q.

Step 2 Time 0 Conditional on g and B_t sample the initial value of h_t denoted by h_0 from the log normal density

$$f\left(h_0|h_1\right) = h_0^{-1} \exp\left(\frac{-\left(\ln h_0 - \mu_0\right)^2}{2\sigma_0}\right)$$

where the mean $\mu_0 = \sigma_0\left(\frac{\bar{\mu}}{\bar{\sigma}} + \frac{\ln h_1}{g}\right)$ and $\sigma_0 = \frac{\bar{\sigma}g}{\bar{\sigma}+g}$.

ep 2 Time 1 to T-1 For *each date* t=1 to T-1 draw a new value for h_t (conditional on g and B_t) from the candidate density (call the draw $h_{t,new}$)

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-\left(\ln h_t - \mu\right)^2}{2\sigma_h}\right)$$

where $\mu = \frac{(\ln h_{t+1} + \ln h_{t-1})}{2}$ and $\sigma_h = \frac{g}{2}$. Compute the acceptance probability (note that the residuals ε_t are used in the expression below rather than y_t as in the previous example)

$$\alpha = \min\left(\frac{h_{t,new}^{-0.5} \exp\left(\frac{-\varepsilon_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5} \exp\left(\frac{-\varepsilon_t^2}{2h_{t,old}}\right)}, 1\right)$$

Draw $u\tilde{}U(0,1)$. If $u < \alpha$ set $\boldsymbol{h}_t = h_{t,new}$. Otherwise retain the old draw.

Step 2 Time T For the last time period $t = T$ compute $\mu = \ln h_{t-1}$ and $\sigma_h = g$ and draw $h_{t,new}$ from the candidate density

$$q\left(\Phi^{G+1}\right) = h_t^{-1} \exp\left(\frac{-\left(\ln h_t - \mu\right)^2}{2\sigma_h}\right)$$

Compute the acceptance probability

$$\alpha = \min\left(\frac{h_{t,new}^{-0.5}\exp\left(\frac{-\varepsilon_t^2}{2h_{t,new}}\right)}{h_{t,old}^{-0.5}\exp\left(\frac{-\varepsilon_t^2}{2h_{t,old}}\right)}, 1\right)$$

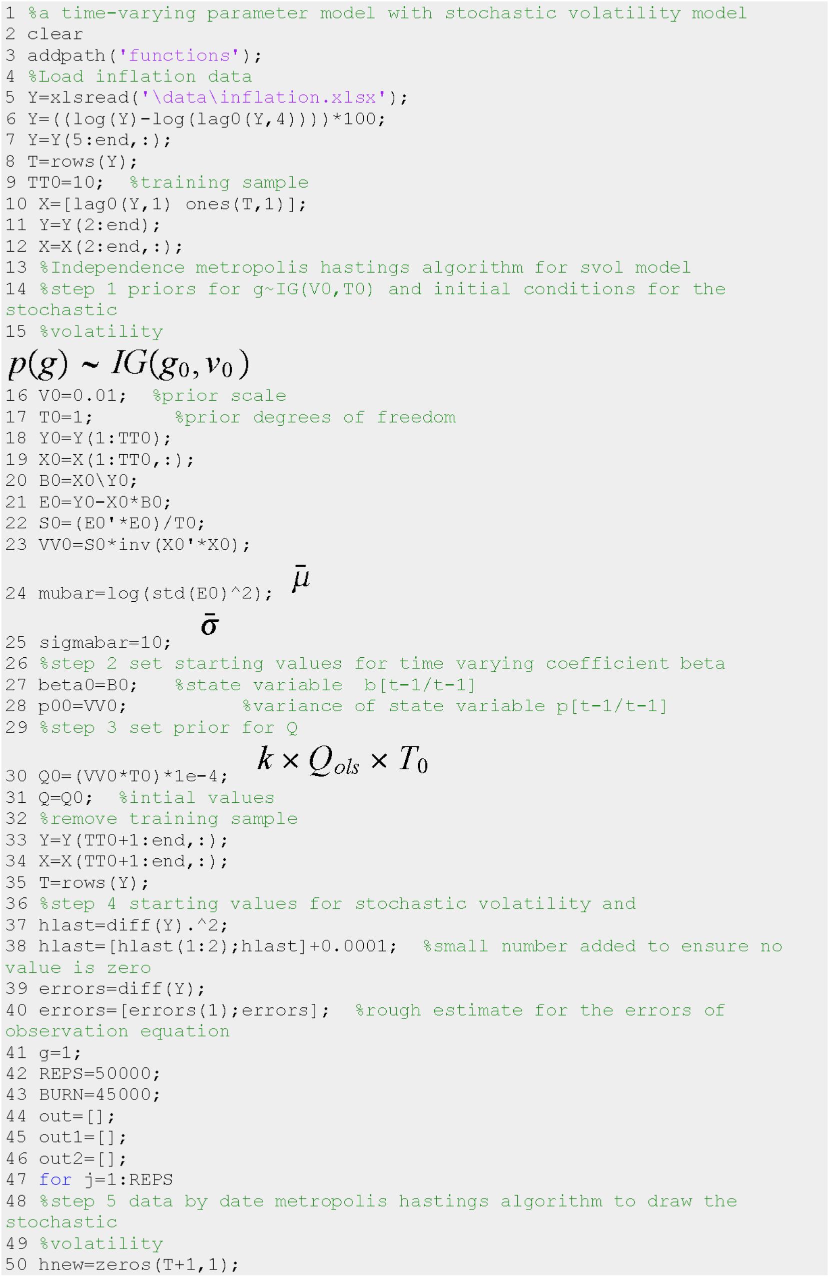
Draw $u\tilde{}U(0,1)$. If $u < \alpha$ set $\boldsymbol{h}_t = h_{t,new}$. Otherwise retain the old draw.

Step 3 Given a draw for h_t compute the residuals of the transition equation $v_t = \ln h_t - \ln h_{t-1}$. Draw g from the inverse Gamma distribution with scale parameter $\frac{v_t' v_t + g_0}{2}$ and degrees of freedom $\frac{T+v_0}{2}$. Note that this is an example of a combination of Metropolis and Gibbs sampling algorithms.

Step 4 Conditional on h_t and Q sample B_t using the Carter and Kohn algorithm as described in Chapter 3. This algorithm remains apart from the minor difference that the variance of the error to observation equation is different at each point in time. This is easily incorporated into the Kalman filter by selecting the appropriate variance at each point in time.

Step 5 Sample Q from the inverse Wishart distribution (conditional on B_t) with scale matrix $(B_t - B_{t-1})'(B_t - B_{t-1}) + Q_0$ and degrees of freedom $T_0 + T$.

Step 6 Repeat steps 2 and 5 M times. The last L draws of h_t, g, B_t and Q provide an approximation to the marginal posterior distributions.



```
1 %a time-varying parameter model with stochastic volatility model
2 clear
3 addpath('functions');
4 %Load inflation data
5 Y=xlsread('\data\inflation.xlsx');
6 Y=((log(Y)-log(lag0(Y,4))))*100;
7 Y=Y(5:end,:);
8 T=rows(Y);
9 TT0=10;  %training sample
10 X=[lag0(Y,1) ones(T,1)];
11 Y=Y(2:end);
12 X=X(2:end,:);
13 %Independence metropolis hastings algorithm for svol model
14 %step 1 priors for g~IG(V0,T0) and initial conditions for the
stochastic
15 %volatility
```

$$p(g) \sim IG(g_0, v_0)$$

```
16 V0=0.01;  %prior scale
17 T0=1;       %prior degrees of freedom
18 Y0=Y(1:TT0);
19 X0=X(1:TT0,:);
20 B0=X0\Y0;
21 E0=Y0-X0*B0;
22 S0=(E0'*E0)/T0;
23 VV0=S0*inv(X0'*X0);
```

$\bar{\mu}$

```
24 mubar=log(std(E0)^2);
```

$\bar{\sigma}$

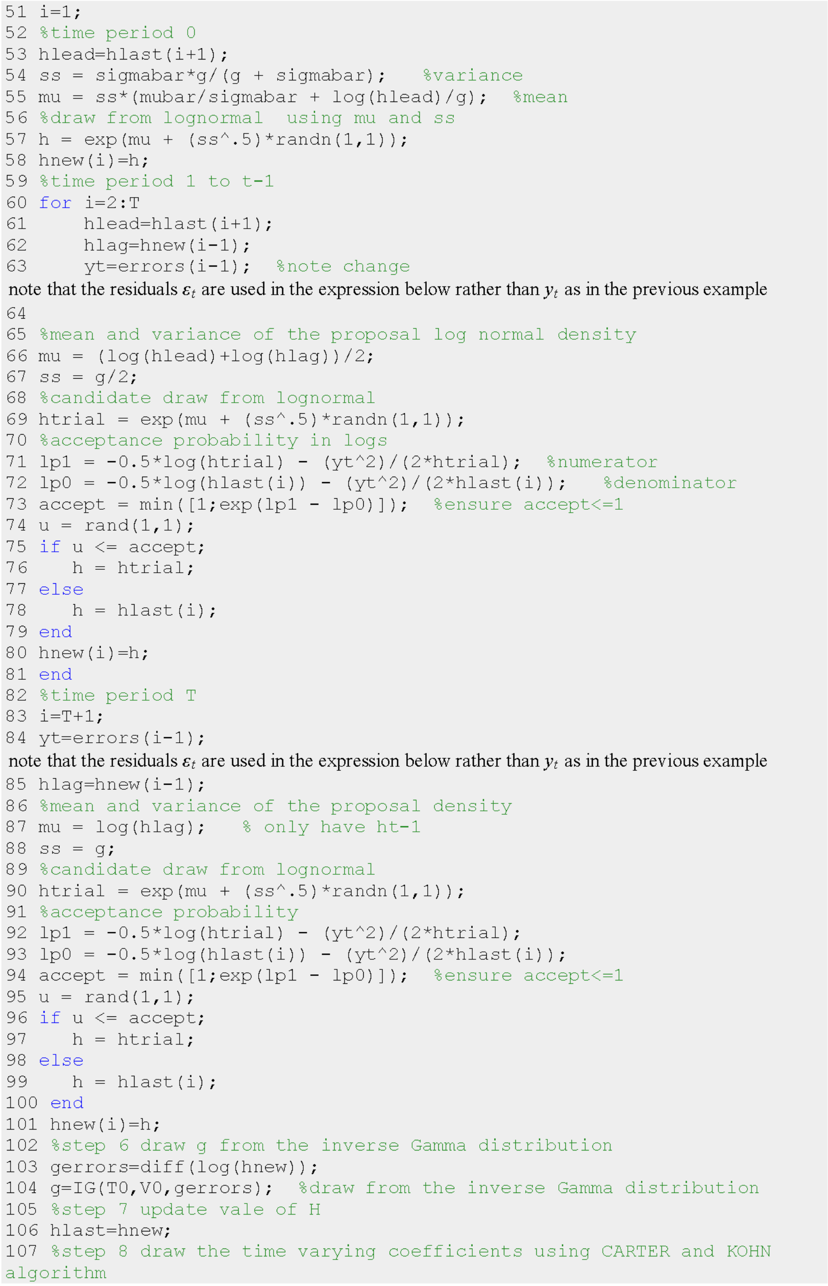
```
25 sigmabar=10;
26 %step 2 set starting values for time varying coefficient beta
27 beta0=B0;   %state variable  b[t-1/t-1]
28 p00=VV0;          %variance of state variable p[t-1/t-1]
29 %step 3 set prior for Q
```

$$k \times Q_{ols} \times T_0$$

```
30 Q0=(VV0*T0)*1e-4;
31 Q=Q0;  %intial values
32 %remove training sample
33 Y=Y(TT0+1:end,:);
34 X=X(TT0+1:end,:);
35 T=rows(Y);
36 %step 4 starting values for stochastic volatility and
37 hlast=diff(Y).^2;
38 hlast=[hlast(1:2);hlast]+0.0001;  %small number added to ensure no
value is zero
39 errors=diff(Y);
40 errors=[errors(1);errors];  %rough estimate for the errors of
observation equation
41 g=1;
42 REPS=50000;
43 BURN=45000;
44 out=[];
45 out1=[];
46 out2=[];
47 for j=1:REPS
48 %step 5 data by date metropolis hastings algorithm to draw the
stochastic
49 %volatility
50 hnew=zeros(T+1,1);
```

Figure 25. Matlab code for the time-varying parameter AR model with stochastic volatility

The matlab code for this example (example5.m) is shown in figures 25, 26 and 27. Lines 18 to 23 of the code estimate an AR(1) model via OLS on a training sample of 10 observations. Line 24 sets $\bar{\mu}$ as the log of the error variance using this OLS residuals. Lines 27 and 28 set the initial value of the time varying coefficients and the associated variance as the OLS estimates. Line 30 sets the prior scale matrix Q_0 using the OLS estimate of the coefficient covariance. Lines 37 to 40 set an initial value for h_t and ε_t and line 47 starts the algorithm. Lines 48 to 101 sample h_t using the independence MH step described in the previous example. The only change is that the residuals from the observation equation ε_t are used to evaluate the densities $h_{t,new}^{-0.5} \exp\left(\frac{-\varepsilon_t^2}{2h_{t,new}}\right)$ and $h_{t,old}^{-0.5} \exp\left(\frac{-\varepsilon_t^2}{2h_{t,new}}\right)$ when calculating the acceptance probability. Line 104 samples g from the inverse Gamma distribution. Line 108 samples

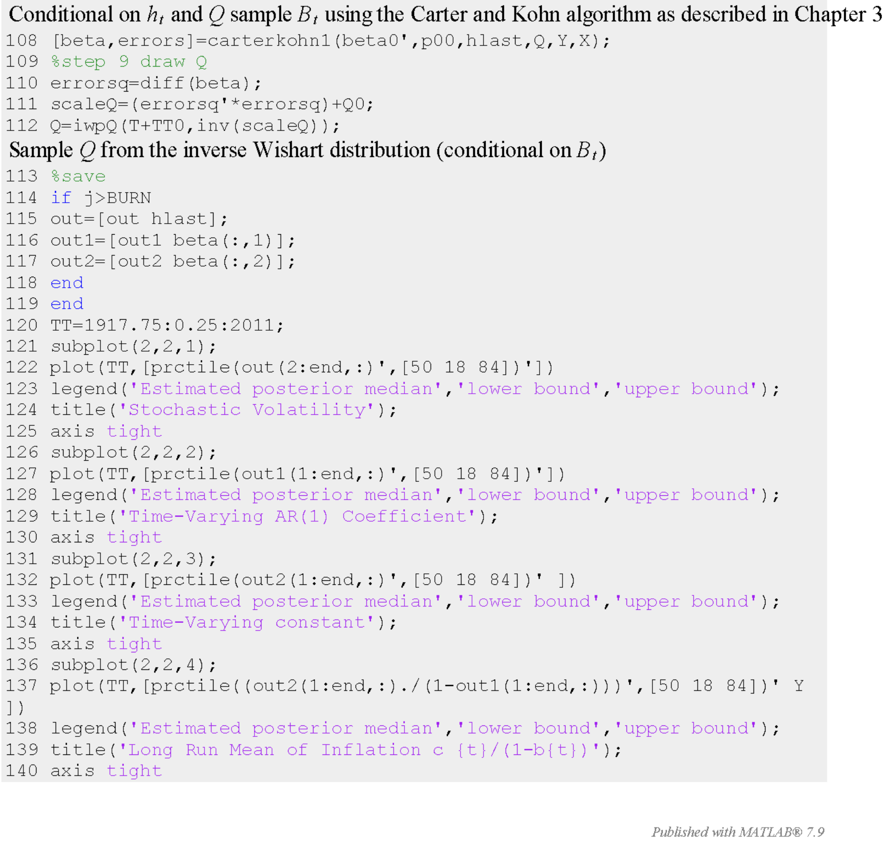


```
51 i=1;
52 %time period 0
53 hlead=hlast(i+1);
54 ss = sigmabar*g/(g + sigmabar);   %variance
55 mu = ss*(mubar/sigmabar + log(hlead)/g);  %mean
56 %draw from lognormal  using mu and ss
57 h = exp(mu + (ss^.5)*randn(1,1));
58 hnew(i)=h;
59 %time period 1 to t-1
60 for i=2:T
61     hlead=hlast(i+1);
62     hlag=hnew(i-1);
63     yt=errors(i-1);  %note change
note that the residuals εt are used in the expression below rather than yt as in the previous example
64
65 %mean and variance of the proposal log normal density
66 mu = (log(hlead)+log(hlag))/2;
67 ss = g/2;
68 %candidate draw from lognormal
69 htrial = exp(mu + (ss^.5)*randn(1,1));
70 %acceptance probability in logs
71 lp1 = -0.5*log(htrial) - (yt^2)/(2*htrial);  %numerator
72 lp0 = -0.5*log(hlast(i)) - (yt^2)/(2*hlast(i));   %denominator
73 accept = min([1;exp(lp1 - lp0)]);  %ensure accept<=1
74 u = rand(1,1);
75 if u <= accept;
76    h = htrial;
77 else
78    h = hlast(i);
79 end
80 hnew(i)=h;
81 end
82 %time period T
83 i=T+1;
84 yt=errors(i-1);
note that the residuals εt are used in the expression below rather than yt as in the previous example
85 hlag=hnew(i-1);
86 %mean and variance of the proposal density
87 mu = log(hlag);   % only have ht-1
88 ss = g;
89 %candidate draw from lognormal
90 htrial = exp(mu + (ss^.5)*randn(1,1));
91 %acceptance probability
92 lp1 = -0.5*log(htrial) - (yt^2)/(2*htrial);
93 lp0 = -0.5*log(hlast(i)) - (yt^2)/(2*hlast(i));
94 accept = min([1;exp(lp1 - lp0)]);  %ensure accept<=1
95 u = rand(1,1);
96 if u <= accept;
97    h = htrial;
98 else
99    h = hlast(i);
100 end
101 hnew(i)=h;
102 %step 6 draw g from the inverse Gamma distribution
103 gerrors=diff(log(hnew));
104 g=IG(T0,V0,gerrors);  %draw from the inverse Gamma distribution
105 %step 7 update vale of H
106 hlast=hnew;
107 %step 8 draw the time varying coefficients using CARTER and KOHN
algorithm
```

FIGURE 26. Matlab code for the time-varying parameter AR model with stochastic volatility continued

the time-varying coefficients using the Carter Kohn algorithm. For simplicity, the code for this algorithm is moved into a seperate function carterkohn1.m saved in the functions folder. This code is identical to the examples discussed in the previous chapter apart from the minor difference that the value of the variance of the errors of the observation equation at time t is set to h_t. See line 18 in carterkohn1.m. Note that this function also returns the updated value of the error term ε_t. The inputs to this function are as follows: (1) the initial state $B_{0|0}$ (2) Variance of the initial state (3) the time-varying variance of shock to the observation equation h_t (4) Q (5) Y_t the dependent variable and (6) X_t the independent variables. Conditional on a value for B_t line 112 samples Q from the inverse Wishart distribution.

Figure 28 shows the estimated stochastic volatility and the time-varying coefficients.



Conditional on h_t and Q sample B_t using the Carter and Kohn algorithm as described in Chapter 3

```
[beta,errors]=carterkohn1(beta0',p00,hlast,Q,Y,X);
%step 9 draw Q
errorsq=diff(beta);
scaleQ=(errorsq'*errorsq)+Q0;
Q=iwpQ(T+TT0,inv(scaleQ));
```

Sample Q from the inverse Wishart distribution (conditional on B_t)

```
%save
if j>BURN
out=[out hlast];
out1=[out1 beta(:,1)];
out2=[out2 beta(:,2)];
end
end
TT=1917.75:0.25:2011;
subplot(2,2,1);
plot(TT,[prctile(out(2:end,:)',[50 18 84])'])
legend('Estimated posterior median','lower bound','upper bound');
title('Stochastic Volatility');
axis tight
subplot(2,2,2);
plot(TT,[prctile(out1(1:end,:)',[50 18 84])'])
legend('Estimated posterior median','lower bound','upper bound');
title('Time-Varying AR(1) Coefficient');
axis tight
subplot(2,2,3);
plot(TT,[prctile(out2(1:end,:)',[50 18 84])' ])
legend('Estimated posterior median','lower bound','upper bound');
title('Time-Varying constant');
axis tight
subplot(2,2,4);
plot(TT,[prctile((out2(1:end,:)./(1-out1(1:end,:)))',[50 18 84])' Y
])
legend('Estimated posterior median','lower bound','upper bound');
title('Long Run Mean of Inflation c {t}/(1-b{t})');
axis tight
```

Published with MATLAB® 7.9

FIGURE 27. Matlab code for the time-varying parameter AR model with stochastic volatility continued

5. A VAR with time-varying coefficients and stochastic volatility

We re-examine an extended version of the time-varying parameter VAR model shown in the previous chapter. The extension involves allowing the variance covariance matrix of the error terms to be time-varying. This model has been used in several recent studies (see for e.g. Primiceri (2005)) and is especially suited to examining the time-varying transmission of structural shocks to the economy.

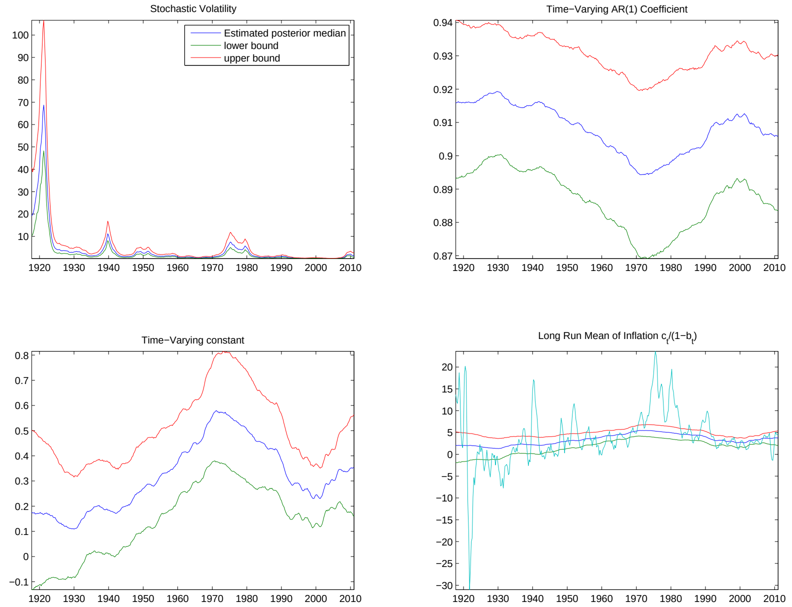


FIGURE 28. Estimates from the time-varying AR model with stochastic volatility

We consider the following VAR model with time-varying parameters

$$
\begin{aligned}
Y_t &= c_t + \sum_{j=1}^{P} B_{j,t} Y_{t-j} + v_t, VAR\left(v_t\right) = R_t \qquad (5.1)\\
\beta_t &= \{c_t, B_{1,t} .... B_{P,t}\}\\
\beta_t &= \beta_{t-1} + e_t, VAR(e_t) = Q
\end{aligned}
$$

The covariance matrix of the error term v_t i.e. R_t has time-varying elements. For simplicity most studies consider the following structure for R_t

$$R_t = A_t^{-1} H_t A_t^{-1\prime} \qquad (5.2)$$

where A_t is a lower triangular matrix with elements $a_{ij,t}$ and H_t is a diagonal matrix with diagonal elements $h_{i,t}$. For example for a three variable VAR

$$A_t = \begin{pmatrix} 1 & 0 & 0 \\ a_{12,t} & 1 & 0 \\ a_{13,t} & a_{23,t} & 1 \end{pmatrix}, H_t = \begin{bmatrix} h_{1,t} & 0 & 0 \\ 0 & h_{2,t} & 0 \\ 0 & 0 & h_{3,t} \end{bmatrix}$$

where

$$a_{ij,t} = a_{ij,t-1} + V_t, VAR(V_t) = D$$

and

$$\ln h_{i,t} = \ln h_{i,t-1} + z_{i,t}, VAR(z_{i,t}) = g_i$$

for $i = 1..3$. Therefore, this model has two sets of time varying 'coefficients' β_t and $a_{ij,t}$ and a stochastic volatility model for the diagonal elements $h_{i,t}$. As in the previous example, this VAR model can be estimated by combining the Carter and Kohn algorithm to draw β_t and $a_{ij,t}$ with the independence MH algorithm for the stochastic volatility. Before we describe the algorithm, it is worth noting the following relationship

$$A_t v_t = \varepsilon_t \qquad (5.3)$$

where $VAR\left(\varepsilon_t\right) = H_t$. For a three variable VAR this relationship implies the following set of equations

$$\begin{pmatrix} 1 & 0 & 0 \\ a_{12,t} & 1 & 0 \\ a_{13,t} & a_{23,t} & 1 \end{pmatrix} \begin{pmatrix} v_{1,t} \\ v_{2,t} \\ v_{3,t} \end{pmatrix} = \begin{pmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \\ \varepsilon_{3,t} \end{pmatrix} \qquad (5.4)$$

or expanding

$$
\begin{aligned}
v_{1,t} &= \varepsilon_{1,t} \qquad (5.5)\\
v_{2,t} &= -a_{12,t} v_{1,t} + \varepsilon_{2,t}\\
v_{3,t} &= -a_{13,t} v_{1,t} - a_{23,t} v_{2,t} + \varepsilon_{3,t}
\end{aligned}
$$

where $VAR(\varepsilon_{2,t}) = h_{2t}$ and $VAR(\varepsilon_{3,t}) = h_{3t}$ and

$$a_{12,t} = a_{12,t-1} + V_{1t}, VAR(V_{1t}) = D_1 \tag{5.6}$$

$$\begin{pmatrix} a_{13,t} \\ a_{23,t} \end{pmatrix} = \begin{pmatrix} a_{13,t-1} \\ a_{23,t-1} \end{pmatrix} + \begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}, VAR(\begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}) = D_2 \tag{5.7}$$

Therefore, $a_{ij,t}$ are time varying coefficients on regressions involving the VAR residuals and can be sampled using the method described in the previous example. The Gibbs and MH algorithm for estimating this three variable time-varying VAR model consists of the following steps

Step 1a Set a prior for Q and starting values for the Kalman filter. The prior for Q is inverse Wishart $p(Q) \sim IW(Q_0, T_0)$. Note that this prior is quite crucial as it influences the amount of time-variation allowed for in the VAR model. In other words, a large value for the scale matrix Q_0 would imply more fluctuation in β_t. This prior is typically set using a training sample. The first T_0 observations of the sample are used to estimate a standard fixed coefficient VAR via OLS such that $\beta_0 = (X_{0t}' X_{0t})^{-1} (X_{0t}' Y_{0t})$ with a coefficient covariance matrix given by $p_{0|0} = \Sigma_0 \otimes (X_{0t}' X_{0t})^{-1}$ where $X_{0t} = \{Y_{0t-1}, ...Y_{0t-p}, 1\}$, $\Sigma_0 = \frac{(Y_{0t} - X_{0t}\beta_0)'(Y_{0t} - X_{0t}\beta_0)}{T_0 - K}$ and the subscript 0 denotes the fact that this is the training sample. The scale matrix Q_0 is set equal to $p_{0|0} \times T_0 \times \tau$ where τ is a scaling factor chosen by the researcher. Some studies set $\tau = 3.510^{-4}$ i.e. a small number to reflect the fact that the training sample in typically short and the resulting estimates of $p_{0|0}$ maybe imprecise. Note that one can control the apriori amount of time-variation in the model by varying τ. Set a starting value for Q. The initial state is set equal to $\beta_{0|0} = vec(\ \beta_0)'$ and the intial state covariance is given by $p_{0|0}$.

Step 1b Set the prior for D_1 and D_2. The prior for D_1 is inverse Gamma $p(D_1) \sim IG(D_{10}, T_0)$ and the prior for D_2 is inverse Wishart $p(D_2) \sim IW(D_{20}, T_0)$. Benati and Mumtaz (2006) set $D_{10} = 0.001$ and $D_{20} = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix}$. Let $C = \Sigma_0^{1/2}$ and let $C0$ denote the inverse of the matrix C with the diagonal normalised to 1. The initial values for $a_{ij,t}$ (i.e. the initial state $a_{ij,0|0}$) are the non-zero elements of $C0$ with the variance of the initial state set equal to $abs(a_{ij}) \times 10$ (as in Benati and Mumtaz (2006)). Set a starting value for $a_{ij,t}$.

Step 1c Obtain a starting value for $h_{i,t}, t = 0....T$ and $i = 1..3$ as $\hat{v}_{it}^2$ and set the prior $\bar{\mu}_i, \bar{\sigma}$. $\bar{\mu}_i$ can be set equal to the log of the ith diagonal element of Σ_0 and $\bar{\sigma}$ to a large number. Set an inverse Gamma prior for g_i i.e. $p(g_i) \sim IG(g_0, v_0)$. Set a starting value for g_i.

Step 2 Conditional on A_t, H_t and Q draw β_t using the Carter and Kohn algorithm. The algorithm exactly as described for the time-varying VAR without stochastic volatility in Chapter 3 with the difference that the variance of v_t changes at each point in time and this needs to be taken into account when running the Kalman filter.

Step 3 Using the draw for β_t calculate the residuals of the transition equation $\beta_t - \beta_{t-1} = e_t$ and sample Q from the inverse Wishart distribution using the scale matrix $e_t' e_t + Q_0$ and degrees of freedom $T + T_0$.

Step 4 Draw $a_{ij,t}$ the elements of A_t using the Carter and Kohn algorithm (conditional on β_t, H_t, D_1 and D_2). The state space formulation for $a_{12,t}$ is

$$\begin{aligned} v_{2,t} &= -a_{12,t}v_{1,t} + \varepsilon_{2,t}, VAR(\varepsilon_{2,t}) = h_{2,t} \\ a_{12,t} &= a_{12,t-1} + V_{1t}, VAR(V_{1t}) = D_1 \end{aligned}$$

The state space formulation for $a_{13,t}$ and $a_{23,t}$ is

$$\begin{aligned} v_{3,t} &= -a_{13,t}v_{1,t} - a_{23,t}v_{2,t} + \varepsilon_{3,t}, VAR(\varepsilon_{3,t}) = h_{3,t} \\ \begin{pmatrix} a_{13,t} \\ a_{23,t} \end{pmatrix} &= \begin{pmatrix} a_{13,t-1} \\ a_{23,t-1} \end{pmatrix} + \begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}, VAR(\begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}) = D_2 \end{aligned}$$

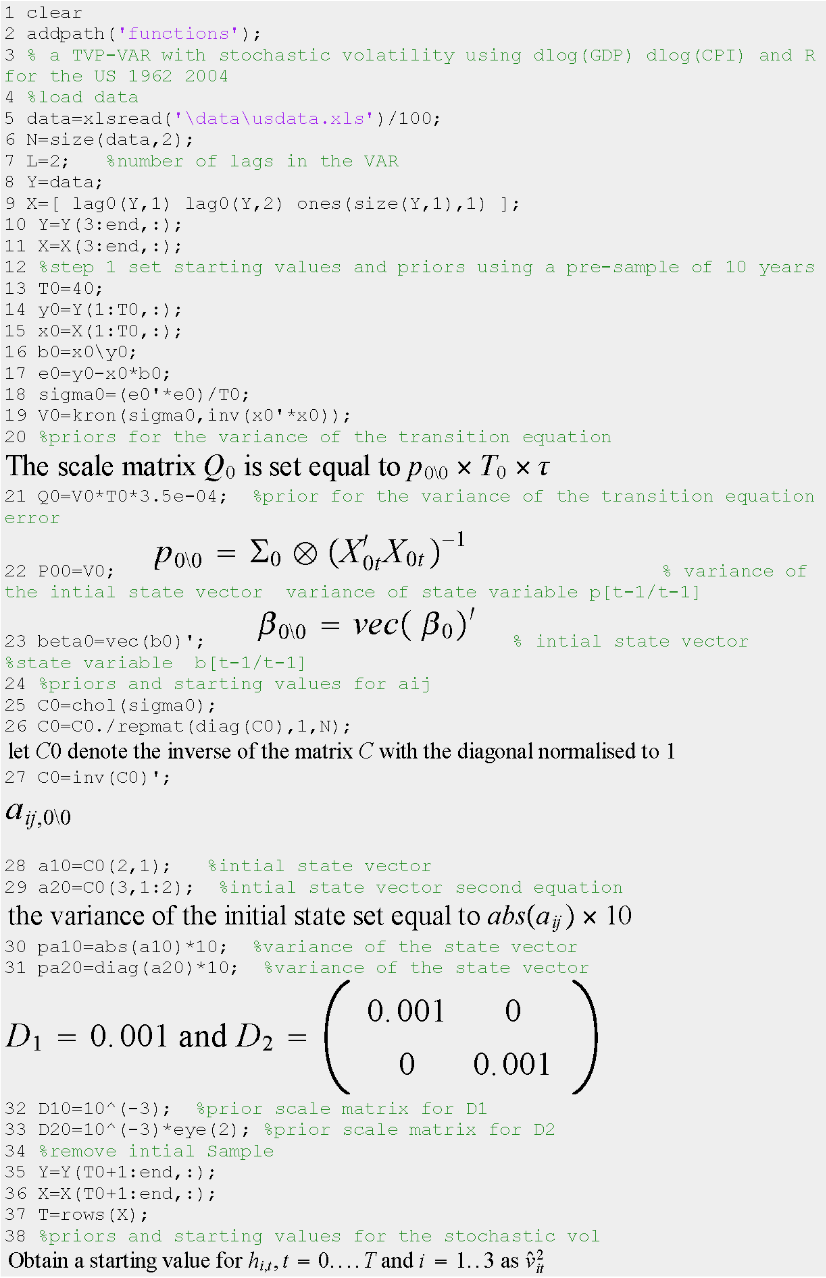
Note that these two formulations are just time-varying regressions in the residuals and the Carter and Kohn algorithm is applied to each seperately to draw $a_{12,t}, a_{13,t}$ and $a_{23,t}$.

Step 5. Conditional on a draw for $a_{12,t}, a_{13,t}$ and $a_{23,t}$ calculate the residuals V_{1t}, V_{2t} and V_{3t}. Draw D_1 from the inverse Gamma distribution with scale parameter $\frac{V_{1t}'V_{1t} + D_{1,0}}{2}$ and degrees of freedom $\frac{T+T_0}{2}$. Draw D_2 from the inverse Wishart distribution with scale matrix $V_{2t}'V_{2t} + D_{2,0}$ and degrees of freedom $T + T_0$.

Step 6 Using the draw of A_t from step 4 calculate $\varepsilon_t = A_t v_t$ where $\varepsilon_t = \begin{pmatrix} \varepsilon_{1,t} \\ \varepsilon_{2,t} \\ \varepsilon_{3,t} \end{pmatrix}$. Note that ε_t are contemporaneously uncorrelated. We can therefore draw $h_{i,t}$ for $i = 1..3$ separately by simply applying the independence MH algorithm described above for each ε_t (conditional on a draw for g_i).

Step 7 Conditional on a draw for $h_{i,t}$ for $i = 1..3$ draw g_i from the inverse Gamma distribution with scale parameter $\frac{(\ln h_{i,t} - \ln h_{i,t-1})'(\ln h_{i,t} - \ln h_{i,t-1}) + g_0}{2}$ and degrees of freedom $\frac{T+v_0}{2}$.

Step 8 Repeat steps 2 and 7 M times. The last L draws provide an approximation to the marginal posterior distributions of the model parameters.



```
1 clear
2 addpath('functions');
3 % a TVP-VAR with stochastic volatility using dlog(GDP) dlog(CPI) and R
for the US 1962 2004
4 %load data
5 data=xlsread('\data\usdata.xls')/100;
6 N=size(data,2);
7 L=2;   %number of lags in the VAR
8 Y=data;
9 X=[ lag0(Y,1) lag0(Y,2) ones(size(Y,1),1) ];
10 Y=Y(3:end,:);
11 X=X(3:end,:);
12 %step 1 set starting values and priors using a pre-sample of 10 years
13 T0=40;
14 y0=Y(1:T0,:);
15 x0=X(1:T0,:);
16 b0=x0\y0;
17 e0=y0-x0*b0;
18 sigma0=(e0'*e0)/T0;
19 V0=kron(sigma0,inv(x0'*x0));
20 %priors for the variance of the transition equation
```

The scale matrix Q_0 is set equal to $p_{0\backslash 0} \times T_0 \times \tau$

```
21 Q0=V0*T0*3.5e-04;  %prior for the variance of the transition equation
error
```

$$p_{0\backslash 0} = \Sigma_0 \otimes (X'_{0t}X_{0t})^{-1}$$

```
22 P00=V0;                                              % variance of
the intial state vector  variance of state variable p[t-1/t-1]
```

$$\beta_{0\backslash 0} = vec(\ \beta_0)'$$

```
23 beta0=vec(b0)';                       % intial state vector
%state variable  b[t-1/t-1]
24 %priors and starting values for aij
25 C0=chol(sigma0);
26 C0=C0./repmat(diag(C0),1,N);
```

let $C0$ denote the inverse of the matrix C with the diagonal normalised to 1

```
27 C0=inv(C0)';
```

$a_{ij,0\backslash 0}$

```
28 a10=C0(2,1);   %intial state vector
29 a20=C0(3,1:2);  %intial state vector second equation
```

the variance of the initial state set equal to $abs(a_{ij}) \times 10$

```
30 pa10=abs(a10)*10;  %variance of the state vector
31 pa20=diag(a20)*10;  %variance of the state vector
```

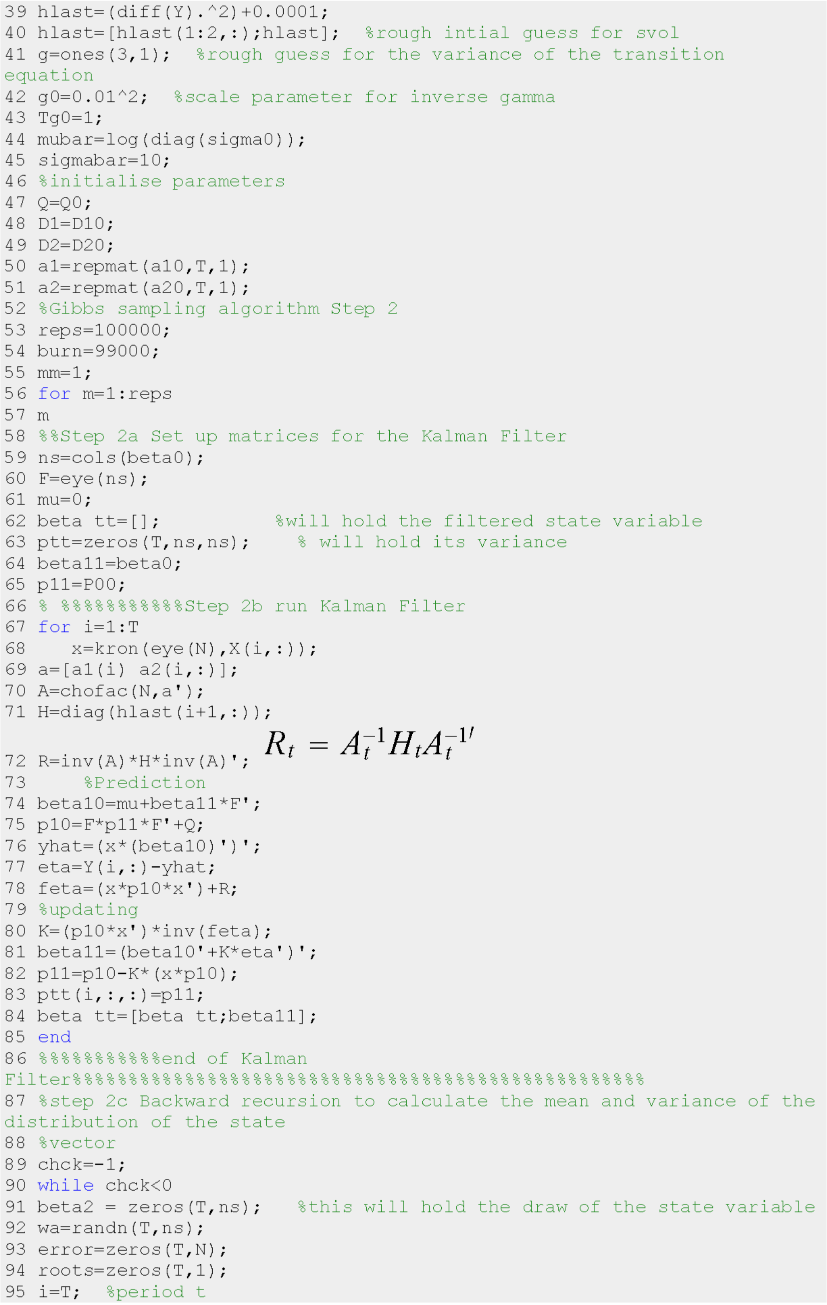
$$D_1 = 0.001 \text{ and } D_2 = \begin{pmatrix} 0.001 & 0 \\ 0 & 0.001 \end{pmatrix}$$

```
32 D10=10^(-3);  %prior scale matrix for D1
33 D20=10^(-3)*eye(2); %prior scale matrix for D2
34 %remove intial Sample
35 Y=Y(T0+1:end,:);
36 X=X(T0+1:end,:);
37 T=rows(X);
38 %priors and starting values for the stochastic vol
```

Obtain a starting value for $h_{i,t}, t = 0....T$ and $i = 1..3$ as $\hat{v}_{it}^2$

FIGURE 29. Matlab code for the time-varying VAR with stochastic volatility

The matlab code for estimating this model (example6.m) is shown in figures 29, 30, 31 and 32. We consider a time-varying VAR model with two lags using US data on GDP growth, CPI inflation and the Federal Funds rate over the period 1954Q3 to 2010Q2 in this code. Lines 25 to 27 set the initial values for the elements of A_t by calculating the matrix $C0$. Lines 30 amd 31 set the variance around these initial values. Lines 32 and 33 set the prior scale matrices D_{10} and D_{20}. Lines 38 to 45 set the priors and starting values for the stochastic volatility models for the transformed VAR residuals ε_t. Lines 59 to 112 contain the Carter and Kohn algorithm to sample the VAR coefficients β_t. The only change relative to the example in chapter 3 is on lines 69 to 72. Line 70 using the function chofac.m to reshape the value of $a_{ij,t}$ at time t into a lower triangular matrix. Line 72 calculates the VAR error covariance



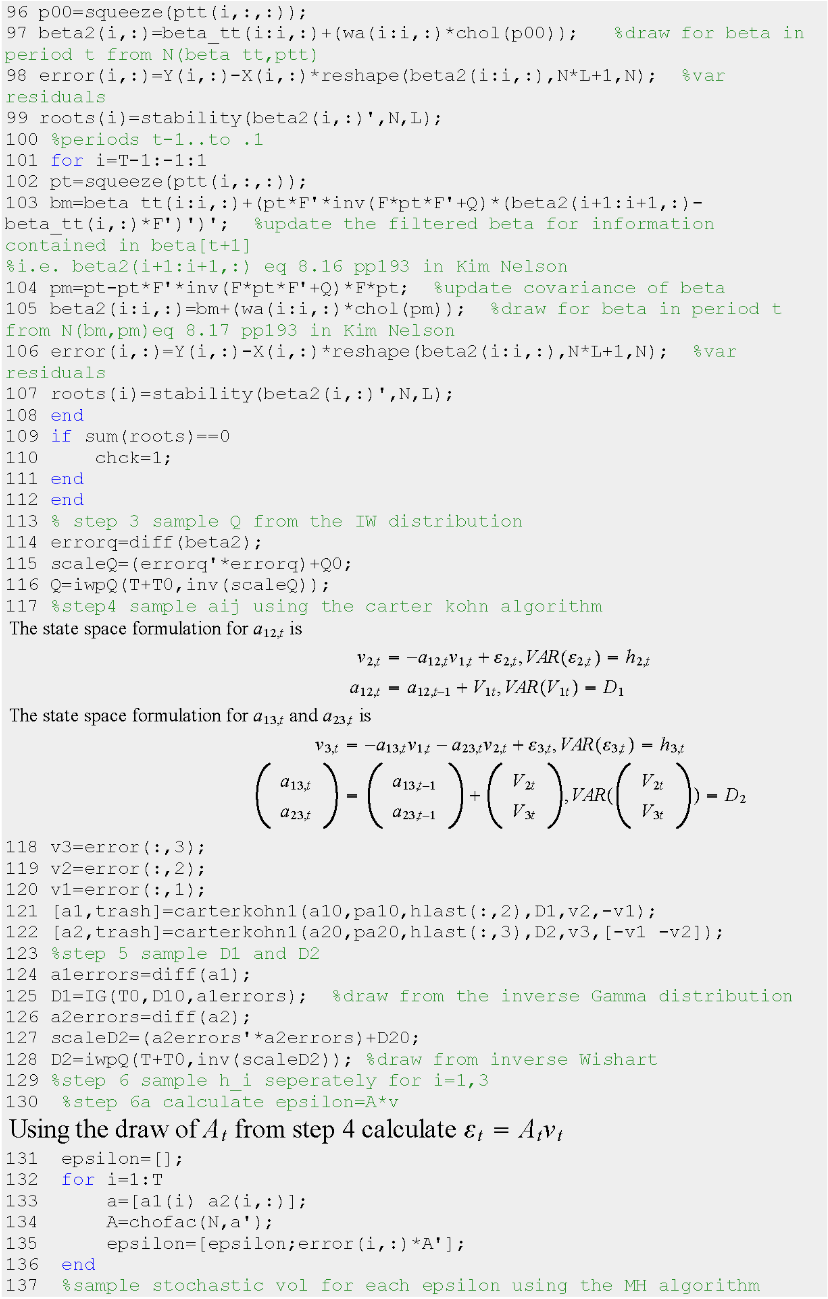
```
39 hlast=(diff(Y).^2)+0.0001;
40 hlast=[hlast(1:2,:);hlast];  %rough intial guess for svol
41 g=ones(3,1);  %rough guess for the variance of the transition
equation
42 g0=0.01^2;  %scale parameter for inverse gamma
43 Tg0=1;
44 mubar=log(diag(sigma0));
45 sigmabar=10;
46 %initialise parameters
47 Q=Q0;
48 D1=D10;
49 D2=D20;
50 a1=repmat(a10,T,1);
51 a2=repmat(a20,T,1);
52 %Gibbs sampling algorithm Step 2
53 reps=100000;
54 burn=99000;
55 mm=1;
56 for m=1:reps
57 m
58 %%Step 2a Set up matrices for the Kalman Filter
59 ns=cols(beta0);
60 F=eye(ns);
61 mu=0;
62 beta tt=[];           %will hold the filtered state variable
63 ptt=zeros(T,ns,ns);    % will hold its variance
64 beta11=beta0;
65 p11=P00;
66 % %%%%%%%%%%%Step 2b run Kalman Filter
67 for i=1:T
68    x=kron(eye(N),X(i,:));
69 a=[a1(i) a2(i,:)];
70 A=chofac(N,a');
71 H=diag(hlast(i+1,:));
```

$$R_t = A_t^{-1} H_t A_t^{-1'}$$

```
72 R=inv(A)*H*inv(A)';
73     %Prediction
74 beta10=mu+beta11*F';
75 p10=F*p11*F'+Q;
76 yhat=(x*(beta10)')';
77 eta=Y(i,:)-yhat;
78 feta=(x*p10*x')+R;
79 %updating
80 K=(p10*x')*inv(feta);
81 beta11=(beta10'+K*eta')';
82 p11=p10-K*(x*p10);
83 ptt(i,:,:)=p11;
84 beta tt=[beta tt;beta11];
85 end
86 %%%%%%%%%%end of Kalman
Filter%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
87 %step 2c Backward recursion to calculate the mean and variance of the
distribution of the state
88 %vector
89 chck=-1;
90 while chck<0
91 beta2 = zeros(T,ns);   %this will hold the draw of the state variable
92 wa=randn(T,ns);
93 error=zeros(T,N);
94 roots=zeros(T,1);
95 i=T;  %period t
```

FIGURE 30. Matlab code for the time-varying VAR with stochastic volatility

matrix for that time period and this is used in Kalman filter equations. Line 116 samples Q from the inverse Wishart distribution. Line 121 uses the Carter and Kohn algorithm to sample $a_{12,t}$ (where for simplicity the code for the algorithm is in the function carterkohn1.m). Line 122 samples $a_{13,t}$ and $a_{23,t}$ using the same function. Lines 124 and 125 sample D_1 from the inverse Gamma distribution. Lines 127 and 128 sample D_2 from the inverse Wishart distribution. Lines 131 to 136 calculate $\varepsilon_t = A_t v_t$. Lines 138 to 142 use the independence MH algorithm to draw $h_{i,t}, i = 1..3$ using these ε_t. The code for the algorithm is identical to the two previous examples but is included in the function getsvol.m for simplicity. This function takes in the following inputs (1) the previous draw of $h_{i,t}$ (2) g_i (3) $\bar{\mu}$ (4) $\bar{\sigma}$ (5) ε_t and returns a draw for $h_{i,t}$. Lines 145 to 148 draw g_i from the inverse Gamma distribution.



```
p00=squeeze(ptt(i,:,:));
beta2(i,:)=beta_tt(i:i,:)+(wa(i:i,:)*chol(p00));   %draw for beta in period t from N(beta tt,ptt)
error(i,:)=Y(i,:)-X(i,:)*reshape(beta2(i:i,:),N*L+1,N);  %var residuals
roots(i)=stability(beta2(i,:)',N,L);
%periods t-1..to .1
for i=T-1:-1:1
pt=squeeze(ptt(i,:,:));
bm=beta tt(i:i,:)+(pt*F'*inv(F*pt*F'+Q)*(beta2(i+1:i+1,:)-beta_tt(i,:)*F')')';  %update the filtered beta for information contained in beta[t+1]
%i.e. beta2(i+1:i+1,:) eq 8.16 pp193 in Kim Nelson
pm=pt-pt*F'*inv(F*pt*F'+Q)*F*pt;  %update covariance of beta
beta2(i:i,:)=bm+(wa(i:i,:)*chol(pm));  %draw for beta in period t from N(bm,pm)eq 8.17 pp193 in Kim Nelson
error(i,:)=Y(i,:)-X(i,:)*reshape(beta2(i:i,:),N*L+1,N);  %var residuals
roots(i)=stability(beta2(i,:)',N,L);
end
if sum(roots)==0
    chck=1;
end
end
% step 3 sample Q from the IW distribution
errorq=diff(beta2);
scaleQ=(errorq'*errorq)+Q0;
Q=iwpQ(T+T0,inv(scaleQ));
%step4 sample aij using the carter kohn algorithm
```

The state space formulation for $a_{12,t}$ is

$$v_{2,t} = -a_{12,t}v_{1,t} + \varepsilon_{2,t}, VAR(\varepsilon_{2,t}) = h_{2,t}$$

$$a_{12,t} = a_{12,t-1} + V_{1t}, VAR(V_{1t}) = D_1$$

The state space formulation for $a_{13,t}$ and $a_{23,t}$ is

$$v_{3,t} = -a_{13,t}v_{1,t} - a_{23,t}v_{2,t} + \varepsilon_{3,t}, VAR(\varepsilon_{3,t}) = h_{3,t}$$

$$\begin{pmatrix} a_{13,t} \\ a_{23,t} \end{pmatrix} = \begin{pmatrix} a_{13,t-1} \\ a_{23,t-1} \end{pmatrix} + \begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}, VAR(\begin{pmatrix} V_{2t} \\ V_{3t} \end{pmatrix}) = D_2$$

```
v3=error(:,3);
v2=error(:,2);
v1=error(:,1);
[a1,trash]=carterkohn1(a10,pa10,hlast(:,2),D1,v2,-v1);
[a2,trash]=carterkohn1(a20,pa20,hlast(:,3),D2,v3,[-v1 -v2]);
%step 5 sample D1 and D2
a1errors=diff(a1);
D1=IG(T0,D10,a1errors);  %draw from the inverse Gamma distribution
a2errors=diff(a2);
scaleD2=(a2errors'*a2errors)+D20;
D2=iwpQ(T+T0,inv(scaleD2)); %draw from inverse Wishart
%step 6 sample h_i seperately for i=1,3
 %step 6a calculate epsilon=A*v
```

Using the draw of A_t from step 4 calculate $\varepsilon_t = A_t v_t$

```
 epsilon=[];
 for i=1:T
     a=[a1(i) a2(i,:)];
     A=chofac(N,a');
     epsilon=[epsilon;error(i,:)*A'];
 end
 %sample stochastic vol for each epsilon using the MH algorithm
```

FIGURE 31. Matlab code for the time-varying VAR with stochastic volatility

Figure 33 plots the estimated impulse response to a monetary policy shock (identified via sign restrictions) and the estimated stochastic volatility.

6. Convergence of the MH algorithm

```
  hnew=[];
  for i=1:N
      htemp=getsvol(hlast(:,i),g(i),mubar(i),sigmabar,epsilon(:,i));
      hnew=[hnew htemp];
  end
hlast=hnew;
%step 7 Sample G for IG distribution
for i=1:N
    gerrors=diff(log(hnew(:,i)));
g(i)=IG(Tg0,g0,gerrors);  %draw from the inverse Gamma distribution
end
if m>burn
    %save output from Gibbs sampler
    out1(mm,1:T,:)=beta2;
    out2(mm,1:T,1:N)=hlast(2:end,:);
    out3(mm,1:N*(N*L+1),1:N*(N*L+1))=Q;
    out4(mm,1:T,1:(N*(N-1))/2)=[a1 a2];
    out5(mm,1)=D1;
    out6(mm,1:2,1:2)=D2;
    out7(mm,1:N)=g';
    mm=mm+1;
end
end
%save results
save tvp.mat out1 out2 out3 out4 out5 out6 out7
%compute irf to a policy shock using sign restrictions
horz=40;% impulse response horizon
irfmat=zeros(size(out1,1),T,horz,N); %empty matrix to save impulse response to a policy shock
for i=1:size(out1,1);

    for j=1:size(out1,2)

       H=diag(squeeze(out2(i,j,:)));
       a=squeeze(out4(i,j,:));
        A=chofac(N,a);
        sigma=inv(A)*H*inv(A)';  %covariance matrix
   %sign restrictions
        chck=-1;
        while chck<0
        K=randn(N,N);
        QQ=getQR(K);
        A0hat=chol(sigma);
        A0hat1=(QQ*A0hat);  %candidate draw
        for m=1:N
        %check signs in each row
        e1=A0hat1(m,1)<0;  %Response of Y
        e2=A0hat1(m,2)<0;  %Response of P
        e3=A0hat1(m,3)>0;  %Response of R

        if e1+e2+e3==3
            MP=A0hat1(m,:);
            chck=10;
        else
            %check signs but reverse them
         e1=-A0hat1(m,1)<0;  %Response of Y
        e2=-A0hat1(m,2)<0;  %Response of P
        e3=-A0hat1(m,3)>0;  %Response of R

        if e1+e2+e3==3
            MP=-A0hat1(m,:);
```

Figure 32. Matlab code for the time-varying VAR with stochastic volatility

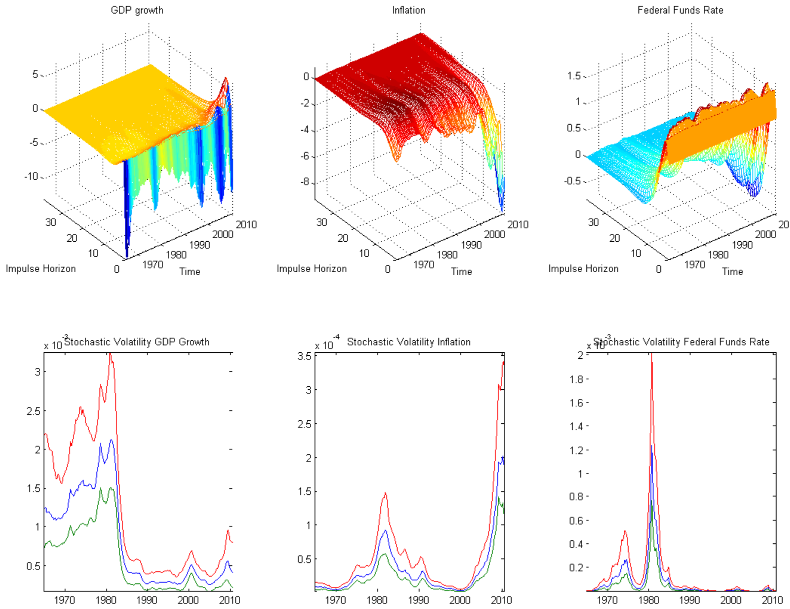


FIGURE 33. Response to a monetary policy shock from the time-varying VAR with stochastic volatility (Top panel) and the estimated stochastic volatility (bottom panel)

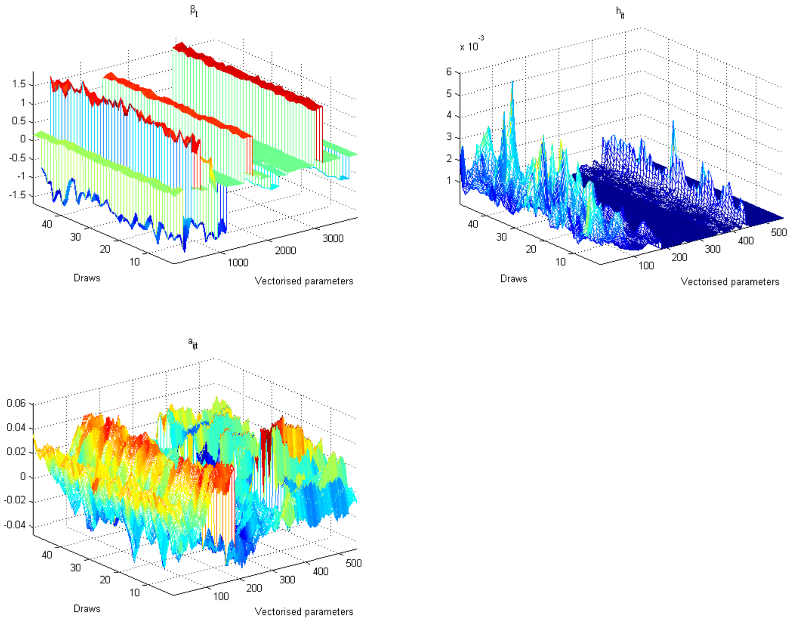


FIGURE 34. Recursive mean for key parameters of the time-varying VAR model

Most of the methods for checking convergence of the Gibbs sampler (see Chapter 1) can be applied immediately to output from the MH algorithm. Several studies present simple statistics such as recursive means of the MH draws and the autocorrelation functions to test if the algorithm has converged. As an example we present the recursive means of the retained draws for the time-varying parameter VAR considered in the previous section. As described above this model is estimated using a mixture of Gibbs and MH steps. Figure 34 presents the recursive means calculated every 20 draws for β_t, h_{it} and $a_{ij,t}$. The X-axis of each panel represents these parameterised vectorised. The Y-axis represents the draws. The recursive means usggest convergence for β_t, $a_{ij,t}$ but indicate some variation in the means for h_{it} possibly suggesting that more draws are required for this model.

Gelman and Rubin (1992) suggest a diagnostic for monitoring the convergence of multiple MH chains (for estimating the same model) started from different starting values. For every parameter of interest ξ Gelman and Rubin (1992) calculate the *within chain variance* as

$$\begin{aligned} W_T &= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{T}\sum_{t=1}^{T}\left(\bar{\xi}_{t,m}-\bar{\xi}_m\right)^2, \\ \bar{\xi}_m &= \frac{1}{T}\sum_{t=1}^{T}\xi_{tm}, \bar{\xi} = \frac{1}{M}\sum_{m=1}^{M}\bar{\xi}_m \end{aligned}$$

where T denotes the total number of iterations in each of the M MH algorithms.

Gelman and Rubin (1992) calculate the between chain variance

$$B_T = \frac{1}{M}\sum_{m=1}^{M}\left(\bar{\xi}_m-\bar{\xi}\right)^2$$

They argue that W_T underestimates the variance of ξ (before convergence) as the MH algorithm has not explored the parameter space. In contrast, $\sigma_T^2 = \frac{T-1}{T}W_T + B_T$ overestimates this variance due to dispersed starting values. If the MH algorithm has converged then W_T and σ_T^2 should be similar. Gelman and Rubin (1992) suggest calculating the statistic

$$R_T = \frac{\sigma_T^2+\frac{B_T}{M}}{W_T}\frac{v_T}{v_T-2}$$

where $v_T = \frac{2\left(\sigma_T^2+\frac{B_T}{M}\right)^2}{W_T}$ and checking if this is close to 1 which would indicate convergence of the MH algorithm.

7. Further Reading

- Koop (2003) chapter 5 provides an excellent description of the Metropolis Hastings algorithm.

8. Appendix: Computing the marginal likelihood using the Gelfand and Dey method

Gelfand and Dey (1994) introduce a method for computing the marginal likelihood that is particularly convenient to use when employing the Metropolis Hastings algorithm. This method is based on the following result.

$$E\left[\frac{f(\Phi)}{F(Y|\Phi)\times P(\Phi)}|Y\right]=\frac{1}{F(Y)} \tag{8.1}$$

where $F(Y|\Phi)$ denotes the likelihood function, $P(\Phi)$ is the prior distribution, $F(Y)$ is the marginal likelihood and $f(\Phi)$ is any pdf with support Θ defined within the region of the posterior. The proof of equation 8.1 can be obtained by noting that $E\left[\frac{f(\Phi)}{F(Y|\Phi)\times P(\Phi)}|Y\right]=\int\frac{f(\Phi)}{F(Y|\Phi)\times P(\Phi)}\times H(\Phi|Y)\,d\Phi$ where $H(\Phi|Y)$ is the posterior distribution. Note that $H(\Phi|Y)=\frac{F(Y|\Phi)\times P(\Phi)}{F(Y)}$ and the density $f(\Phi)$ integrates to 1 leaving us with the right hand side in equation 8.1.

We can approximate the marginal likelihood as $\frac{1}{M}\sum_{j=1}^{M}\frac{f(\Phi_j)}{F(Y|\Phi_j)\times P(\Phi_j)}$ where Φ_j denotes draws of the parameters from Metropolis Hastings algorithm and $F(Y|\Phi_j)\times P(\Phi_j)$ is the posterior evaluated at each draw. Geweke (1998) recommends using a truncated normal distribution for $f(\Phi)$. This distribution is truncated at the tails to ensure that $f(\Phi)$ is bounded from above, a requirement in Gelfand and Dey (1994). In particular, Geweke (1998) suggest using

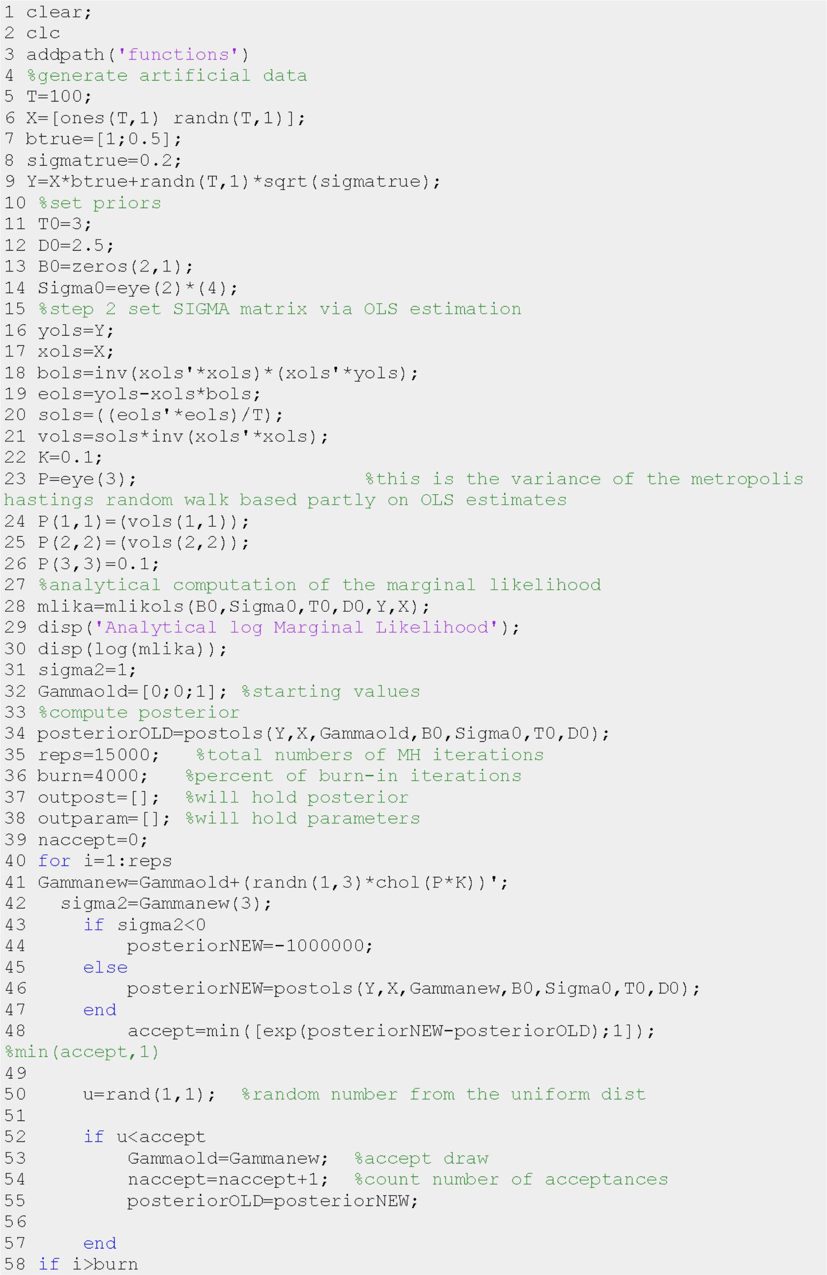
$$f(\Phi)=\frac{1}{p(2\pi)^{k/2}}\left|\hat{\Sigma}\right|^{-1/2}\exp\left[-0.5\left(\Phi_j-\hat{\Phi}\right)\hat{\Sigma}^{-1}\left(\Phi_j-\hat{\Phi}\right)'\right]\times I\left(\Phi_j\in\hat{\Theta}\right) \tag{8.2}$$

where $\hat{\Phi}$ is the posterior mean, $\hat{\Sigma}$ is the posterior covariance and k is the number of parameters. The indicator function $I\left(\Phi_j\in\hat{\Theta}\right)$ takes a value of 1 if

$$\left[\left(\Phi-\hat{\Phi}\right)\hat{\Sigma}^{-1}\left(\Phi-\hat{\Phi}\right)'\right]\leq\chi^2_{1-p}(k)$$

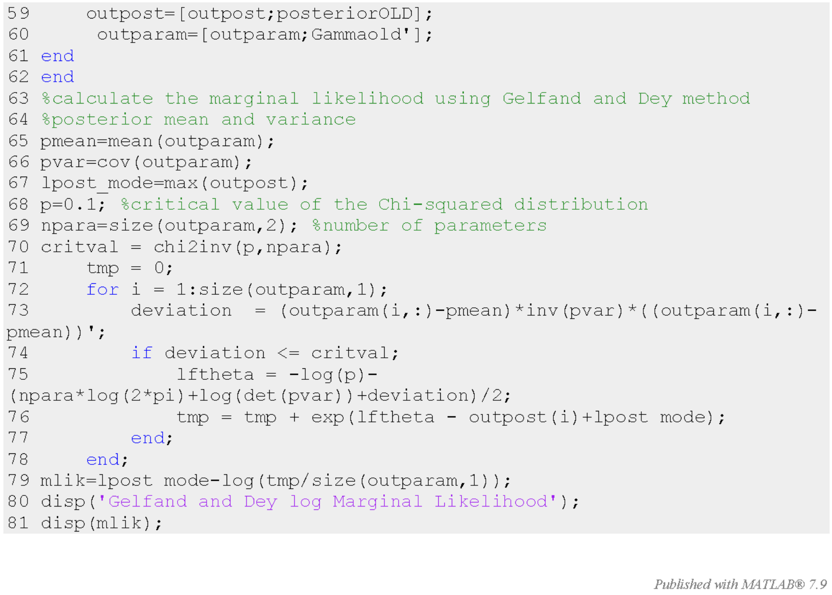
where $\chi^2_{1-p}(k)$ is the inverse χ^2 cumulative distribution function with degrees of freedom k and probability p. Thus $\chi^2_{1-p}(k)$ denotes the value that exceeds $1-p\%$ of the samples from a χ^2 distribution with k degrees of freedom. The indicator function $I\left(\Phi_j\in\hat{\Theta}\right)$ therefore removes 'extreme' values of Φ_j. For more details, see Koop (2003) page 104.

In figures 35 and 36 we estimate the marginal likelihood for a linear regression model via the Gelfand and Dey method. The model is exactly used in the appendix to Chapter 1 and is based on artifical data. A simple random walk Metropolis Hastings algorithm is used to approximate the posterior on lines 35 to 62 and we save the log posterior evaluated at each draw and each draw of the parameters. Lines 65 and 66 calculate the posterior mean and variance. We set $1-p=0.1$ on line 68. In practice, different value of $1-p$ can be tried to check robustness of the estimate. On line 70 we evaluate the inverse χ^2 CDF. Line 71 to 78, loop through the saved draws of the parameters. On 73 we calculate $\left(\Phi-\hat{\Phi}\right)\hat{\Sigma}^{-1}\left(\Phi-\hat{\Phi}\right)'$. If this is less than or equal to $\chi^2_{1-p}(k)$ we evaluate $\frac{f(\Phi_j)}{F(Y|\Phi_j)\times P(\Phi_j)}$ in logs, adding the constant lpost_mode to prevent overflow.



```
clear;
clc
addpath('functions')
%generate artificial data
T=100;
X=[ones(T,1) randn(T,1)];
btrue=[1;0.5];
sigmatrue=0.2;
Y=X*btrue+randn(T,1)*sqrt(sigmatrue);
%set priors
T0=3;
D0=2.5;
B0=zeros(2,1);
Sigma0=eye(2)*(4);
%step 2 set SIGMA matrix via OLS estimation
yols=Y;
xols=X;
bols=inv(xols'*xols)*(xols'*yols);
eols=yols-xols*bols;
sols=((eols'*eols)/T);
vols=sols*inv(xols'*xols);
K=0.1;
P=eye(3);                    %this is the variance of the metropolis hastings random walk based partly on OLS estimates
P(1,1)=(vols(1,1));
P(2,2)=(vols(2,2));
P(3,3)=0.1;
%analytical computation of the marginal likelihood
mlika=mlikols(B0,Sigma0,T0,D0,Y,X);
disp('Analytical log Marginal Likelihood');
disp(log(mlika));
sigma2=1;
Gammaold=[0;0;1]; %starting values
%compute posterior
posteriorOLD=postols(Y,X,Gammaold,B0,Sigma0,T0,D0);
reps=15000;   %total numbers of MH iterations
burn=4000;   %percent of burn-in iterations
outpost=[];  %will hold posterior
outparam=[]; %will hold parameters
naccept=0;
for i=1:reps
Gammanew=Gammaold+(randn(1,3)*chol(P*K))';
  sigma2=Gammanew(3);
    if sigma2<0
        posteriorNEW=-1000000;
    else
        posteriorNEW=postols(Y,X,Gammanew,B0,Sigma0,T0,D0);
    end
        accept=min([exp(posteriorNEW-posteriorOLD);1]); %min(accept,1)

    u=rand(1,1);  %random number from the uniform dist

    if u<accept
        Gammaold=Gammanew;  %accept draw
        naccept=naccept+1;  %count number of acceptances
        posteriorOLD=posteriorNEW;

    end
if i>burn
```

FIGURE 35. Matlab code to calculate the marginal likelihood via the Gelfand and Dey Method



```
    outpost=[outpost;posteriorOLD];
     outparam=[outparam;Gammaold'];
end
end
%calculate the marginal likelihood using Gelfand and Dey method
%posterior mean and variance
pmean=mean(outparam);
pvar=cov(outparam);
lpost_mode=max(outpost);
p=0.1; %critical value of the Chi-squared distribution
npara=size(outparam,2); %number of parameters
critval = chi2inv(p,npara);
    tmp = 0;
    for i = 1:size(outparam,1);
        deviation  = (outparam(i,:)-pmean)*inv(pvar)*((outparam(i,:)-pmean))';
        if deviation <= critval;
            lftheta = -log(p)-(npara*log(2*pi)+log(det(pvar))+deviation)/2;
            tmp = tmp + exp(lftheta - outpost(i)+lpost mode);
        end;
    end;
mlik=lpost mode-log(tmp/size(outparam,1));
disp('Gelfand and Dey log Marginal Likelihood');
disp(mlik);
```

Published with MATLAB® 7.9

Figure 36. Matlab code to calculate the marginal likelihood via the Gelfand and Dey Method (continued)

CHAPTER 6

Bayesian estimation of Linear DSGE models

This chapter considers the Bayesian estimation of Dynamic Stochastic General Equilibrium (DSGE) models using the random walk metropolis hastings algorithm. These models are popular in academia and central banks for policy analysis. Several menu driven computer packages (e.g. DYNARE) are now available for the estimation of these models and several papers and books discuss the econometrics of DSGE models (see An and Schorfheide (2007)). The focus of the chapter is practical – it offers a step by step guide to DSGE estimation from a Bayesian perspective and tries to clarify the practical aspects of the problem. It, therefore, offers a useful starting point for researchers who are new to DSGE *estimation* but are familiar with the economics behind these models.

1. The DSGE model

In this chapter we consider the estimation of the following simple log-linearised DSGE model:

$$\begin{aligned}
x_t &= E_t x_{t+1} - (1/\sigma)(i_t - E_t \pi_{t+1}) + g_t \\
\pi_t &= \beta E_t \pi_{t+1} + \kappa x_t + u_t \\
i_t &= \delta \pi_t + v_t \\
\kappa &= \frac{(1-\varpi)(1-\beta\varpi)}{\alpha\varpi} \\
g_t &= \rho_1 g_{t-1} + \varepsilon_{1t}, \varepsilon_{1t}\tilde{\ } N(0,\sigma_1^2) \\
u_t &= \rho_2 u_{t-1} + \varepsilon_{2t}, \varepsilon_{2t}\tilde{\ } N(0,\sigma_2^2) \\
v_t &= \rho_3 v_{t-1} + \varepsilon_{3t}, \varepsilon_{3t}\tilde{\ } N(0,\sigma_3^2)
\end{aligned} \tag{1.1}$$

Here x_t is output gap, π_t is inflation and i_t is the short-term interest rate. The first equation is the IS curve linking the output gap to the real interest rate and a 'demand shock' g_t. The second equation is the Phillips curve linking inflation to inflation expectations and the output gap while u_t is a supply shock. The third equation is a simple policy rule that postulates that interest rates are set in response to inflation developments with the policy shock denoted by v_t. The three shocks follow AR(1) processes as shown by the last three equations. Our aim is to estimate the unknown parameters $\Theta = (\sigma, \delta, \alpha, \varpi, \rho_1, \rho_2, \rho_3, \sigma_1^2, \sigma_2^2, \sigma_3^2)$. As is typical in the literature, we fix $\beta = 0.99$.

1.1. Solving the model. The model in equation 1.1 cannot be estimated in its current form as it includes unobserved variables dated in the future on the right hand side of the first two equations. One way to proceed is to solve the model so that it can be re-written in a VAR form:

$$\beta_t = F\beta_{t-1} + gz_t \tag{1.2}$$

where $\beta_t = \begin{pmatrix} x_t \\ \pi_t \\ i_t \\ g_t \\ u_t \\ v_t \end{pmatrix}$ and z_t are iid shocks with covariance matrix $\begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}$. The elements of the coefficient matrix F and the contemporaneous impact matrix g are functions of the model parameters Θ. If we treat β_t as state variables, then equation 1.2 is a transition equation. As described below, once this is combined with an observation equation we have a linear state space model that can be easily estimated as the Kalman filter can be used to calculate the likelihood of the model.

Therefore, model solution is a key step in the estimation process. There are several solution algorithms available. In this application we use the algorithm developed in Sims (2002). To use this algorithm and Chris Sims' code, the model needs to be written in matrix form:

$$\gamma_0 \tilde{\beta}_t = \gamma_1 \tilde{\beta}_{t-1} + \xi \varepsilon_t + \bar{\kappa} v_t$$

where v_t denotes expectational errors $x_t - E_{t-1}x_t$ and $\pi_t - E_{t-1}\pi_t$. Here $\tilde{\beta}_t$ is:

$$\tilde{\beta}_t = \begin{pmatrix} x_t \\ \pi_t \\ i_t \\ g_t \\ u_t \\ v_t \\ E_t x_{t+1} \\ E_t \pi_{t+1} \end{pmatrix}$$

For the model in equation 1.1 these matrices are defined as:

$$\gamma_0 = \begin{array}{cc} \begin{pmatrix} 1 & 0 & 1/\sigma & -1 & 0 & 0 & -1 & -1/\sigma \\ -\kappa & 1 & 0 & 0 & -1 & 0 & 0 & -\beta \\ 0 & -\delta & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{IS curve} \\ \text{Phillips curve} \\ \text{Policy rule} \\ \text{Demand shock} \\ \text{Supply shock} \\ \text{Policy shock} \\ \text{Exp. error 1} \\ \text{Exp. error 2} \end{array} \end{array} \tag{1.3}$$

$$\gamma_1 = \begin{array}{cc} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \rho_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \rho_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \rho_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} & \begin{array}{c} \text{IS curve} \\ \text{Phillips curve} \\ \text{Policy rule} \\ \text{Demand shock} \\ \text{Supply shock} \\ \text{Policy shock} \\ \text{Exp. error 1} \\ \text{Exp. error 2} \end{array} \end{array} \tag{1.4}$$

$$\xi = \begin{array}{cc} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{array}{c} \text{IS curve} \\ \text{Phillips curve} \\ \text{Policy rule} \\ \text{Demand shock} \\ \text{Supply shock} \\ \text{Policy shock} \\ \text{Exp. error 1} \\ \text{Exp. error 2} \end{array} \end{array} \tag{1.5}$$

$$\bar{\kappa} = \begin{array}{cc} \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} & \begin{array}{c} \text{IS curve} \\ \text{Phillips curve} \\ \text{Policy rule} \\ \text{Demand shock} \\ \text{Supply shock} \\ \text{Policy shock} \\ \text{Exp. error 1} \\ \text{Exp. error 2} \end{array} \end{array} \tag{1.6}$$

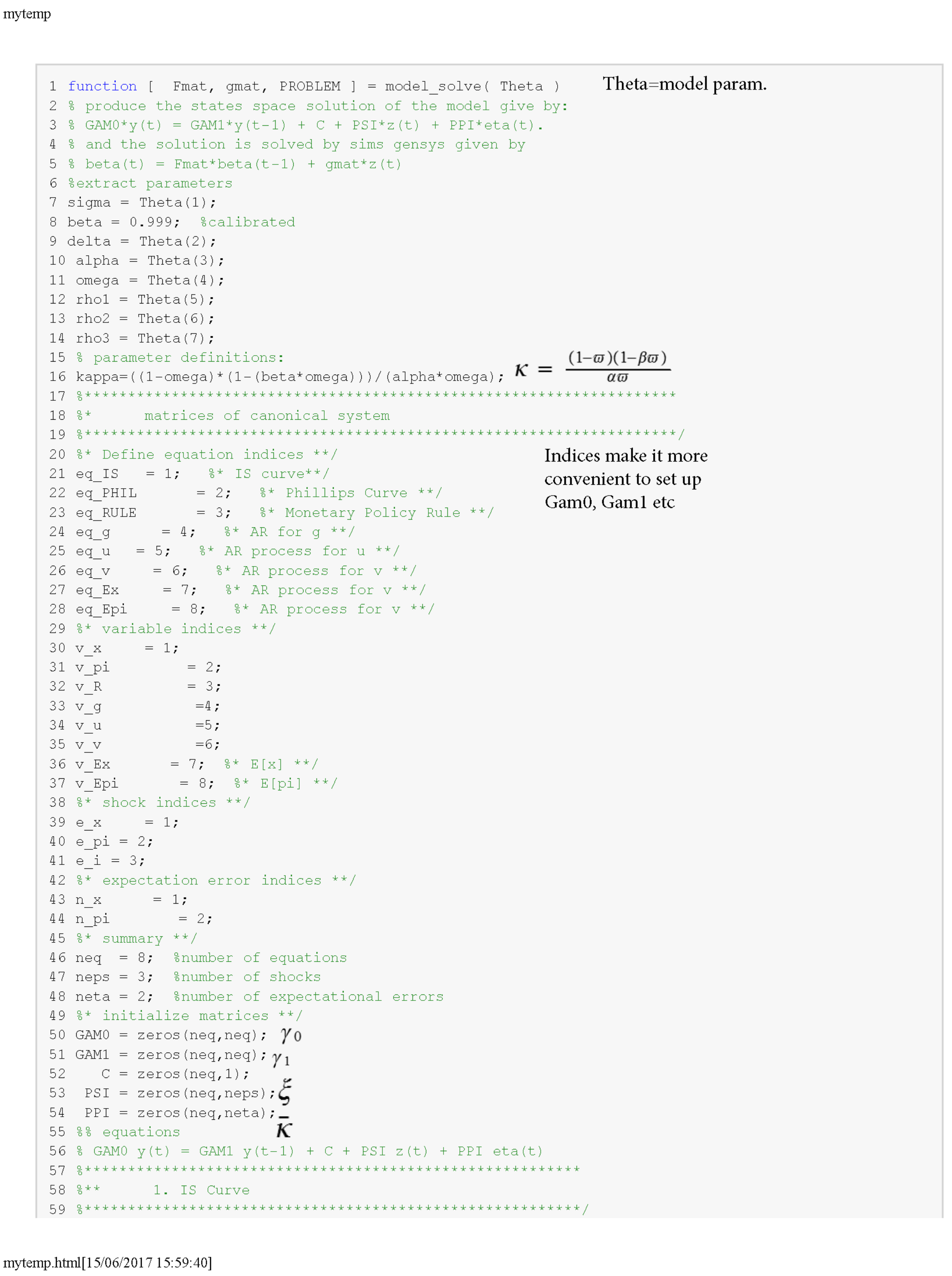
Equation 1.3 shows γ_0 which is an 8×8 matrix in our case. The dimensions reflect the number of variables in the model including the 2 expectational errors. The dimensions of γ_1 are also 8×8 while the number of columns of ξ and κ reflect the fact that the model has three iid structural shocks and two expectational errors.

The file example1.m demonstrates the solution of the model using the Sims (2002) method. For this example, we use assume that the parameters have the following values:

$$\sigma = 1, \delta = 1.5, \alpha = 3, \varpi = 1.5, \rho_1 = 0.7, \rho_2 = 0.7, \rho_3 = 0.7$$

Line 16 calls the main function model_solve.m that sets up $\gamma_0, \gamma_1, \xi, \bar{\kappa}$ and calls the function to solve the model.

Figures 1 and 2 display the code for this function. The input to the function is the vector of parameters which are extracted on lines 7 to 16. In order to set up $\gamma_0, \gamma_1, \xi, \bar{\kappa}$ while minimising coding errors, it is helpful to set up indices of equations, variables and shocks. These are set up on lines 21 to 44. Line 60 to 65 modifies the first row of γ_0 to insert the coefficients of the IS equation. Lines 68 to 71 insert the coefficients of the Phillips curve in the second row while lines 75 to 77 deal with the policy rule. Lines 83 to 85 modify γ_0, γ_1 and ξ to reflect the coefficients of the demand shock equation $g_t = \rho_1 g_{t-1} + \varepsilon_{1t}$. Lines 87 to 93 do exactly the same for the supply and policy shocks. Finally, the expectational errors are dealt with on lines 97 to 104. The function written by Sims to solve the model is called *gensys*. The inputs to this function are γ_0, γ_1, a vector C that specifies constants in the model (as shown on line 52, this is a vector of zeros as all variables are in deviations from the steady state in the example model), $\xi, \bar{\kappa}$ and a number *div* which specifies the criteria for stable roots. Typically div=1. In other words, the function call is



mytemp

```
1 function [  Fmat, gmat, PROBLEM ] = model_solve( Theta )
2 % produce the states space solution of the model give by:
3 % GAM0*y(t) = GAM1*y(t-1) + C + PSI*z(t) + PPI*eta(t).
4 % and the solution is solved by sims gensys given by
5 % beta(t) = Fmat*beta(t-1) + gmat*z(t)
6 %extract parameters
7 sigma = Theta(1);
8 beta = 0.999;  %calibrated
9 delta = Theta(2);
10 alpha = Theta(3);
11 omega = Theta(4);
12 rho1 = Theta(5);
13 rho2 = Theta(6);
14 rho3 = Theta(7);
15 % parameter definitions:
16 kappa=((1-omega)*(1-(beta*omega)))/(alpha*omega);
17 %*********************************************************************
18 %*      matrices of canonical system
19 %*********************************************************************/
20 %* Define equation indices **/
21 eq_IS   = 1;   %* IS curve**/
22 eq_PHIL       = 2;   %* Phillips Curve **/
23 eq_RULE       = 3;   %* Monetary Policy Rule **/
24 eq_g      = 4;   %* AR for g **/
25 eq_u   = 5;   %* AR process for u **/
26 eq_v     = 6;   %* AR process for v **/
27 eq_Ex     = 7;   %* AR process for v **/
28 eq_Epi     = 8;   %* AR process for v **/
29 %* variable indices **/
30 v_x     = 1;
31 v_pi         = 2;
32 v_R          = 3;
33 v_g           =4;
34 v_u           =5;
35 v_v           =6;
36 v_Ex       = 7;  %* E[x] **/
37 v_Epi       = 8;  %* E[pi] **/
38 %* shock indices **/
39 e_x     = 1;
40 e_pi = 2;
41 e_i = 3;
42 %* expectation error indices **/
43 n_x      = 1;
44 n_pi        = 2;
45 %* summary **/
46 neq  = 8;  %number of equations
47 neps = 3;  %number of shocks
48 neta = 2;  %number of expectational errors
49 %* initialize matrices **/
50 GAM0 = zeros(neq,neq);
51 GAM1 = zeros(neq,neq);
52    C = zeros(neq,1);
53   PSI = zeros(neq,neps);
54   PPI = zeros(neq,neta);
55 %% equations
56 % GAM0 y(t) = GAM1 y(t-1) + C + PSI z(t) + PPI eta(t)
57 %*********************************************************
58 %**      1. IS Curve
59 %*********************************************************/
```

Theta=model param.

$\kappa = \frac{(1-\varpi)(1-\beta\varpi)}{\alpha\varpi}$

Indices make it more convenient to set up Gam0, Gam1 etc

γ_0

γ_1

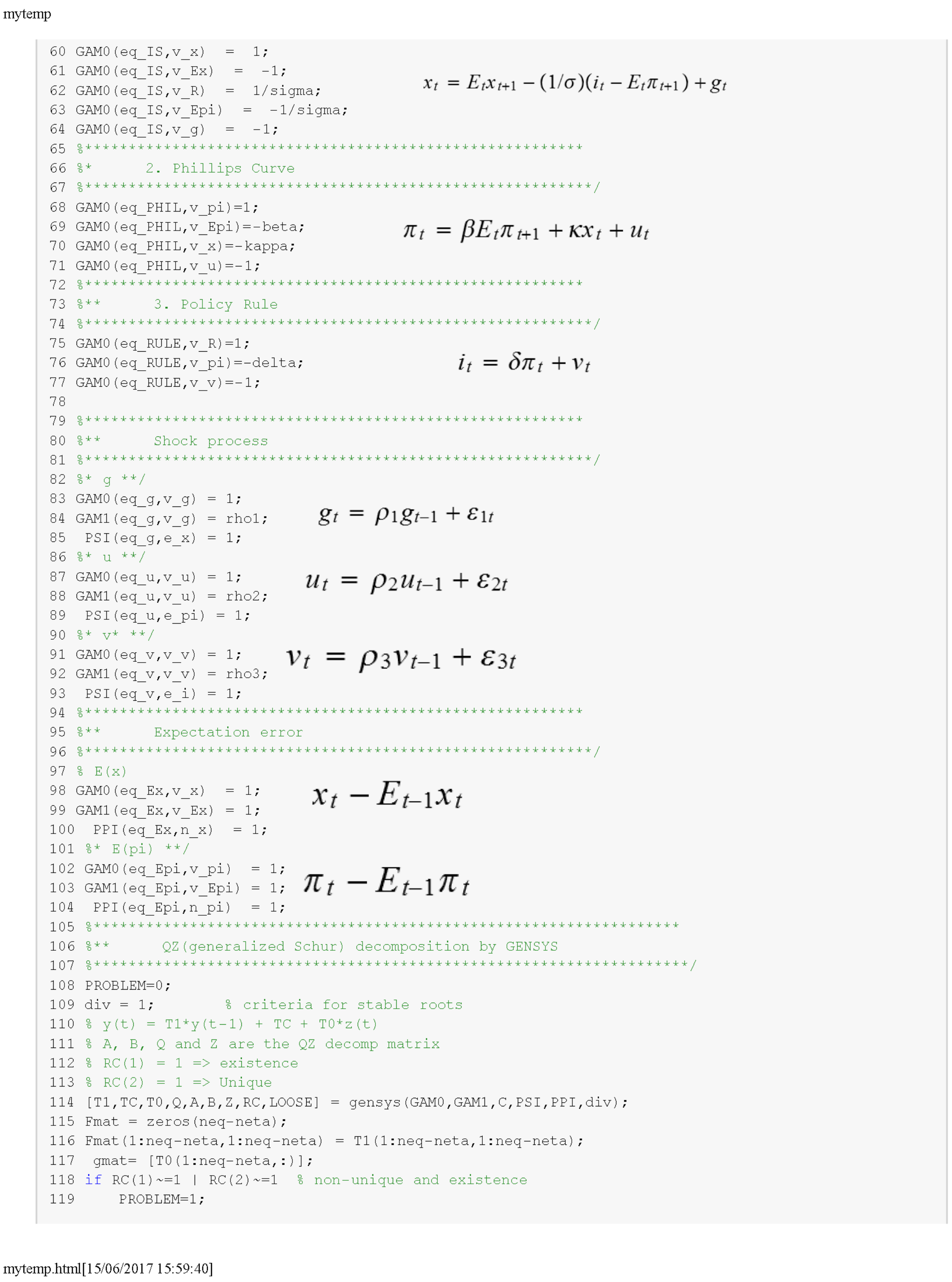
ξ

$\bar{\kappa}$

mytemp.html[15/06/2017 15:59:40]

FIGURE 1. Model Solution

gensys($\underset{\gamma_0}{GAM0}$,$\underset{\gamma_1}{GAM1}$,C,$\underset{\xi}{PSI}$,$\underset{\bar{\kappa}}{PPI}$,div). The function is called on line 114. The first key output from this function is RC which is a 2×1 vector. If the first element of this vector equals 1, then a solution to the model exists. If the second element equals 1, the solution is unique. The top 6×6 block of the return *T1* is the matrix F in the solution (see equation 1.2). Similarly, the top 6 rows of *T0* correspond to the matrix g in equation 1.2. The final two rows (and columns in case of T1) correspond to the expectational errors which are not directly relevant for estimation.



mytemp

```
GAM0(eq_IS,v_x)  =  1;
GAM0(eq_IS,v_Ex)  =  -1;
GAM0(eq_IS,v_R)  =  1/sigma;
GAM0(eq_IS,v_Epi)  =  -1/sigma;
GAM0(eq_IS,v_g)  =  -1;
```

$$x_t = E_t x_{t+1} - (1/\sigma)(i_t - E_t\pi_{t+1}) + g_t$$

```
%*******************************************************
%*      2. Phillips Curve
%*******************************************************/
GAM0(eq_PHIL,v_pi)=1;
GAM0(eq_PHIL,v_Epi)=-beta;
GAM0(eq_PHIL,v_x)=-kappa;
GAM0(eq_PHIL,v_u)=-1;
```

$$\pi_t = \beta E_t\pi_{t+1} + \kappa x_t + u_t$$

```
%*******************************************************
%**      3. Policy Rule
%*******************************************************/
GAM0(eq_RULE,v_R)=1;
GAM0(eq_RULE,v_pi)=-delta;
GAM0(eq_RULE,v_v)=-1;
```

$$i_t = \delta\pi_t + v_t$$

```

%*******************************************************
%**      Shock process
%*******************************************************/
%* g **/
GAM0(eq_g,v_g) = 1;
GAM1(eq_g,v_g) = rho1;
 PSI(eq_g,e_x) = 1;
```

$$g_t = \rho_1 g_{t-1} + \varepsilon_{1t}$$

```
%* u **/
GAM0(eq_u,v_u) = 1;
GAM1(eq_u,v_u) = rho2;
 PSI(eq_u,e_pi) = 1;
```

$$u_t = \rho_2 u_{t-1} + \varepsilon_{2t}$$

```
%* v* **/
GAM0(eq_v,v_v) = 1;
GAM1(eq_v,v_v) = rho3;
 PSI(eq_v,e_i) = 1;
```

$$v_t = \rho_3 v_{t-1} + \varepsilon_{3t}$$

```
%*******************************************************
%**      Expectation error
%*******************************************************/
% E(x)
GAM0(eq_Ex,v_x)  = 1;
GAM1(eq_Ex,v_Ex) = 1;
 PPI(eq_Ex,n_x)  = 1;
```

$$x_t - E_{t-1}x_t$$

```
%* E(pi) **/
GAM0(eq_Epi,v_pi)  = 1;
GAM1(eq_Epi,v_Epi) = 1;
 PPI(eq_Epi,n_pi)  = 1;
```

$$\pi_t - E_{t-1}\pi_t$$

```
%*******************************************************************
%**      QZ(generalized Schur) decomposition by GENSYS
%*******************************************************************/
PROBLEM=0;
div = 1;        % criteria for stable roots
% y(t) = T1*y(t-1) + TC + T0*z(t)
% A, B, Q and Z are the QZ decomp matrix
% RC(1) = 1 => existence
% RC(2) = 1 => Unique
[T1,TC,T0,Q,A,B,Z,RC,LOOSE] = gensys(GAM0,GAM1,C,PSI,PPI,div);
Fmat = zeros(neq-neta);
Fmat(1:neq-neta,1:neq-neta) = T1(1:neq-neta,1:neq-neta);
 gmat= [T0(1:neq-neta,:)];
if RC(1)~=1 | RC(2)~=1  % non-unique and existence
    PROBLEM=1;
```

mytemp.html[15/06/2017 15:59:40]

FIGURE 2. Model Solution

For the calibration considered in the example, the following unique solution is produced:

$$\underbrace{\begin{pmatrix} x_t \\ \pi_t \\ i_t \\ g_t \\ u_t \\ v_t \end{pmatrix}}_{\beta_t} = \underbrace{\begin{pmatrix} 0 & 0 & 0 & 1.56 & -4.16 & -1.56 \\ 0 & 0 & 0 & 0.28 & 1.56 & -0.28 \\ 0 & 0 & 0 & 0.43 & 2.34 & 0.26 \\ 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.7 \end{pmatrix}}_{F} \underbrace{\begin{pmatrix} x_{t-1} \\ \pi_{t-1} \\ i_{t-1} \\ g_{t-1} \\ u_{t-1} \\ v_{t-1} \end{pmatrix}}_{\beta_{t-1}} + \underbrace{\begin{pmatrix} 2.23 & -5.94 & -2.23 \\ 0.41 & 2.23 & -0.41 \\ 0.61 & 3.34 & 0.38 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{g} \underbrace{\begin{pmatrix} z_{1t} \\ z_{2t} \\ z_{3t} \end{pmatrix}}_{z_t} \tag{1.7}$$

As discussed above, the solution in equation 1.7 is in the form of a VAR(1). It is now straightforward to produce impulse responses to any of the 3 structural shocks and calculate objects such as variance decomposition. Moreover, the solution can be used to calculate the likelihood function of the model via the Kalman filter. We turn to this next.

1.2. Calculating the log likelihood and log posterior. We treat β_t as our vector of unobserved state (model) variables and consider the model solution $\beta_t = F\beta_{t-1} + gz_t$ as a transition equation of a state-space model. We can obtain data on 'real world' counterparts of the model variables $\tilde{x}_t$ (output gap) , $\tilde{\pi}_t$ (inflation) and $\tilde{\imath}_t$ (short-term interest rate). Typically the output gap would be measured as de-trended GDP, inflation as the log difference of CPI and interest rate as the policy rate set by the central bank. As we do not allow for constants in the model, the data is demeaned.

The observation equation of the model is then given as:

$$\underbrace{\begin{pmatrix} \tilde{x}_t \\ \tilde{\pi}_t \\ \tilde{\imath}_t \end{pmatrix}}_{Y_t} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{H} \underbrace{\begin{pmatrix} x_t \\ \pi_t \\ i_t \\ g_t \\ u_t \\ v_t \end{pmatrix}}_{\beta_t} \tag{1.8}$$

Notice that we have three observable variables $\begin{pmatrix} \tilde{x}_t \\ \tilde{\pi}_t \\ \tilde{\imath}_t \end{pmatrix}$ and three structural shocks in this model. If the number of shocks is less than the number of observables, then the model is *stochastically singular* and the Kalman filter cannot be used to calculate the likelihood function (see Ruge-Murcia (2007) for further explanations on this point).

Equations 1.8 and 1.2 thus gives a linear state-space model:

$$\begin{aligned} Y_t &= H\beta_t \\ \beta_t &= F\beta_{t-1} + \tilde{z}_t \\ var(\tilde{z}_t) &= g(var(z_t))g' \end{aligned}$$

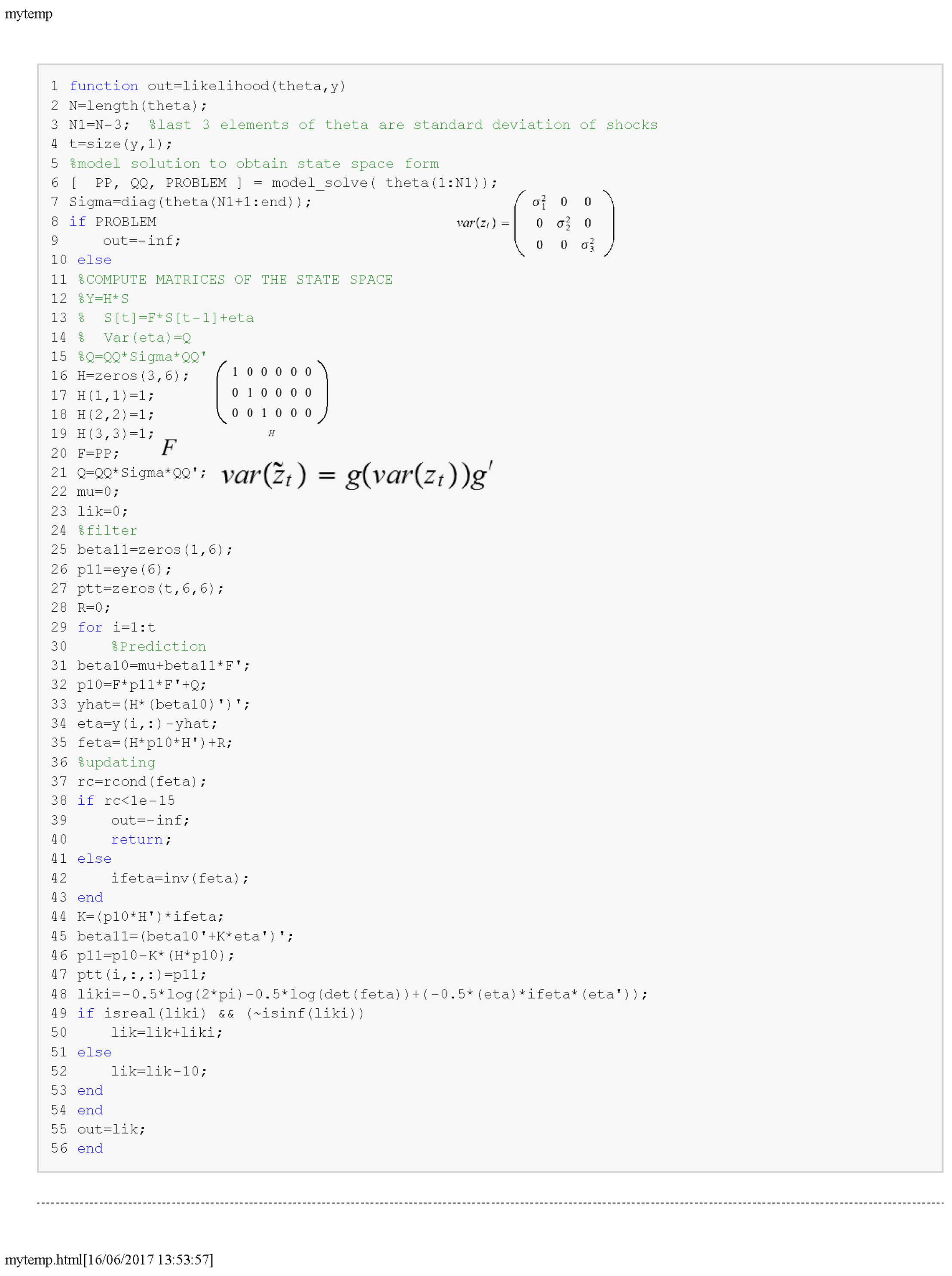
where $var(z_t) = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}$. The file example2.m demonstrates the calculation of the likelihood using artificial data generated from the calibrated model in example1.m. On line 8, the function *likelihood* is called to carry out this calculation. This function is shown in figure 3. The input to the function is the parameter vector $\Theta = (\sigma, \delta, \alpha, \varpi, \rho_1, \rho_2, \rho_3, \sigma_1^2, \sigma_2^2, \sigma_3^2)$. Note from line 6, that when the model is solved, only the parameters $\sigma, \delta, \alpha, \varpi, \rho_1, \rho_2, \rho_3$ are required. With the solution of the model at hand, the next step is to build the matrices of the state-space. However, this is only useful if the model solution exists and is unique. Therefore, lines 8 to 10 are used to terminate the likelihood calculation (and to set the log likelihood to minus infinity), if existence or uniqueness are rejected. The matrices of the state-space are created on lines 16 to 22. Lines 25 and 26 set the initial state vector and its covariance. The Kalman filter begins on line 29. Note that on line 37, the code calculates the reciprocal condition number of the variance of the prediction error. The program terminates if the reciprocal condition number is small indicating that the variance may not have an inverse (which is used to calculate the log likelihood on line 48). The function returns the log likelihood of the model in the scalar out.

Note from Bayes rule that the log posterior is proportional to the log likelihood plus the log prior:

$$\underbrace{\ln g(\Theta|Y_t)}_{\text{log posterior}} \propto \underbrace{\ln L(\Theta|Y_t)}_{\text{log likelihood}} + \underbrace{\ln P(\Theta)}_{\text{log prior}}$$

Therefore to calculate the log posterior we have to evaluate the log prior $\ln P(\Theta)$. For simplicity we assume that there is an independent prior on each parameter and thus $\ln P(\Theta) = \ln P(\sigma) + \ln P(\delta) + \ln P(\varpi) + \ln P(\alpha) + \ln P(\rho_1) + \ln P(\rho_2) + \ln P(\rho_3) + \ln P(\sigma_1^2) + \ln P(\sigma_2^2) + \ln P(\sigma_3^2)$, i.e. the sum of the log prior on each parameter. The question now arises: what prior distributions should be used? In the current example, the following choices for the prior distributions seem reasonable:

Parameter	Distribution(mean,variance)	95% interval
σ	Gamma(1,1)	$[0.02, 3.76]$
δ	Gamma(1.5,1)	$[0.21, 4.00]$
α	Gamma(3,1)	$[1.40, 5.25]$
ϖ	Gamma(1.5,1)	$[0.21, 4.00]$
ρ_1	Beta(0.5,0.2)	$[0.13, 0.87]$
ρ_2	Beta(0.5,0.2)	$[0.13, 0.87]$
ρ_3	Beta(0.5,0.2)	$[0.13, 0.87]$
σ_1^2	Inverse Gamma(1,0.5)	$[0.34, 2.68]$
σ_2^2	Inverse Gamma(1,0.5)	$[0.34, 2.68]$
σ_3^2	Inverse Gamma(1,0.5)	$[0.34, 2.68]$



mytemp

```
function out=likelihood(theta,y)
N=length(theta);
N1=N-3;  %last 3 elements of theta are standard deviation of shocks
t=size(y,1);
%model solution to obtain state space form
[  PP, QQ, PROBLEM ] = model_solve( theta(1:N1));
Sigma=diag(theta(N1+1:end));
if PROBLEM
    out=-inf;
else
%COMPUTE MATRICES OF THE STATE SPACE
%Y=H*S
%  S[t]=F*S[t-1]+eta
%  Var(eta)=Q
%Q=QQ*Sigma*QQ'
H=zeros(3,6);
H(1,1)=1;
H(2,2)=1;
H(3,3)=1;
F=PP;
Q=QQ*Sigma*QQ';
mu=0;
lik=0;
%filter
beta11=zeros(1,6);
p11=eye(6);
ptt=zeros(t,6,6);
R=0;
for i=1:t
    %Prediction
beta10=mu+beta11*F';
p10=F*p11*F'+Q;
yhat=(H*(beta10)')';
eta=y(i,:)-yhat;
feta=(H*p10*H')+R;
%updating
rc=rcond(feta);
if rc<1e-15
    out=-inf;
    return;
else
    ifeta=inv(feta);
end
K=(p10*H')*ifeta;
beta11=(beta10'+K*eta')';
p11=p10-K*(H*p10);
ptt(i,:,:)=p11;
liki=-0.5*log(2*pi)-0.5*log(det(feta))+(-0.5*(eta)*ifeta*(eta'));
if isreal(liki) && (~isinf(liki))
    lik=lik+liki;
else
    lik=lik-10;
end
end
out=lik;
end
```

$$var(z_t) = \begin{pmatrix} \sigma_1^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_3^2 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

H

F

$$var(\tilde{z}_t) = g(var(z_t))g'$$

mytemp.html[16/06/2017 13:53:57]

FIGURE 3. Kalman filter

The prior for σ is assumed to be a Gamma distribution with a mean and variance of 1. The Gamma distribution is appropriate for this parameter as we expect σ to be greater than zero. Note that this (and other) distributions can be parameterised via their means and variances or parameters such as scale, degrees of freedom etc. As discussed in the appendix to Bauwens *et al.* (1999), the mean μ and the variance V of a Gamma(α, β) distribution are given by $\mu = \alpha\beta$ and $V = \alpha\beta^2$. It is then easy to calculate that $\beta = \frac{V}{\mu}$ and $\alpha = \frac{\mu}{\beta}$. This transformation is useful as functions to simulate and evaluate the Gamma distribution in Matlab require the user to provide values for α and β which can be backed out from μ and V. The Beta distribution is chosen as a prior for the autoregressive parameters to ensure that they remain between 0 and 1. Note that for a Beta(α, β) distribution, the mean and variance is defined as: $\mu = \frac{\alpha}{\alpha+\beta}, V = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$. As before, we can specify the prior in terms of μ and V and back out the implied parameters α, β. Finally, we use an Inverse Gamma prior for the shock variances, following the practice in reduced form econometric models. As described in Bauwens *et al.* (1999) (page 292), the Inverse Gamma-2 distribution

$IG(s, \nu)$ has the following first two moments: $\mu = \frac{s}{\nu-2}, V = \frac{2}{\nu-4}\mu^2$. This allows us to solve for s, ν given an value for the mean and the variance. The matlab code logprior.m evaluates these prior distributions at a given value of the model parameters and prior means and variances and returns $\ln P(\Theta)$.

Note that one may want to check the shape of the prior distributions implied by these choices. This can be easily done by simulating random numbers from the prior distribution and examining the percentiles of the resulting distribution (see matlab file simprior.m). The final column of the table above lists the 5th and 95th percentile for each chosen prior. In practice, this interval can provide information about whether the prior distribution covers the range of estimates of these parameters reported in previous papers.

2. Metropolis Hastings Algorithm

We are now ready to proceed to a description of the algorithm to estimate the DSGE model described above. Once the prior distributions are set, the estimation proceeds in the following steps:

(1) Numerically maximise the log posterior $\ln g(\Theta|Y_t)$ to obtain the parameter estimates at the posterior mode Θ^{MAX} and the associated covariance matrix of the estimates V^{MAX}. Set the initial value of the parameters $\Theta^{Old} = \Theta^{MAX}$. The candidate density for this random walk Metropolis algorithm will be of the form $\Theta^{New} = \Theta^{Old} + e, e\tilde{\ } N(0, cV^{MAX})$ where c is a scaling factor used to control the acceptance rate.

(2) Draw the parameter vector from the candidate density and compute the acceptance probability

$$\alpha = \exp\left(\ln g\left(\Theta^{New}|Y_t\right) - \ln g\left(\Theta^{Old}|Y_t\right)\right)$$

If $\alpha > u\tilde{\ }U(0,1)$ accept the draw and set $\Theta^{Old} = \Theta^{New}$. Otherwise Θ^{Old} is retained. Note that the posterior is evaluated in this step when calculating α. Recall from the discussion in the previous section, that this involves the following steps:

(a) Given the vector of parameters Θ, solve the model to obtain the reduced form $\beta_t = F\beta_{t-1} + \tilde{z}_t$.

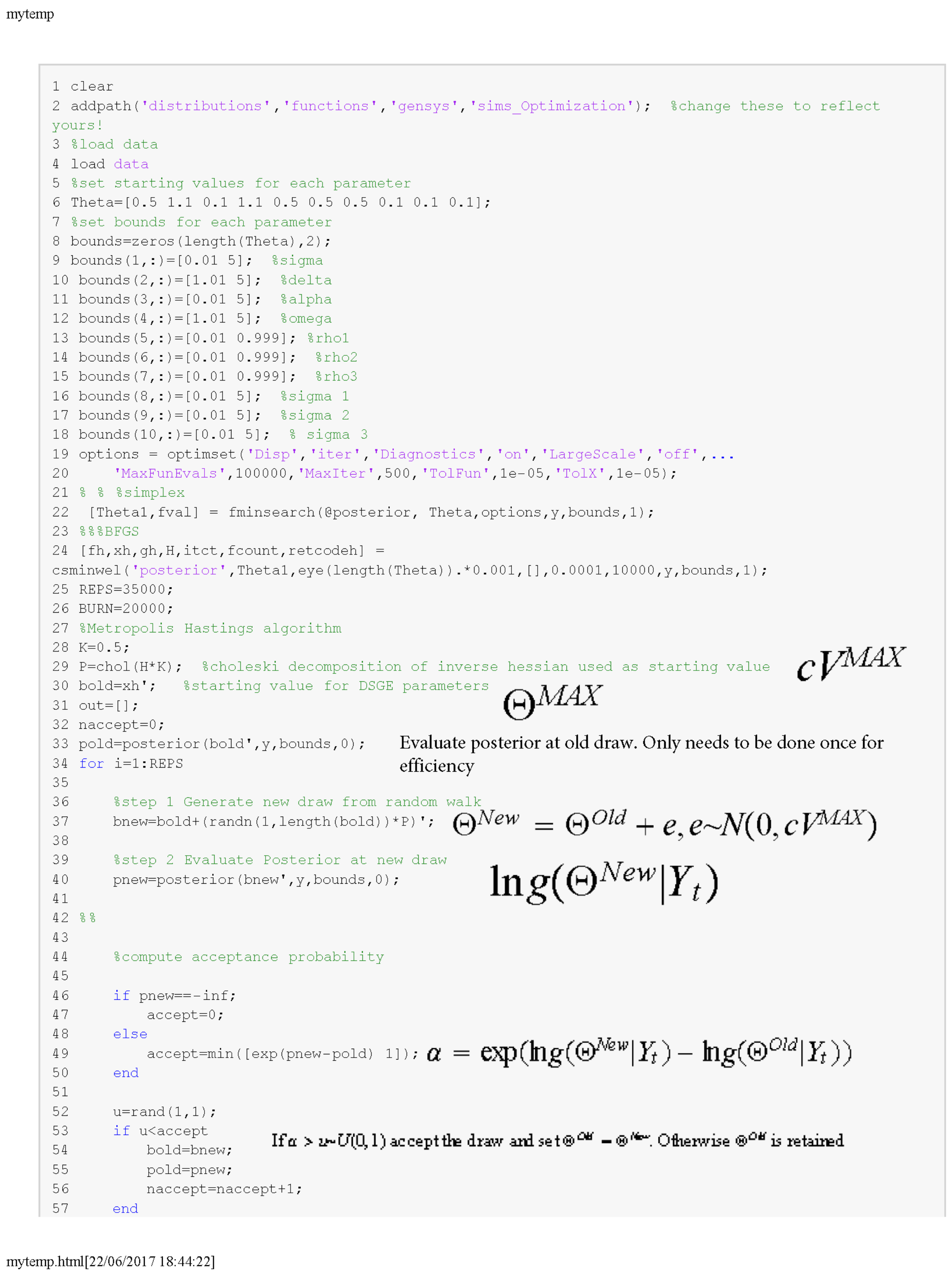
(b) Combine this with an observation equation $Y_t = H\beta_t$ and calculate the likelihood $\ln L(\Theta|Y_t)$ by running the Kalman filter.

(c) Evaluate the prior $\ln P(\Theta)$ and obtain the log posterior as $\ln g(\Theta|Y_t) = \ln L(\Theta|Y_t) + \ln P(\Theta)$.

(3) Repeat step 2 until convergence is detected. The scaling factor c can be used to keep the acceptance rate between 20% and 40%.

The code for the Metropolis algorithm for the DSGE model is shown in figures 4 and 5. Note that this example uses artificial data generated in example1.m. This data is loaded on line 4. Line 6 sets starting values for the model parameters. Lines 8 to 18 set a lower and upper bound for the model parameters. This ensures that implausible values for the parameters are not considered. For example, if one believes that the monetary authority reacts by more than one to one to inflation developments, then $\delta > 1$ but a value bigger than 5 would indicate a degree of activism that has not been observed for many countries. Whenever, the value of the parameters violates these bounds, the value of the likelihood (and posterior) is set to minus infinity. Thus such a parameter vector is disregarded in the algorithm. To obtain starting values for the algorithm, we first numerically maximise the log posterior (or minimise minus log posterior). As this is a difficult task, we proceed by using two optimisation algorithms. First, the starting values are refined by employing 500 iterations of the derivative free Simplex method (line 22). The file posterior.m evaluates the posterior of the DSGE model. If the parameters are within bounds and the model solution exists, then the log likelihood is calculated via the Kalman filter and the log prior is evaluated. Otherwise log posterior is set to minus infinity. The output from the Simplex optimisation is used as starting values in CSMINWEL (line 24). The model parameters at the mode of the log posterior are stored in xh, while H denotes the covariance of the maximum posterior estimates. These estimates are used to initialise the Metropolis algorithm. Line 29 sets the covariance of the candidate density as a scaling factor times H. Line 30 sets the starting values as the maximum posterior estimates. The MH algorithm begins on line 34. Line 37 involves a draw of the model parameters from the random walk canidate density. The posterior is evaluated on line 40 at these parameter values. As explained above, this involves solving the model and evaluating the likelihood and prior distributions. The acceptance probability is calculated on line 49. If this probability is larger than a number from the standard uniform density, then the value of the parameters and log posterior are updated. Otherwise, the previous values are retained. The parameter draws are used to produce three objects.

First, the estimated marginal posterior distributions of the parameters are compared to the prior distributions (obtained using random draws from the prior distributions). The results are shown in figure 6. In the figure the blue vertical line shows the maximum posterior value of each parameter, while the dotted line shows the true values used to generate the data. A comparison of the estimated posterior of the parameters (blue curve) obtained via the MH algorithm and the prior distributions (red curve) can be informative about the information contained in the data about the parameters. Note for example that for the parameter σ, the prior and posterior distributions are quite different indicating that the data leads to an update in the value of σ. In contrast, the prior and posterior distributions for α lie on top of each other. This might imply that the data containes limited information for estimating this parameter. This might occur if the parameter is hard to identify – i.e. the likelihood function is relatively flat with respect to this parameter and attains similar values for different estimates of α. In our example, this may occur as α enters the model in a highly non-linear fashion and it may not be easy to distinguish this parameter from ϖ.



mytemp

```
clear
addpath('distributions','functions','gensys','sims_Optimization');  %change these to reflect yours!
%load data
load data
%set starting values for each parameter
Theta=[0.5 1.1 0.1 1.1 0.5 0.5 0.5 0.1 0.1 0.1];
%set bounds for each parameter
bounds=zeros(length(Theta),2);
bounds(1,:)=[0.01 5];  %sigma
bounds(2,:)=[1.01 5];  %delta
bounds(3,:)=[0.01 5];  %alpha
bounds(4,:)=[1.01 5];  %omega
bounds(5,:)=[0.01 0.999]; %rho1
bounds(6,:)=[0.01 0.999];  %rho2
bounds(7,:)=[0.01 0.999];  %rho3
bounds(8,:)=[0.01 5];  %sigma 1
bounds(9,:)=[0.01 5];  %sigma 2
bounds(10,:)=[0.01 5];  % sigma 3
options = optimset('Disp','iter','Diagnostics','on','LargeScale','off',...
    'MaxFunEvals',100000,'MaxIter',500,'TolFun',1e-05,'TolX',1e-05);
% % %simplex
 [Theta1,fval] = fminsearch(@posterior, Theta,options,y,bounds,1);
%%%BFGS
[fh,xh,gh,H,itct,fcount,retcodeh] = csminwel('posterior',Theta1,eye(length(Theta)).*0.001,[],0.0001,10000,y,bounds,1);
REPS=35000;
BURN=20000;
%Metropolis Hastings algorithm
K=0.5;
P=chol(H*K);  %choleski decomposition of inverse hessian used as starting value
bold=xh';   %starting value for DSGE parameters
out=[];
naccept=0;
pold=posterior(bold',y,bounds,0);
for i=1:REPS

    %step 1 Generate new draw from random walk
    bnew=bold+(randn(1,length(bold))*P)';

    %step 2 Evaluate Posterior at new draw
    pnew=posterior(bnew',y,bounds,0);

%%

    %compute acceptance probability

    if pnew==-inf;
        accept=0;
    else
        accept=min([exp(pnew-pold) 1]);
    end

    u=rand(1,1);
    if u<accept
        bold=bnew;
        pold=pnew;
        naccept=naccept+1;
    end
```

cV^{MAX}

Θ^{MAX}

Evaluate posterior at old draw. Only needs to be done once for efficiency

$\Theta^{New} = \Theta^{Old} + e, e \sim N(0, cV^{MAX})$

$\ln g(\Theta^{New}|Y_t)$

$\alpha = \exp(\ln g(\Theta^{New}|Y_t) - \ln g(\Theta^{Old}|Y_t))$

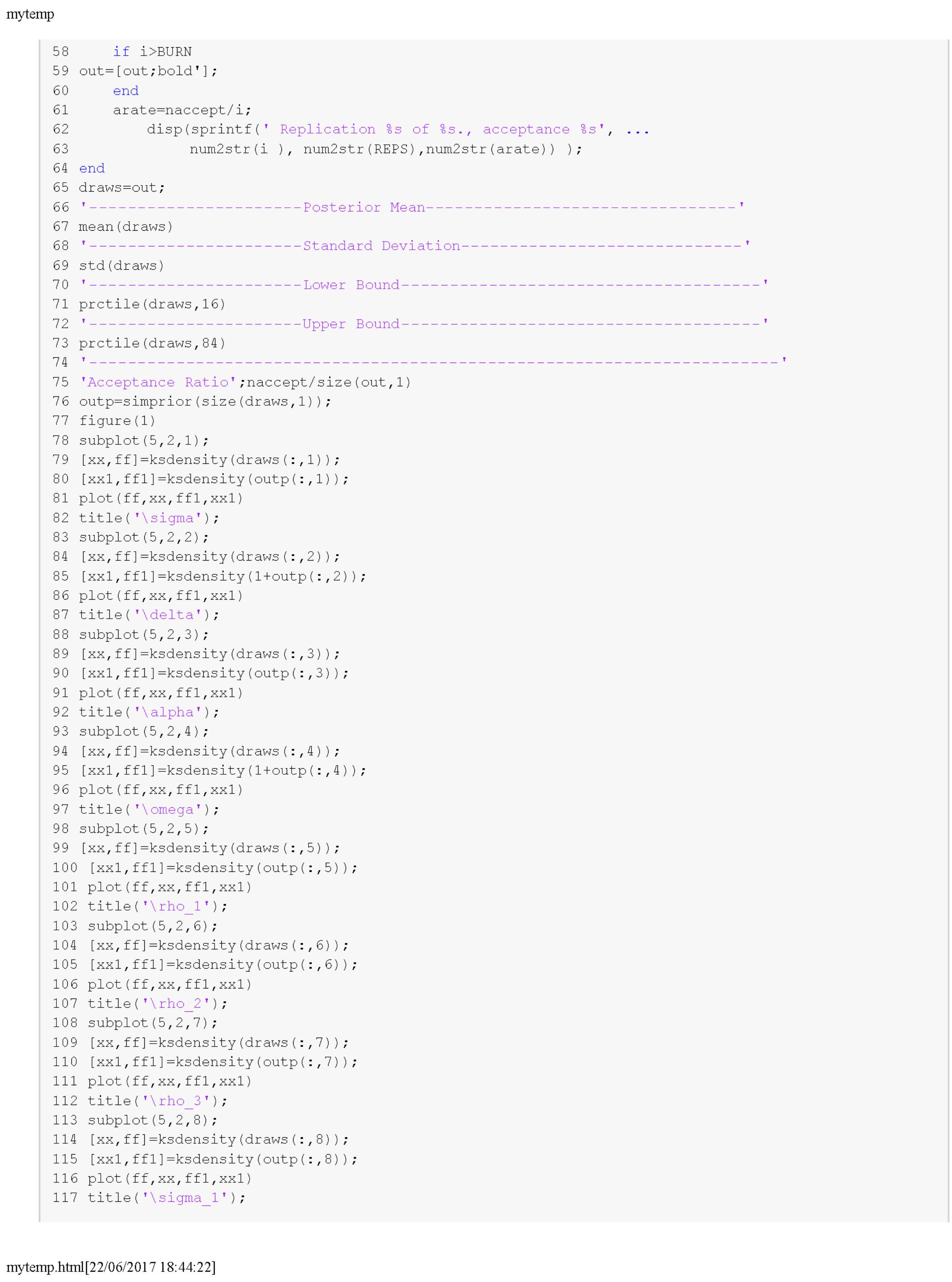
If $\alpha > u \sim U(0,1)$ accept the draw and set $\Theta^{Old} = \Theta^{New}$. Otherwise Θ^{Old} is retained

mytemp.html[22/06/2017 18:44:22]

FIGURE 4. MH algorithm for the DSGE model

Note, secondly that due to the asymmetry of the posterior distributions, it is likely that the posterior mean does not coincide with the maximum posterior estimates. This may also reflect the fact that the maximum of the posterior found by CSMINWEL is a local maximum.

Figure 7 compares the estimated distribution of the response to monetary policy shocks with the response calculated at the true parameter values and shows that the algorithm performs reasonably well. Finally, note that the last part of example3.m (see actual code), calculates the marginal likelihood of the model using the Gelfand and Dey Harmonic mean method (see appendix to previous chapter). This simply requires one to store the value of the log posterior and the parameters for each draw. Then the marginal likelihood $F(Y_t)$ is approximated by using the



mytemp

```
    if i>BURN
out=[out;bold'];
    end
    arate=naccept/i;
        disp(sprintf(' Replication %s of %s., acceptance %s', ...
              num2str(i ), num2str(REPS),num2str(arate)) );
end
draws=out;
'-----------------------Posterior Mean---------------------------------'
mean(draws)
'-----------------------Standard Deviation-----------------------------'
std(draws)
'-----------------------Lower Bound-------------------------------------'
prctile(draws,16)
'-----------------------Upper Bound-------------------------------------'
prctile(draws,84)
'--------------------------------------------------------------------------'
'Acceptance Ratio';naccept/size(out,1)
outp=simprior(size(draws,1));
figure(1)
subplot(5,2,1);
[xx,ff]=ksdensity(draws(:,1));
[xx1,ff1]=ksdensity(outp(:,1));
plot(ff,xx,ff1,xx1)
title('\sigma');
subplot(5,2,2);
[xx,ff]=ksdensity(draws(:,2));
[xx1,ff1]=ksdensity(1+outp(:,2));
plot(ff,xx,ff1,xx1)
title('\delta');
subplot(5,2,3);
[xx,ff]=ksdensity(draws(:,3));
[xx1,ff1]=ksdensity(outp(:,3));
plot(ff,xx,ff1,xx1)
title('\alpha');
subplot(5,2,4);
[xx,ff]=ksdensity(draws(:,4));
[xx1,ff1]=ksdensity(1+outp(:,4));
plot(ff,xx,ff1,xx1)
title('\omega');
subplot(5,2,5);
[xx,ff]=ksdensity(draws(:,5));
[xx1,ff1]=ksdensity(outp(:,5));
plot(ff,xx,ff1,xx1)
title('\rho_1');
subplot(5,2,6);
[xx,ff]=ksdensity(draws(:,6));
[xx1,ff1]=ksdensity(outp(:,6));
plot(ff,xx,ff1,xx1)
title('\rho_2');
subplot(5,2,7);
[xx,ff]=ksdensity(draws(:,7));
[xx1,ff1]=ksdensity(outp(:,7));
plot(ff,xx,ff1,xx1)
title('\rho_3');
subplot(5,2,8);
[xx,ff]=ksdensity(draws(:,8));
[xx1,ff1]=ksdensity(outp(:,8));
plot(ff,xx,ff1,xx1)
title('\sigma_1');
```

mytemp.html[22/06/2017 18:44:22]

FIGURE 5. MH algorithm for the DSGE model

equation $\frac{1}{F(Y_t)}=\frac{1}{M}\sum_{j=1}^{M}\frac{f(\Theta_j)}{g(\Theta_j|Y_t)}$ where $f(\Theta_j)=\frac{1}{p(2\pi)^{k/2}}\left|\hat{\Sigma}\right|^{-1/2}\exp\left[-0.5\left(\Theta_j-\hat{\Theta}\right)\hat{\Sigma}^{-1}\left(\Theta_j-\hat{\Theta}\right)'\right]\times I\left(\Theta_j\in\tilde{\Theta}\right)$ where $\hat{\Theta}$ and $\hat{\Sigma}$ denote the mean and covariance of the draws of the DSGE parameters and $I\left(\Theta_j\in\tilde{\Theta}\right)$ is an indicator function that equals 1 if $-0.5\left(\Theta_j-\hat{\Theta}\right)\hat{\Sigma}^{-1}\left(\Theta_j-\hat{\Theta}\right)'\leq\chi^2_{1-p}(k)$ with k equal to number of parameters.

The estimated log marginal likelihood for this model is -945.61. The file example4.m estimates a version of the model that restricts $\rho_1=0$. The estimated log marginal likelihood of this restricted model is -1072.34. Unsurprisingly,

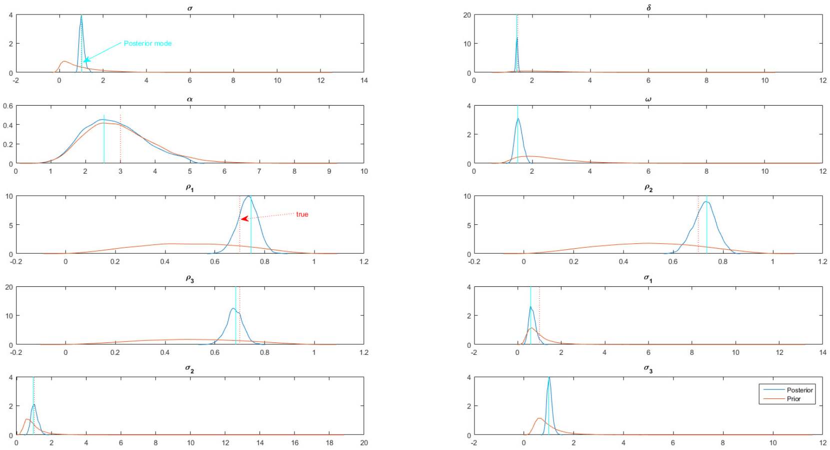


Figure 6. Posterior estimates

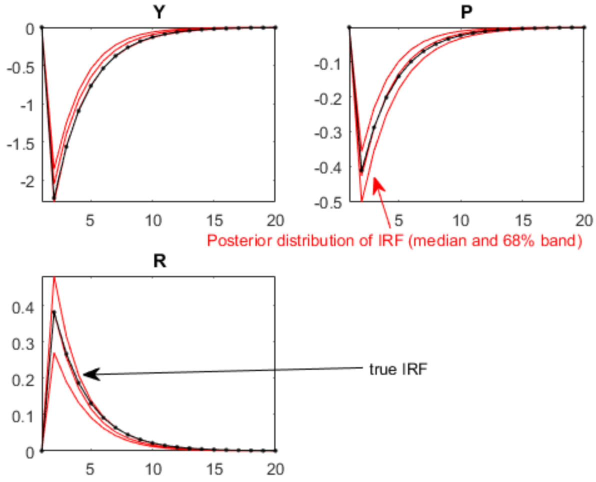


Figure 7. Impulse response (IRF) to a monetary policy shock

the (artificial) data favour the benchmark model. More formally one can consider the posterior odds of the benchmark model M_0 against the restricted model M_1. The posterior odds ratio is defined as

$$PO_{01} = \frac{\pi_0}{\pi_1} \times \frac{F\left(Y_t|M_0\right)}{F\left(Y_t|M_1\right)}$$

The first term in this expression is the prior odds ratio, i.e. the ratio of the prior probabilities (weights) attached to the two models. If we assume that $\pi_0 = \pi_1$, then the posterior odds ratio (collapses to the Bayes Factor) and evaluates to $PO_{01} = \exp(-945.61 - (-1072.34)) = 1.09 \times 10^{55}$ suggesting overwhelming evidence in favour of M_0. Lubik and Schorfheide (2007) provide an interesting application of model comparison across DSGE models.

3. Further Reading

- Canova and Sala (2009) provide a detailed discussion of identification issues in DSGE models.
- A book on Bayesian estimation of DSGE models by Herbst and Schorfheide (2015)

Part 3

Further Topics

CHAPTER 7

State-Space models with time-varying parameters

1. Introduction

A number of recent papers have shown that time-variation in the parameters and shock variances of state-space models can be useful in many empirical contexts. Some recent examples include Negro and Otrok (2008) and Mumtaz and Surico (2012) who estimate dynamic factor models (DFM) with time-varying parameters and stochastic volatility, while STOCK and WATSON (2007) introduce stochastic volatility in an unobserved component model. This chapter considers the Gibbs sampling algorithm for such models in detail. In particular, it describes the algorithm and code for estimating the dynamic factor model of Mumtaz and Surico (2012). The methods reviewed in this chapter can be applied to several variants of these extended state-space models considered in the literature.

2. A dynamic factor model with time-varying parameters and stochastic volatility

Mumtaz and Surico (2012) consider the following DFM

$$\pi_{it} = \beta_i^W F_t^W + \beta_i^c F_t^c + e_{it} \tag{2.1}$$

where π_{it} is a cross-country data set with N time-series (country-specific inflation measures in Mumtaz and Surico (2012)), F_t^W is a world factor, F_t^c denotes a set of country-specific factors for $c = 1, 2, ..., C$ countries in the data set and e_{it} are the idisyncratic components/factors for $i = 1, 2, ...N$ series. β_i^W and β_i^c denote the factor loadings on the world and country factors.

The world factor follows an AR(p) process:

$$\begin{aligned} F_t^W &= \alpha_t^W + \sum_{p=1}^{P} \rho_{p,t}^W F_{t-p}^W + \left(h_t^W\right)^{\frac{1}{2}} \varepsilon_t^W, & (2.2) \\ & \varepsilon_t^W \tilde{} N(0,1) & (2.3) \end{aligned}$$

This AR model features time-varying coefficients and stochastic volatility. The coefficients $\Phi_t^W = vec\left(\left[\alpha_t^W, \rho_{1,t}^W, .., \rho_{P,t}^W\right]\right)$ are assumed to evolve as random walks:

$$\begin{aligned} \Phi_t^W &= \Phi_{t-1}^W + \left(Q^W\right)^{1/2} \eta_t^W, \\ & \eta_t^W \tilde{} N(0,1) \end{aligned}$$

Similarly the log-variances $\ln h_t^W$ also evolve as a random walk:

$$\begin{aligned} \ln h_t^W &= \ln h_{t-1}^W + \left(g^W\right)^{1/2} u_t^W, \\ & u_t^W \tilde{} N(0,1) \end{aligned}$$

Exactly the same formulation is used for the country factors, with each of them described by an AR(p) process with time-varying parameters and stochastic volatility . That is the country factors follow:

$$\begin{aligned} F_t^c &= \alpha_t^c + \sum_{p=1}^{P} \rho_{p,t}^c F_{t-p}^c + \left(h_t^c\right)^{\frac{1}{2}} \varepsilon_t^c, & (2.4) \\ & \varepsilon_t^c \tilde{} N(0,1) & (2.5) \end{aligned}$$

The coefficients $\Phi_t^c = vec\left(\left[\alpha_t^c, \rho_{1,t}^c, .., \rho_{P,t}^c\right]\right)$ are assumed to evolve as random walks:

$$\begin{aligned} \Phi_t^c &= \Phi_{t-1}^c + \left(Q^c\right)^{1/2} \eta_t^c, \\ & \eta_t^C \tilde{} N(0,1) \end{aligned}$$

Similarly the log-variances $\ln h_t^c$ also evolve as a random walk:

$$\begin{aligned} \ln h_t^c &= \ln h_{t-1}^c + \left(g^c\right)^{1/2} u_t^c, \\ & u_t^c \tilde{} N(0,1) \end{aligned}$$

Finally, the idiosyncratic factors follow an AR(1) process with time-varying coefficients and stochastic volatility:

$$\begin{aligned} e_{it} &= \rho_{i,t} e_{it-1} + \left(h_{it}\right)^{\frac{1}{2}} \varepsilon_{it}, \\ & \varepsilon_{it} \tilde{} N(0,1) \end{aligned}$$

where:

$$\begin{aligned} \rho_{i,t} &= \rho_{i,t-1} + \left(q_i^{1/2}\right)\eta_{it}, \eta_{it}\tilde{}N(0,1) \\ \ln h_{it} &= \ln h_{it-1} + (g_i)^{1/2} u_{it}, u_{it}\tilde{}N(0,1) \end{aligned}$$

In short, this DFM features time-varying parameters in the transition equations of all the factors. Notice that $var(\pi_{it}) = \left(\beta_i^W\right)^2 var\left(F_t^W\right) + \left(\beta_i^c\right)^2 var\left(F_t^c\right) + var\left(e_{it}\right)$. Because of the time-varying parameters, the variances $var\left(F_t^W\right), var\left(F_t^c\right), var\left(e_{it}\right)$ are time-varying. Thus the contribution of each of these factors to the total variance $var\left(\pi_{it}\right)$ changes over time. This feature is heavily used in Mumtaz and Surico (2012).

The DFM uses a simple assumption to distinguish between the world and country factors. Re-write the observation equations as

$$\pi_{it} = \beta F_t + e_{it}$$

where $F_t = \left[F_t^W, F_t^c\right]$ for $c = 1, 2, ..., C$. Assume that $N = 8$ and $C = 2$ for simplicity. Then the factor loading matrix β looks as follows:

$$\beta = \begin{pmatrix} \beta_1^W & \beta_1^1 & 0 \\ \beta_2^W & \beta_2^1 & 0 \\ \beta_3^W & \beta_3^1 & 0 \\ \beta_4^W & \beta_4^1 & 0 \\ \beta_5^W & 0 & \beta_1^2 \\ \beta_6^W & 0 & \beta_2^2 \\ \beta_7^W & 0 & \beta_3^2 \\ \beta_8^W & 0 & \beta_4^2 \end{pmatrix}$$

In other words, the world factor loads on all 8 series. The country factors only load on the 4 series for each country.

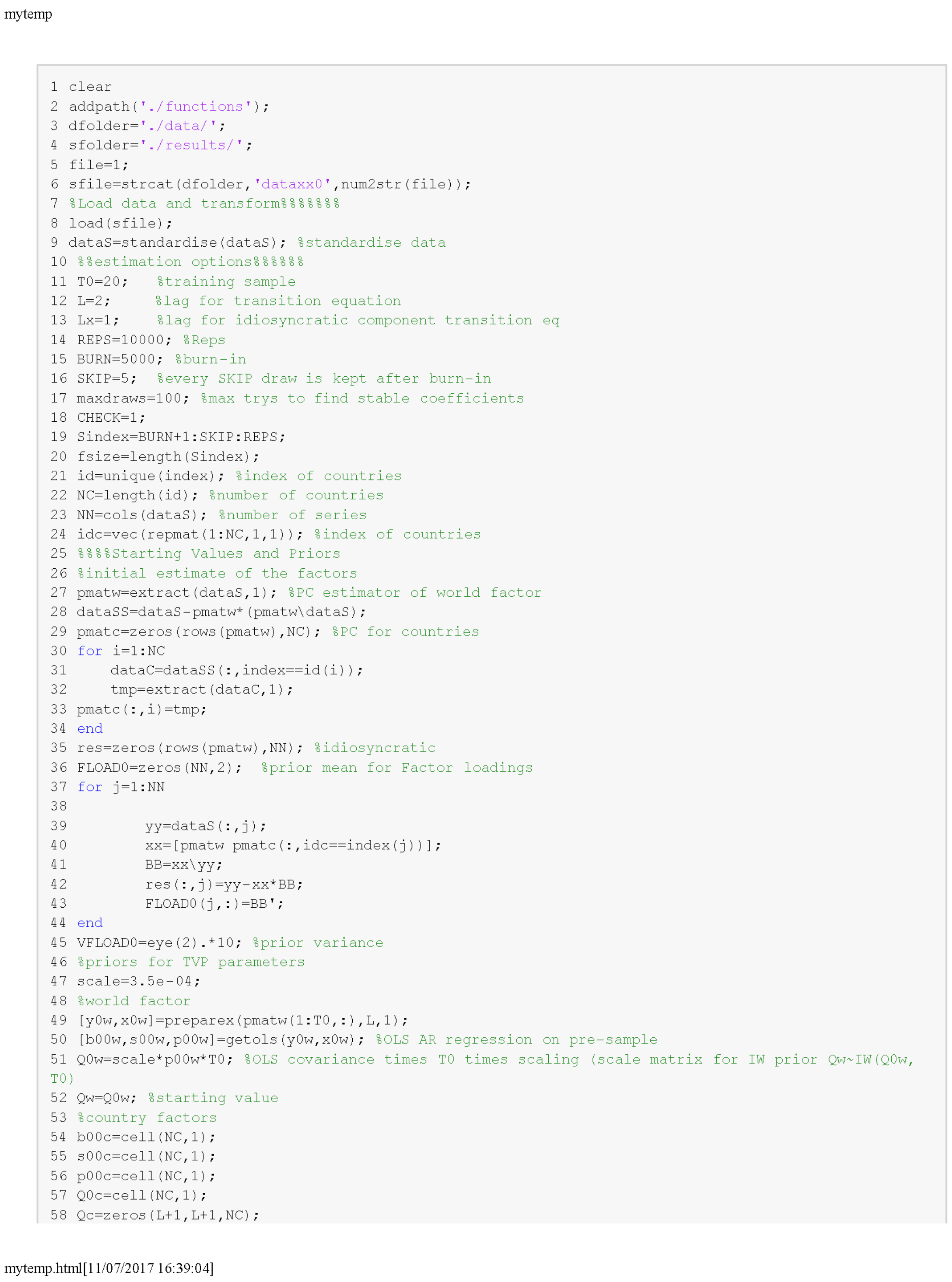
3. Priors and the Gibbs Sampling algorithm

The model contains a large number of unknown parameters and state variables. In particular, the following (blocks of) parameters need to be estimated: β (factor loadings) , F_t (factors) , h_t $\left(\text{stochastic volatilties: } h_t^W, h_t^c, h_{it}\ \right)$, Φ_t $\left(\text{time-varying AR coefficients: } \Phi_t^W, \Phi_t^c, \rho_{it}\right)$, the variance-covariance matrices Q^W, Q^c and the variances, q_i, g^W, g^c, g_i. The Gibbs algorithm thus samples from the conditional posterior distribution of all these parameters. In the sections below we describe the priors used for these parameters and each step of the Gibbs algorithm. As we go through the algorithm we also describe the corresponding Matlab code. The complete code is in the file *example1.m.*

3.1. Priors and starting values. We start the description by explaining how the starting values and priors are set for each parameter. The code for this is shown in figures 1 to 3. Note that this example uses artificial data that is generated from this model by running *generate_ data.m.* This data is loaded on line 8 and standardised on line 9. Line 11 sets a training sample of 20 to calibrate some priors as discussed below.

(1) Factor loadings β: In order to set the prior, we obtain estimates of F_t by using a principal component estimator. With an estimate of the factors $\hat{F}_t^W, \hat{F}_t^c$ in hand , estimates of the factor loadings can easily be obtained by treating the equation $\pi_{it} = \beta_i^W \hat{F}_t^W + \beta_i^c \hat{F}_t^c + e_{it}$ as N linear regressions and obtain the factor loadings via OLS. The prior we use is then given as $p(\beta)\tilde{}N(\hat{\beta}_i, V_\beta)$ where $\hat{\beta}_i$ is the OLS estimate of β_i^W and β_i^c. In figures 1 to 3 this is done starting line 27 that estimates the world factor. The country factor is then estimated via principal components on the residual data (after removing the impact of the world factor) for each country on lines 30 to 34. Lines 37 to 44 run the OLS regression for each series storing the prior mean in *FLOAD0.* Line 45 sets the prior variance V_β. Note that the residuals from these regressions are stored in the matrix *res.*

(2) Priors for Q^W, Q^c, q_i: The prior is assumed to be inverse Wishart. We follow the approach developed in papers by Cogley and Sargent and use a training sample to estimate the scale matrices. Given T_0 training sample observations for $\hat{F}_t^W$ we can estimate an AR model via OLS: $\hat{F}_t^W = \hat{\alpha}^W + \sum_{p=1}^P \hat{\rho}_p^W \hat{F}_{t-p}^W + r_t^W$. Call the coefficient covariance from this regression $\hat{V}^W$. Then the prior for Q^W is set as $p\left(Q^W\right)\tilde{}IW\left(Q_0^W, T_0\right)$ where $Q_0^W = \hat{V}^W \times T_0 \times scale$ where $scale = 3.5 \times 10^{-4}$ in our application. The prior for Q^c for $c = 1, 2, ..C$ is set in exactly the same manner using the principal component estimates of the country factors. The same procedure is used to set the prior for q_i. The scale matrices obtained from this procedure are also used as initial values for these variances. For example Q^W is initialised by assuming that $Q^W = Q_0^W$. For Q_0^W these calculations are implemented on lines 49 to 51. Line 49 prepares the dependent and independent variables for an AR(2) regression. Line 50 runs the regression obtaining the coefficient covariance p00w which is used to calculate the scale matrix on line 51. Exactly the same procedure is repeated for each country factor on lines 53 to 64 and for each idiosyncratic factor on lines 66 to 76. Note that these OLS regressions are also used to obtain the initial conditions for the time-varying coefficients and their variances. For example



mytemp

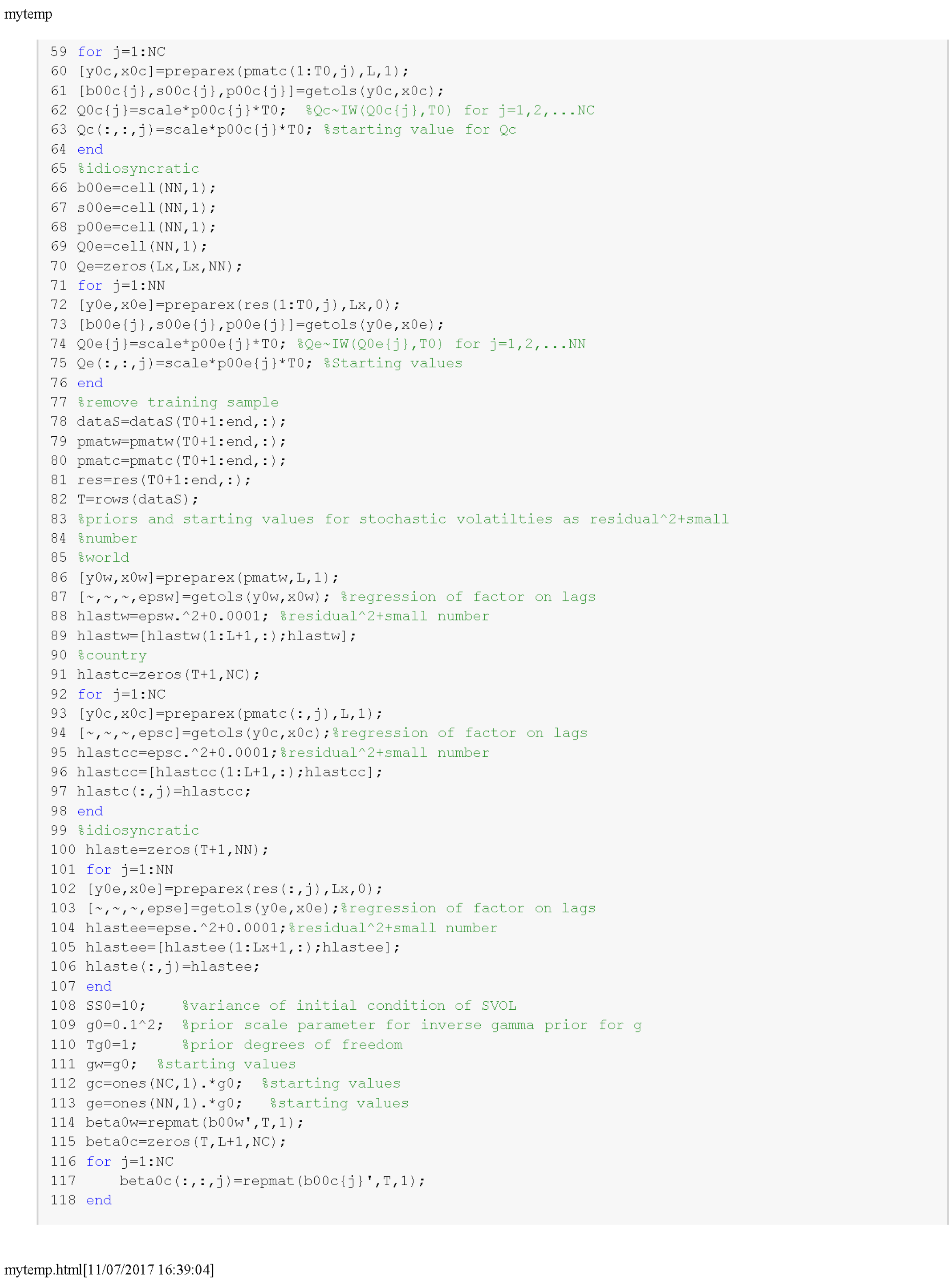
```
1 clear
2 addpath('./functions');
3 dfolder='./data/';
4 sfolder='./results/';
5 file=1;
6 sfile=strcat(dfolder,'dataxx0',num2str(file));
7 %Load data and transform%%%%%%
8 load(sfile);
9 dataS=standardise(dataS); %standardise data
10 %%estimation options%%%%%%
11 T0=20;   %training sample
12 L=2;     %lag for transition equation
13 Lx=1;    %lag for idiosyncratic component transition eq
14 REPS=10000; %Reps
15 BURN=5000; %burn-in
16 SKIP=5;  %every SKIP draw is kept after burn-in
17 maxdraws=100; %max trys to find stable coefficients
18 CHECK=1;
19 Sindex=BURN+1:SKIP:REPS;
20 fsize=length(Sindex);
21 id=unique(index); %index of countries
22 NC=length(id); %number of countries
23 NN=cols(dataS); %number of series
24 idc=vec(repmat(1:NC,1,1)); %index of countries
25 %%%%Starting Values and Priors
26 %initial estimate of the factors
27 pmatw=extract(dataS,1); %PC estimator of world factor
28 dataSS=dataS-pmatw*(pmatw\dataS);
29 pmatc=zeros(rows(pmatw),NC); %PC for countries
30 for i=1:NC
31     dataC=dataSS(:,index==id(i));
32     tmp=extract(dataC,1);
33 pmatc(:,i)=tmp;
34 end
35 res=zeros(rows(pmatw),NN); %idiosyncratic
36 FLOAD0=zeros(NN,2);  %prior mean for Factor loadings
37 for j=1:NN
38
39        yy=dataS(:,j);
40        xx=[pmatw pmatc(:,idc==index(j))];
41        BB=xx\yy;
42        res(:,j)=yy-xx*BB;
43        FLOAD0(j,:)=BB';
44 end
45 VFLOAD0=eye(2).*10; %prior variance
46 %priors for TVP parameters
47 scale=3.5e-04;
48 %world factor
49 [y0w,x0w]=preparex(pmatw(1:T0,:),L,1);
50 [b00w,s00w,p00w]=getols(y0w,x0w); %OLS AR regression on pre-sample
51 Q0w=scale*p00w*T0; %OLS covariance times T0 times scaling (scale matrix for IW prior Qw~IW(Q0w,
T0)
52 Qw=Q0w; %starting value
53 %country factors
54 b00c=cell(NC,1);
55 s00c=cell(NC,1);
56 p00c=cell(NC,1);
57 Q0c=cell(NC,1);
58 Qc=zeros(L+1,L+1,NC);
```

mytemp.html[11/07/2017 16:39:04]

Figure 1. Setting priors

$\Phi_0^W \tilde{} N\left(\Phi_{0|0}^W, P_{\Phi W}\right)$ where $\Phi_{0|0}^W$ is set to the OLS estimates of the coefficients while $P_{\Phi W}$ is the coefficient covariance obtained via OLS.

(3) Starting values for h_t^W, h_t^c, h_{it}: Lines 78 to 82 remove the training sample from the data and the initial estimates of the factors. Line 86 and 87 again conducts the OLS regression $\hat{F}_t^W = \hat{\alpha}^W + \sum_{p=1}^P \hat{\rho}_p^W \hat{F}_{t-p}^W + r_t^W$ using the estimation sample. The starting value for h_t^W is set as $\left(r_t^W\right)^2 + 0.0001$. As explained in Chapter 5 and the description below, this starting value is used in the Metropolis Step devised by Jacquier *et al.*



mytemp

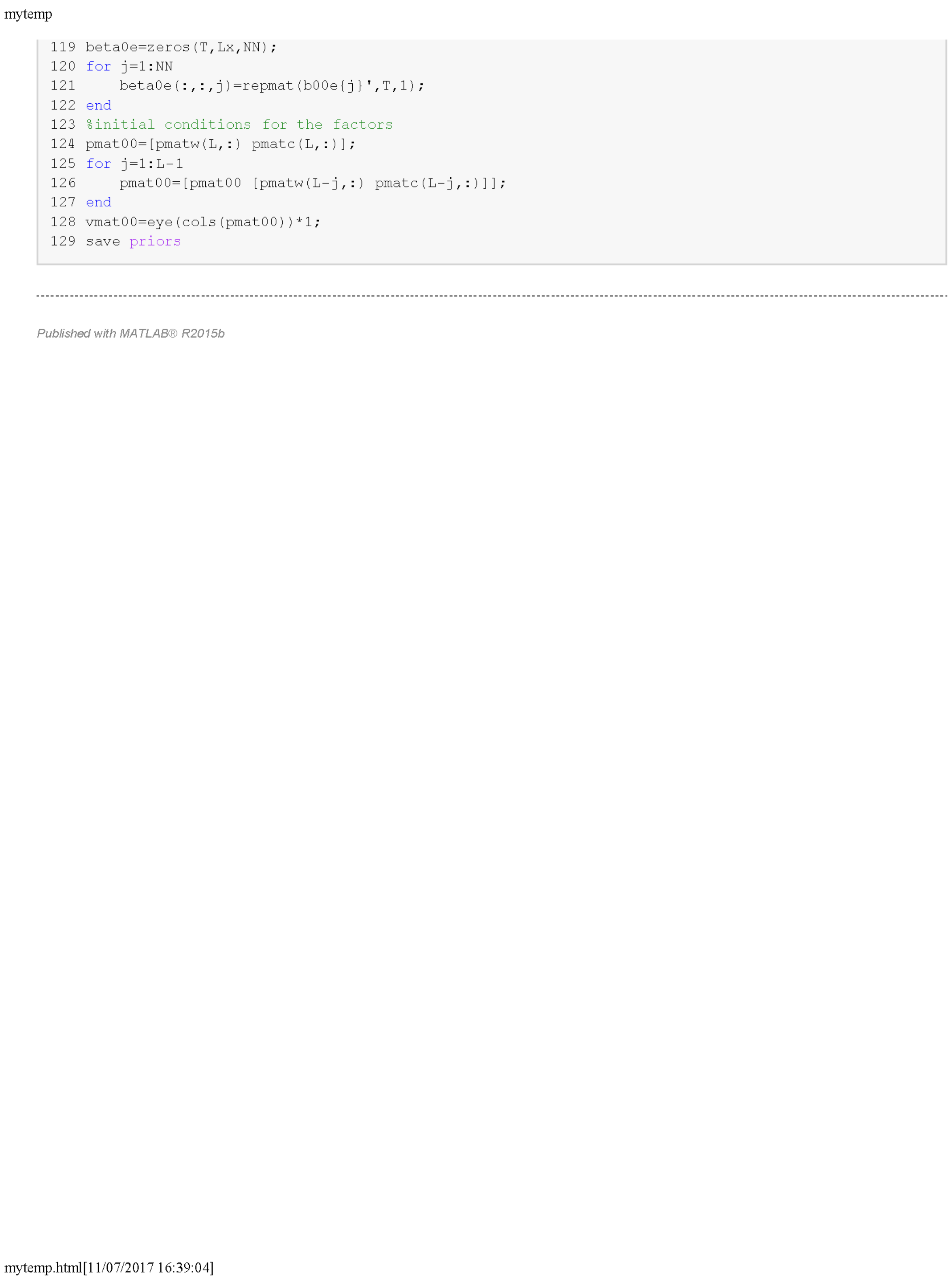
```
59 for j=1:NC
60 [y0c,x0c]=preparex(pmatc(1:T0,j),L,1);
61 [b00c{j},s00c{j},p00c{j}]=getols(y0c,x0c);
62 Q0c{j}=scale*p00c{j}*T0;  %Qc~IW(Q0c{j},T0) for j=1,2,...NC
63 Qc(:,:,j)=scale*p00c{j}*T0; %starting value for Qc
64 end
65 %idiosyncratic
66 b00e=cell(NN,1);
67 s00e=cell(NN,1);
68 p00e=cell(NN,1);
69 Q0e=cell(NN,1);
70 Qe=zeros(Lx,Lx,NN);
71 for j=1:NN
72 [y0e,x0e]=preparex(res(1:T0,j),Lx,0);
73 [b00e{j},s00e{j},p00e{j}]=getols(y0e,x0e);
74 Q0e{j}=scale*p00e{j}*T0; %Qe~IW(Q0e{j},T0) for j=1,2,...NN
75 Qe(:,:,j)=scale*p00e{j}*T0; %Starting values
76 end
77 %remove training sample
78 dataS=dataS(T0+1:end,:);
79 pmatw=pmatw(T0+1:end,:);
80 pmatc=pmatc(T0+1:end,:);
81 res=res(T0+1:end,:);
82 T=rows(dataS);
83 %priors and starting values for stochastic volatilties as residual^2+small
84 %number
85 %world
86 [y0w,x0w]=preparex(pmatw,L,1);
87 [~,~,~,epsw]=getols(y0w,x0w); %regression of factor on lags
88 hlastw=epsw.^2+0.0001; %residual^2+small number
89 hlastw=[hlastw(1:L+1,:);hlastw];
90 %country
91 hlastc=zeros(T+1,NC);
92 for j=1:NC
93 [y0c,x0c]=preparex(pmatc(:,j),L,1);
94 [~,~,~,epsc]=getols(y0c,x0c);%regression of factor on lags
95 hlastcc=epsc.^2+0.0001;%residual^2+small number
96 hlastcc=[hlastcc(1:L+1,:);hlastcc];
97 hlastc(:,j)=hlastcc;
98 end
99 %idiosyncratic
100 hlaste=zeros(T+1,NN);
101 for j=1:NN
102 [y0e,x0e]=preparex(res(:,j),Lx,0);
103 [~,~,~,epse]=getols(y0e,x0e);%regression of factor on lags
104 hlastee=epse.^2+0.0001;%residual^2+small number
105 hlastee=[hlastee(1:Lx+1,:);hlastee];
106 hlaste(:,j)=hlastee;
107 end
108 SS0=10;    %variance of initial condition of SVOL
109 g0=0.1^2;  %prior scale parameter for inverse gamma prior for g
110 Tg0=1;     %prior degrees of freedom
111 gw=g0;  %starting values
112 gc=ones(NC,1).*g0;  %starting values
113 ge=ones(NN,1).*g0;   %starting values
114 beta0w=repmat(b00w',T,1);
115 beta0c=zeros(T,L+1,NC);
116 for j=1:NC
117     beta0c(:,:,j)=repmat(b00c{j}',T,1);
118 end
```

mytemp.html[11/07/2017 16:39:04]

Figure 2. Setting Priors

(2004) to sample from the conditional posterior of the stochastic volatilities. In exactly the same manner, the starting values for h_t^c and h_{it} are set on lines 90 to 107.

(4) Priors for g^W, g^c, g_i: The priors for these variances is inverse Gamma: $IG\,(g_0, T_g)$. In our application we set $T_g = 1, g_0 = 0.01$ (line 109). Note that the metropolis algorithm to draw the stochastic volatility uses a prior for the initial condition. For example one needs to set $\ln h_0^W\,\tilde{}\,N\left(\bar{\mu}, \bar{\sigma}\right)$. We set $\bar{\sigma} = 10$ (line 108) and use the OLS estimate of the error variance (in step 2) to set the prior mean $\bar{\mu}$ (i.e. *s00w* on line 50). The prior for the initial condition for all stochastic volatilties is set this way.



mytemp

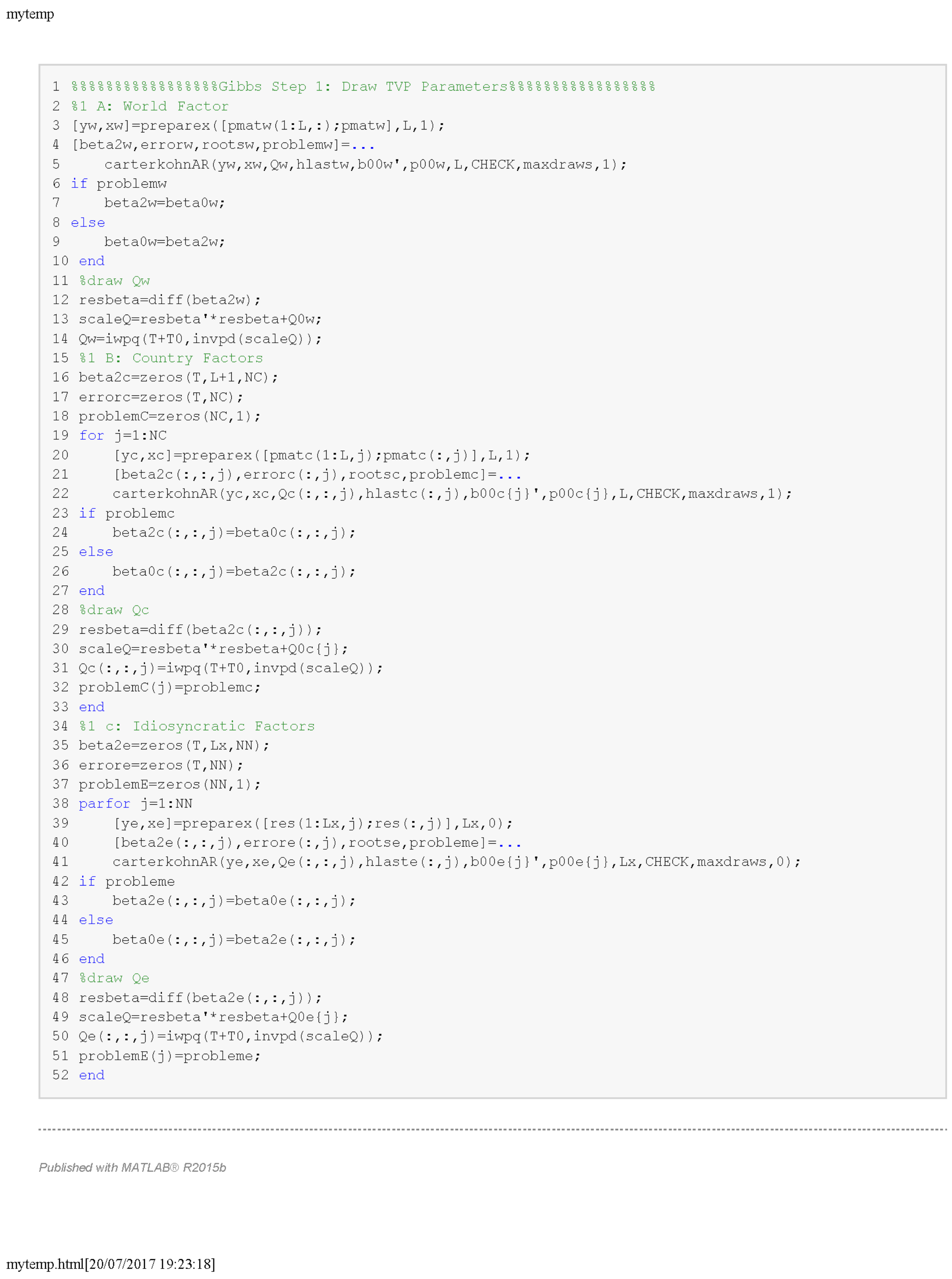
```
119 beta0e=zeros(T,Lx,NN);
120 for j=1:NN
121     beta0e(:,:,j)=repmat(b00e{j}',T,1);
122 end
123 %initial conditions for the factors
124 pmat00=[pmatw(L,:) pmatc(L,:)];
125 for j=1:L-1
126     pmat00=[pmat00 [pmatw(L-j,:) pmatc(L-j,:)]];
127 end
128 vmat00=eye(cols(pmat00))*1;
129 save priors
```

Published with MATLAB® R2015b

mytemp.html[11/07/2017 16:39:04]

Figure 3. Setting Priors

(5) Initial value of the factors: We assume that $\tilde{F}_0\tilde{\ }N\left(F_{0|0},P_F\right)$. Note that: $\tilde{F}_t=\begin{pmatrix} F_t^W \\ F_t^c \\ F_{t-1}^W \\ F_{t-1}^c \\ . \\ . \\ F_{t-P-1}^W \\ F_{t-P-1}^c \end{pmatrix}$ Lines 124 to 127 sets this vector using the initial principal component estimates of the factors. P_F is set equal to an identity matrix on line 128.



mytemp

```
%%%%%%%%%%%%%%%%%Gibbs Step 1: Draw TVP Parameters%%%%%%%%%%%%%%%%%
%1 A: World Factor
[yw,xw]=preparex([pmatw(1:L,:);pmatw],L,1);
[beta2w,errorw,rootsw,problemw]=...
    carterkohnAR(yw,xw,Qw,hlastw,b00w',p00w,L,CHECK,maxdraws,1);
if problemw
    beta2w=beta0w;
else
    beta0w=beta2w;
end
%draw Qw
resbeta=diff(beta2w);
scaleQ=resbeta'*resbeta+Q0w;
Qw=iwpq(T+T0,invpd(scaleQ));
%1 B: Country Factors
beta2c=zeros(T,L+1,NC);
errorc=zeros(T,NC);
problemC=zeros(NC,1);
for j=1:NC
    [yc,xc]=preparex([pmatc(1:L,j);pmatc(:,j)],L,1);
    [beta2c(:,:,j),errorc(:,j),rootsc,problemc]=...
    carterkohnAR(yc,xc,Qc(:,:,j),hlastc(:,j),b00c{j}',p00c{j},L,CHECK,maxdraws,1);
if problemc
    beta2c(:,:,j)=beta0c(:,:,j);
else
    beta0c(:,:,j)=beta2c(:,:,j);
end
%draw Qc
resbeta=diff(beta2c(:,:,j));
scaleQ=resbeta'*resbeta+Q0c{j};
Qc(:,:,j)=iwpq(T+T0,invpd(scaleQ));
problemC(j)=problemc;
end
%1 c: Idiosyncratic Factors
beta2e=zeros(T,Lx,NN);
errore=zeros(T,NN);
problemE=zeros(NN,1);
parfor j=1:NN
    [ye,xe]=preparex([res(1:Lx,j);res(:,j)],Lx,0);
    [beta2e(:,:,j),errore(:,j),rootse,probleme]=...
    carterkohnAR(ye,xe,Qe(:,:,j),hlaste(:,j),b00e{j}',p00e{j},Lx,CHECK,maxdraws,0);
if probleme
    beta2e(:,:,j)=beta0e(:,:,j);
else
    beta0e(:,:,j)=beta2e(:,:,j);
end
%draw Qe
resbeta=diff(beta2e(:,:,j));
scaleQ=resbeta'*resbeta+Q0e{j};
Qe(:,:,j)=iwpq(T+T0,invpd(scaleQ));
problemE(j)=probleme;
end
```

Published with MATLAB® R2015b

mytemp.html[20/07/2017 19:23:18]

Figure 4. Draw time-varying coefficients.

3.2. The Gibbs sampling algorithm. The Gibbs algorithm involves sampling from the following conditional posterior distributions:

(1) Sample from $H\left(\Phi_t^W|\Xi\right)$: Here Ξ all remaining parameters and states in the model. Given a draw of the world factor, the stochastic volatility h_t^W and the variance Q^W, this step involves a TVP regression with a

known error variance:

$$\begin{aligned} F_t^W &= \alpha_t^W + \sum_{p=1}^{P} \rho_{p,t}^W F_{t-p}^W + \left(h_t^W\right)^{\frac{1}{2}} \varepsilon_t^W \\ \Phi_t^W &= vec\left(\left[\alpha_t^W, \rho_{1,t}^W, .., \rho_{P,t}^W\right]\right) \\ \Phi_t^W &= \Phi_{t-1}^W + \left(Q^W\right)^{1/2} \eta_t^W \end{aligned}$$

This is a linear state-space model and the Carter Kohn algorithm is used to draw Φ_t^W. As described in Chapter 3, this involves running the Kalman filter and a backward recursion. As the variance of the error to the observation equation is time-varying (i.e. h_t^W), a slight modification to the Kalman filter is required to ensure that this time-variation is taken into account – i.e. the different value for the variance is selected in each recursion of the Kalman filter. The code for this step is shown in figure 4 lines 9 to 16. Line 19 creates the left and the right hand size variables in this TVP regression. Line 10 draws from $H\left(\Phi_t^W|\Xi\right)$ given previous values of F_t^W, Q^W, h_t^W and the initial conditions $\Phi_{0|0}^W, P_{\Phi^W}$. The function carterKohnAR has the following inputs: 1) dependent variable, 2) independent variable, 3) Q^W, 4) h_t^W, 5) $\Phi_{0|0}^W$, 6) P_{Φ^W}, 7) P, 8) *CHECK*, 9) *maxdraws*, 10) *EX*. If *CHECK*=1, then at most maxdraws attempts are made to find a stable draw. If no stable draw is found *problemw* is set to 1. Finally *EX*=1 implies there is one exogenous regressor, i.e. the intercept. The function returns the draw from conditional posterior using the Carter Kohn algorithm: *beta2w*, the residuals of the regression *errorw*, a dummy variable that equals 1 at a particular time period if the stability condition is violated at that observation (*rootsw*) and *problemw*.

(2) Sample from $H\left(Q^W|\Xi\right)$: This conditional posterior is $IW\left(\left(\Phi_t^W - \Phi_{t-1}^W\right)'\left(\Phi_t^W - \Phi_{t-1}^W\right) + Q_0^W, T + T_0\right)$. Lines 18 to 20 in figure 4 display the draw of this parameter.

(3) Sample from $H\left(\Phi_t^c|\Xi\right)$ for $c = 1, 2, ..., C$: This involves exactly the same calculation as step 1. The only change is that we need to conduct this draw using each country factor seperately. Lines 22 to 33 in figure 4 show the application of the Carter Kohn algorithm using each F_t^c.

(4) Sample from $H\left(Q^c|\Xi\right)$ for $c = 1, 2, ..., C$: The draw from this inverse Wishart conditional posterior is carried out on lines 35 to 38 exactly as in step 2.

(5) Sample from $H\left(\rho_{i,t}|\Xi\right)$ for $i = 1, 2, ..., N$: Given a value for the residuals e_{it}, the stochastic volatilties h_{it} and variances q_i this is imply an application of the Carter and Kohn algorithm to the TVP-AR model that applies to each e_{it}, i.e.

$$\begin{aligned} e_{it} &= \rho_{i,t} e_{it-1} + \left(h_{it}\right)^{\frac{1}{2}} \varepsilon_{it} \\ \rho_{i,t} &= \rho_{i,t-1} + \left(q_i^{1/2}\right) \eta_{it} \end{aligned}$$

The same code as in step 1 is again used (lines 41 to 52), looping over the N residuals. Not that *EX=0* when calling carterkohnAR as the regression has no intercept. Not the use of *parfor*, the parallel for loop which can speed up this step if N is large.

(6) Sample from $H\left(q_i|\Xi\right)$ for $i = 1, 2, ..., N$: Lines 54 to 58 show this draw from the inverse Wishart distribution.

(7) Sample from $H\left(h_t^W|\Xi\right)$: Given the residuals of the transition equation 2.2, the following stochastic volatility model applies:

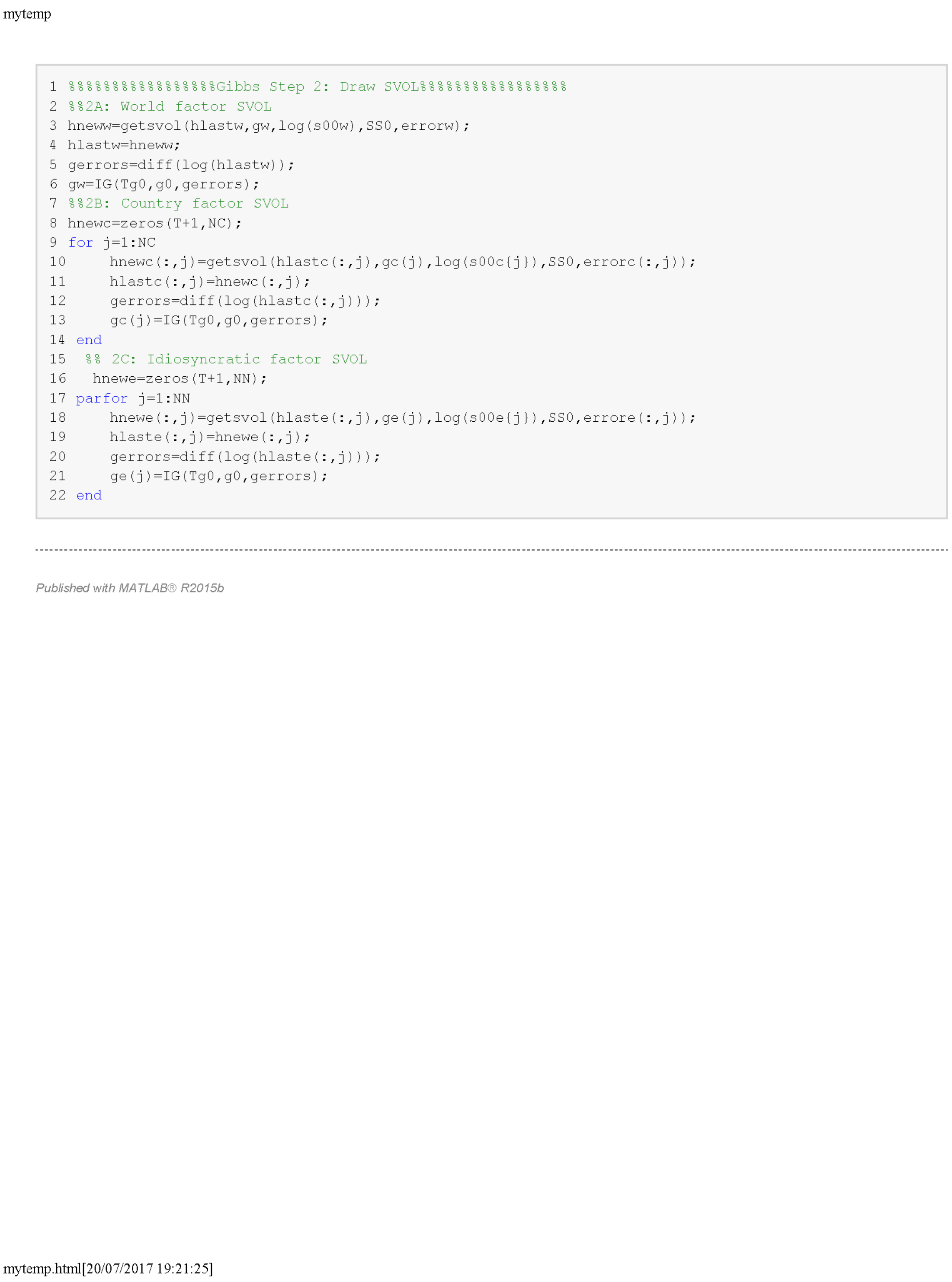
$$\begin{aligned} r_t^W &= \left(h_t^W\right)^{\frac{1}{2}} \varepsilon_t^W \\ \ln h_t^W &= \ln h_{t-1}^W + \left(g^W\right)^{1/2} u_t^W \end{aligned}$$

Given g^W and initial conditions, the independence Metropolis algorithm of Jacquier *et al.* (2004) can be used to draw h_t^W as explained in Chapter 5. The code for this step is shown in figure 5 (lines 3 and 4). Line 3 calls a function getsvol. The inputs to this function are: 1) *hlastw*, the last draw of h_t^W, 2) g^W, 3) the prior mean for the initial volatility $\bar{\mu}$, 4) The variance of the prior for the initial volatility and 5) the residuals r_t^W, i.e. the observed data in the observation equation of the stochastic volatility model. It returns a $(T + 1) \times 1$ vector containing the draw of h_t^W. Line 4 updates *hlastw* for the next draw.

(8) Sample from $H\left(g^W|\Xi\right)$: This conditional posterior is inverse Gamma: $IG(\left(\ln h_t^W - \ln h_{t-1}^W\right)'\left(\ln h_t^W - \ln h_{t-1}^W\right) + g_0, T + T_g)$. Lines 5 and 6 conduct this draw.

(9) Sample from $H\left(h_t^c|\Xi\right)$ for $c = 1, 2, ..., C$: As in step 7, this draw is a simple application of the Jacquier *et al.* (2004) using the residuals from the transition equation of each country factor. This is done on lines 9 to 11 in figure 5.

(10) Sample from $H\left(g^W|\Xi\right)$ for $c = 1, 2, ..., C$: This is simply a series of draws from the inverse Gamma distributions as in step 8. See lines 12 and 13 of the code.



mytemp

```
1 %%%%%%%%%%%%%%%%%Gibbs Step 2: Draw SVOL%%%%%%%%%%%%%%%%%
2 %%2A: World factor SVOL
3 hneww=getsvol(hlastw,gw,log(s00w),SS0,errorw);
4 hlastw=hneww;
5 gerrors=diff(log(hlastw));
6 gw=IG(Tg0,g0,gerrors);
7 %%2B: Country factor SVOL
8 hnewc=zeros(T+1,NC);
9 for j=1:NC
10     hnewc(:,j)=getsvol(hlastc(:,j),gc(j),log(s00c{j}),SS0,errorc(:,j));
11     hlastc(:,j)=hnewc(:,j);
12     gerrors=diff(log(hlastc(:,j)));
13     gc(j)=IG(Tg0,g0,gerrors);
14 end
15  %% 2C: Idiosyncratic factor SVOL
16   hnewe=zeros(T+1,NN);
17 parfor j=1:NN
18     hnewe(:,j)=getsvol(hlaste(:,j),ge(j),log(s00e{j}),SS0,errore(:,j));
19     hlaste(:,j)=hnewe(:,j);
20     gerrors=diff(log(hlaste(:,j)));
21     ge(j)=IG(Tg0,g0,gerrors);
22 end
```

Published with MATLAB® R2015b

mytemp.html[20/07/2017 19:21:25]

Figure 5. Drawing stochastic volatility

(11) Sample from $H\left(h_{it}|\Xi\right)$ for $i=1,2,...,N$: This requires nothing more than an application of the Jacquier *et al.* (2004) algorithm to the series of stochastic volatility models given by

$$\begin{aligned} r_{it} &= \left(h_{it}\right)^{\frac{1}{2}}\varepsilon_{it} \\ \ln h_{it} &= \ln h_{it-1} + \left(g_i\right)^{1/2}u_{it} \end{aligned}$$

where r_{it} denotes the residuals $r_{it}=e_{it}-\rho_{i,t}e_{it-1}$ for $i=1,2,...,N$ collected when conducting step 5 above. See lines 16 to 19 of the code.

(12) Sample from $H\left(g_i|\Xi\right)$ for $i=1,2,...,N$: Line 20 and 21 of figure 5 show the draw of this variance from the inverse Gamma distribution for each i.

(13) Sample from $H(\beta|\Xi)$: The observation equation of the factor model is

$$\pi_{it} = \beta_i^W F_t^W + \beta_i^c F_t^c + e_{it}$$

where the serially correlated, heteroscedastic error term is defined as

$$e_{it} = \rho_{i,t} e_{it-1} + (h_{it})^{\frac{1}{2}} \varepsilon_{it}$$

For each i, given $\rho_{i,t}$ and h_{it} the model can be easily transformed so that the error term has no serial correlation or heteroscedasticity:

$$\begin{aligned} \frac{\pi_{it} - \rho_{i,t}\pi_{it-1}}{(h_{it})^{\frac{1}{2}}} &= \beta_i^W \frac{F_t^W - \rho_{i,t}F_{t-1}^W}{(h_{it})^{\frac{1}{2}}} + \beta_i^c \frac{F_t^c - \rho_{i,t}F_{t-1}^c}{(h_{it})^{\frac{1}{2}}} + \varepsilon_{it}, \\ & \varepsilon_{it} \tilde{} N(0,1) \end{aligned}$$

Letting $Y_t = \frac{\pi_{it} - \rho_{i,t}\pi_{it-1}}{(h_{it})^{\frac{1}{2}}}$, $X_t = [\frac{F_t^W - \rho_{i,t}F_{t-1}^W}{(h_{it})^{\frac{1}{2}}}, \frac{F_t^c - \rho_{i,t}F_{t-1}^c}{(h_{it})^{\frac{1}{2}}}]$, the conditional posterior is normal $N(M, V)$:

$$\begin{aligned} M^* &= \left(V_\beta^{-1} + X_t' X_t\right)^{-1} \left(V_\beta^{-1}\hat{\beta}_i + X_t' Y_t\right) \\ V^* &= \left(V_\beta^{-1} + X_t' X_t\right)^{-1} \end{aligned} \tag{3.1}$$

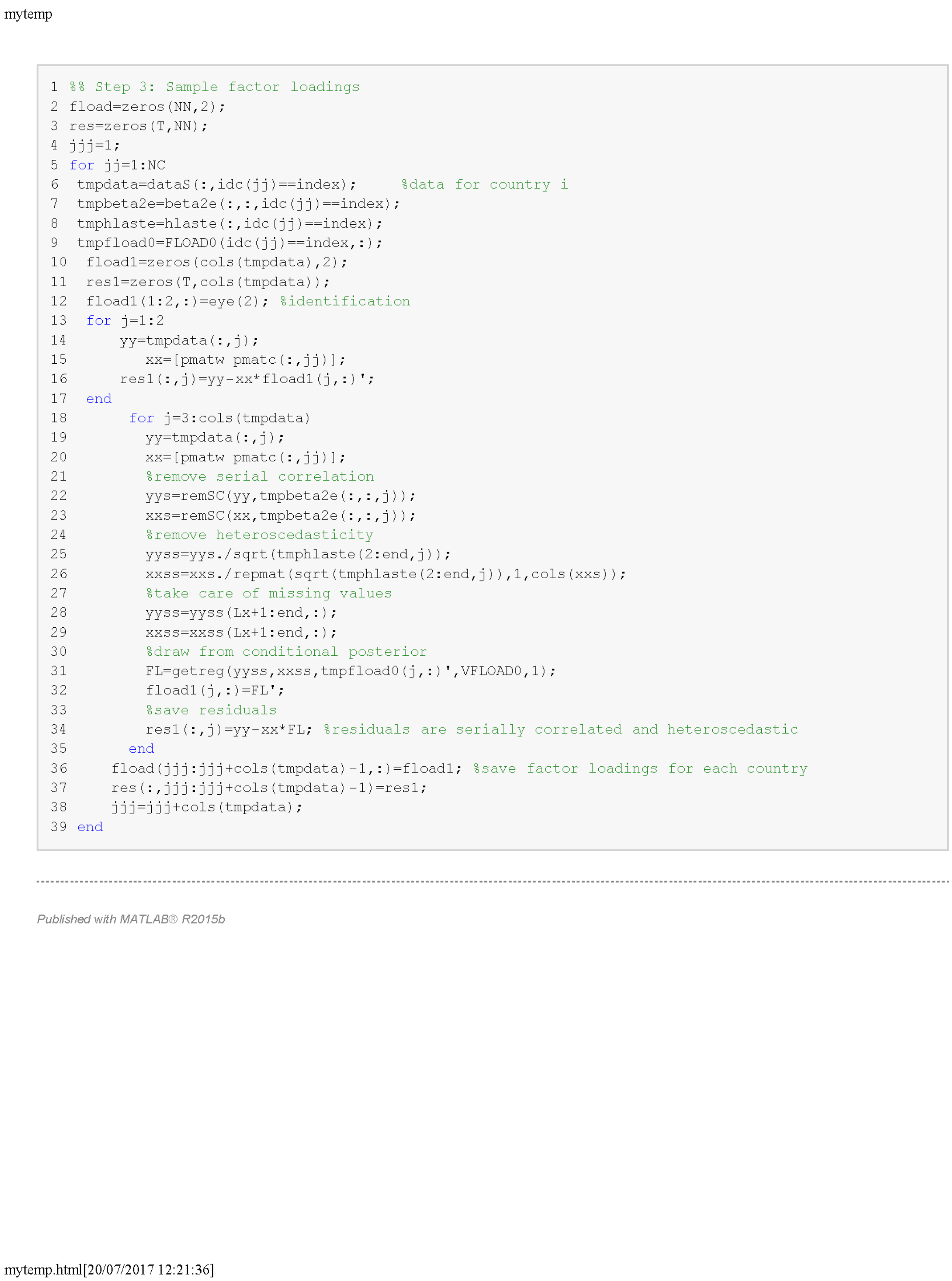
The code for this step is shown in figure 6. Line 5 starts a loop across the countries. Lines 6 to 9, select the data, the sereial correlation coefficients ρ_{it}, the error variances h_{it} and the prior means $\hat{\beta}_i$ for each country. The first two sets of factor loadings for each country are fixed, i.e. the first two rows of the factor loading matrix for each country is set equal to an identity matrix to deal with rotational indeterminancy of the factor model. Line 18 loops over the remaining data series. Lines 19 to 26 remove the serial correlation and heteroscedasticity and line 31 draws the laodings from the conditional posterior. The inputs to the function getreg are: 1)Y_t, 2)X_t, 3) $\hat{\beta}$ 4) V_β and 5) Variance of the error of the regression which equals 1 here. Note that the residuals e_{it} are updated on line 34.

(14) Sample from $H(F_t|\Xi)$: The final step is a draw of the factors from their conditional posterior. This is simply an application of the Carter and Kohn algorithm. However, it is instructive to consider the model in state-space form. Consider, for a simplicity an example where $N = 8$ and $C = 2$ and $P = 2$. The observation equation is then given by:

$$\underbrace{\begin{pmatrix} X_{1t} - \rho_{1t}X_{1t-1} \\ X_{2t} - \rho_{2t}X_{2t-1} \\ X_{3t} - \rho_{3t}X_{3t-1} \\ X_{4t} - \rho_{4t}X_{4t-1} \\ X_{5t} - \rho_{5t}X_{5t-1} \\ X_{6t} - \rho_{6t}X_{6t-1} \\ X_{7t} - \rho_{7t}X_{7t-1} \\ X_{8t} - \rho_{8t}X_{8t-1} \end{pmatrix}}_{y_t} = \underbrace{\begin{pmatrix} \beta_1^W & \beta_1^1 & 0 & -\rho_{1t}\beta_1^W & -\rho_{1t}\beta_1^1 & 0 \\ \beta_2^W & \beta_2^1 & 0 & -\rho_{2t}\beta_2^W & -\rho_{2t}\beta_2^1 & 0 \\ \beta_3^W & \beta_3^1 & 0 & -\rho_{3t}\beta_3^W & -\rho_{3t}\beta_3^1 & 0 \\ \beta_4^W & \beta_4^1 & 0 & -\rho_{4t}\beta_4^W & -\rho_{4t}\beta_4^1 & 0 \\ \beta_5^W & 0 & \beta_1^2 & -\rho_{5t}\beta_5^W & 0 & -\rho_{5t}\beta_1^2 \\ \beta_6^W & 0 & \beta_2^2 & -\rho_{6t}\beta_6^W & 0 & -\rho_{6t}\beta_2^2 \\ \beta_7^W & 0 & \beta_3^2 & -\rho_{7t}\beta_7^W & 0 & -\rho_{7t}\beta_3^2 \\ \beta_8^W & 0 & \beta_4^2 & -\rho_{8t}\beta_8^W & 0 & -\rho_{8t}\beta_4^2 \end{pmatrix}}_{H_t} \underbrace{\begin{pmatrix} F_t^W \\ F_t^1 \\ F_t^2 \\ F_{t-1}^W \\ F_{t-1}^1 \\ F_{t-1}^2 \end{pmatrix}}_{\beta_t} + \underbrace{\begin{pmatrix} \tilde{e}_{1t} \\ \tilde{e}_{2t} \\ \tilde{e}_{3t} \\ \tilde{e}_{4t} \\ \tilde{e}_{5t} \\ \tilde{e}_{6t} \\ \tilde{e}_{7t} \\ \tilde{e}_{8t} \end{pmatrix}}_{\tilde{e}_t}$$

$$var(\tilde{e}_t) = R_t = \begin{pmatrix} h_{1t} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_{2t} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_{3t} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_{4t} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_{5t} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_{6t} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h_{7t} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h_{8t} \end{pmatrix}$$

The observation equation is defined so that the residuals are free from serial correlation. For example, the first line reads $X_{1t} - \rho_{1t}X_{1t-1} = \beta_1^W \left(F_t^W - \rho_{1,t}F_{t-1}^W\right) + \beta_1^1 \left(F_t^1 - \rho_{1,t}F_{t-1}^1\right) + \tilde{e}_{1t}$ where $\tilde{e}_{1t}$ is serially uncorrelated but heteroscedastic. Notice that the matrices H_t, R_t change over time. We need to account



mytemp

```
%% Step 3: Sample factor loadings
fload=zeros(NN,2);
res=zeros(T,NN);
jjj=1;
for jj=1:NC
 tmpdata=dataS(:,idc(jj)==index);     %data for country i
 tmpbeta2e=beta2e(:,:,idc(jj)==index);
 tmphlaste=hlaste(:,idc(jj)==index);
 tmpfload0=FLOAD0(idc(jj)==index,:);
 fload1=zeros(cols(tmpdata),2);
 res1=zeros(T,cols(tmpdata));
 fload1(1:2,:)=eye(2); %identification
 for j=1:2
     yy=tmpdata(:,j);
        xx=[pmatw pmatc(:,jj)];
     res1(:,j)=yy-xx*fload1(j,:)';
 end
      for j=3:cols(tmpdata)
        yy=tmpdata(:,j);
        xx=[pmatw pmatc(:,jj)];
        %remove serial correlation
        yys=remSC(yy,tmpbeta2e(:,:,j));
        xxs=remSC(xx,tmpbeta2e(:,:,j));
        %remove heteroscedasticity
        yyss=yys./sqrt(tmphlaste(2:end,j));
        xxss=xxs./repmat(sqrt(tmphlaste(2:end,j)),1,cols(xxs));
        %take care of missing values
        yyss=yyss(Lx+1:end,:);
        xxss=xxss(Lx+1:end,:);
        %draw from conditional posterior
        FL=getreg(yyss,xxss,tmpfload0(j,:)',VFLOAD0,1);
        fload1(j,:)=FL';
        %save residuals
        res1(:,j)=yy-xx*FL; %residuals are serially correlated and heteroscedastic
      end
    fload(jjj:jjj+cols(tmpdata)-1,:)=fload1; %save factor loadings for each country
    res(:,jjj:jjj+cols(tmpdata)-1)=res1;
    jjj=jjj+cols(tmpdata);
end
```

Published with MATLAB® R2015b

mytemp.html[20/07/2017 12:21:36]

FIGURE 6. Draw factor loadings

for this in the Kalman filter. The transition equation of the model is defined as:

$$\underbrace{\begin{pmatrix} F_t^W \\ F_t^1 \\ F_t^2 \\ F_{t-1}^W \\ F_{t-1}^1 \\ F_{t-1}^2 \end{pmatrix}}_{\beta_t} = \underbrace{\begin{pmatrix} \alpha_t^W \\ \alpha_t^1 \\ \alpha_t^2 \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\mu_t} \underbrace{\begin{pmatrix} \rho_{1,t}^W & 0 & 0 & \rho_{2,t}^W & 0 & 0 \\ 0 & \rho_{1,t}^1 & 0 & 0 & \rho_{2,t}^1 & 0 \\ 0 & 0 & \rho_{1,t}^2 & 0 & 0 & \rho_{2,t}^2 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}}_{F_t} \underbrace{\begin{pmatrix} F_{t-1}^W \\ F_{t-1}^1 \\ F_{t-1}^2 \\ F_{t-2}^W \\ F_{t-2}^1 \\ F_{t-2}^2 \end{pmatrix}}_{\beta_{t-1}} + \underbrace{\begin{pmatrix} \tilde{v}_{1t} \\ \tilde{v}_{2t} \\ \tilde{v}_{3t} \\ 0 \\ 0 \\ 0 \end{pmatrix}}_{\tilde{v}_t}$$

$$var\left(\tilde{v}_t\right) = Q_t = \begin{pmatrix} h_t^W & 0 & 0 & 0 & 0 & 0 \\ 0 & h_t^1 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_t^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Again, the parameters of the transition equation are time-varying. This needs to be accounted for in the Kalman filter and the backward recursion. The code for this step is shown in figures 7 and 8. Lines 2 to 5 calculate $X_{it} - \rho_{it} X_{it-1}$ using the function remSC which takes arguments X_{it} and ρ_{it}. The Kalman filter iteration begins on line 13. Notice that the matrices of the state-space have to be built within the loop as they are time-varying. Lines 18 to 36 construct the matrix H_t shown in the observation equation. Line 38 constructs R_t. The matrices of the transition equation are constructed on lines 42 to 47. Lines 50 to 53 are the equations of the prediction and the update steps of the Kalman filter. The backward recursion of the Carter and Kohn algorithm begins on line 66. As in the Kalman filter, the matrices of the transition equation are constructed at each time period. One important point is the fact that the backward recursion is derived by assuming the following 'observation equation' (see Chapter 3):

$$\begin{aligned} \beta_{t+1} &= \mu_{t+1} + F_{t+1}\beta_t + v_{t+1}, \\ var(v_{t+1}) &= Q_{t+1} \end{aligned}$$

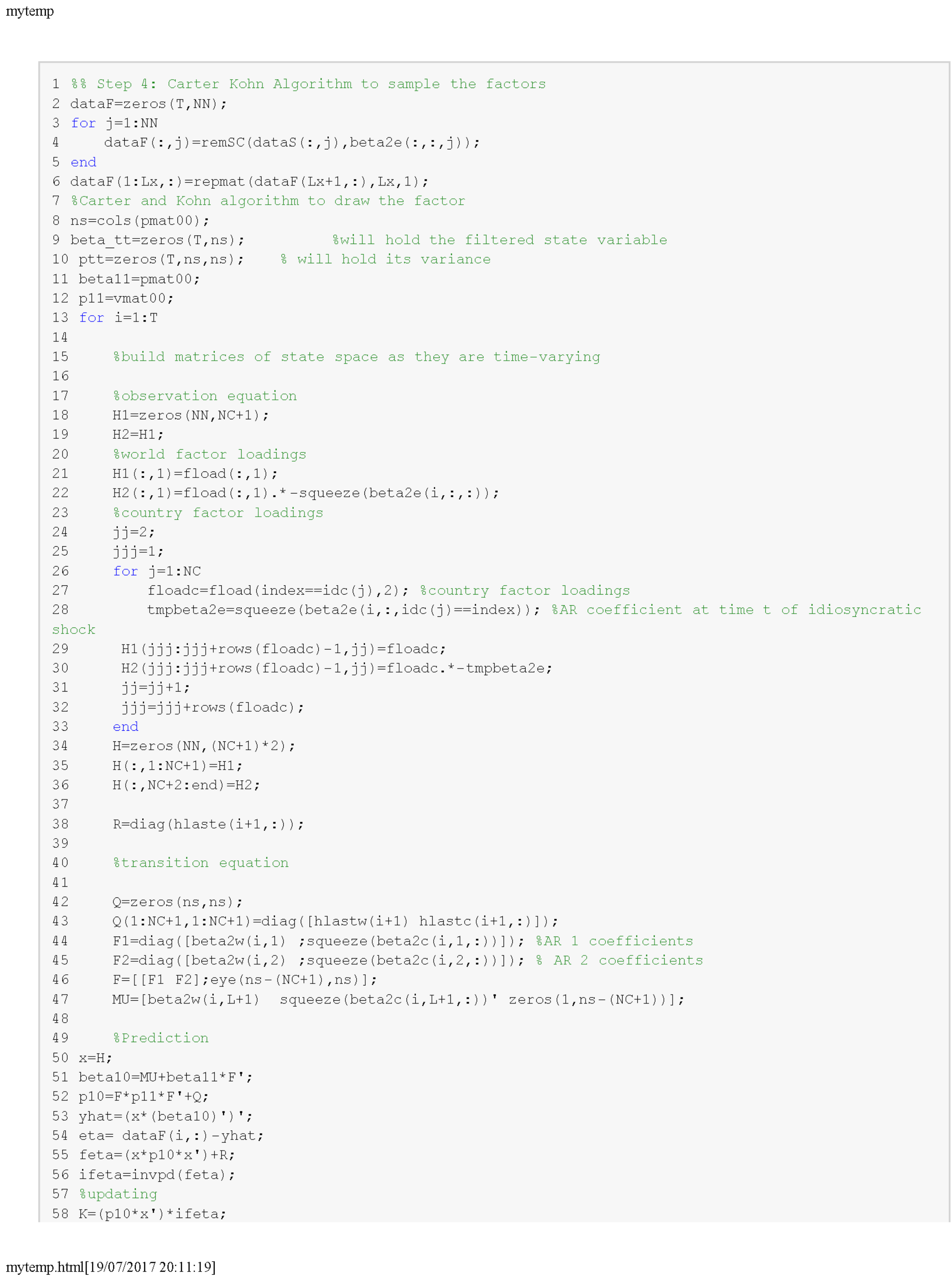
Notice that the matrices of the state-space are dated at time $t+1$. Thus, in the Carter and Kohn backward recusrsion when $i = T-2$, for example, the matrices of the state space are $\mu_{T-1}, F_{T-1}, Q_{T-1}$. These matrices are constructed on lines 77 to 82. Line 84 to 86, select the rows corresponding to the non-singular block of Q. The remaining lines calculate the mean and variance of the conditional distribution and draw the state-variables (see Chapter 3). The factors are extracted on lines 94 to 96.

As mentioned above, the example uses artificial data. Based on a run using 20,000 iterations and a burn-in of 10,000 replications we can compare the estimated contribution of the world factor to the variance of each series with the true value assumed in the data generating process.

Figure 9 shows this comparison for the first 40 series of artificially generated data. The black line shown the true time-varying contribution of the world factor to the variance of the series. The red line shows the posterior estimate obtained by running the algorithm described above.

4. Further reading

- Time-varying factor models are also estimated in Bianchi *et al.* (2009), Baumeister *et al.* (2013), Liu *et al.* (2014), Ellis *et al.* (2014) and Mumtaz and Theodoridis (2017). The algorithms used in these papers is very similar to the one described in this chapter.
- Kim and Nelson (1999) consider state-space models with Markov Switching in Chapter 10.

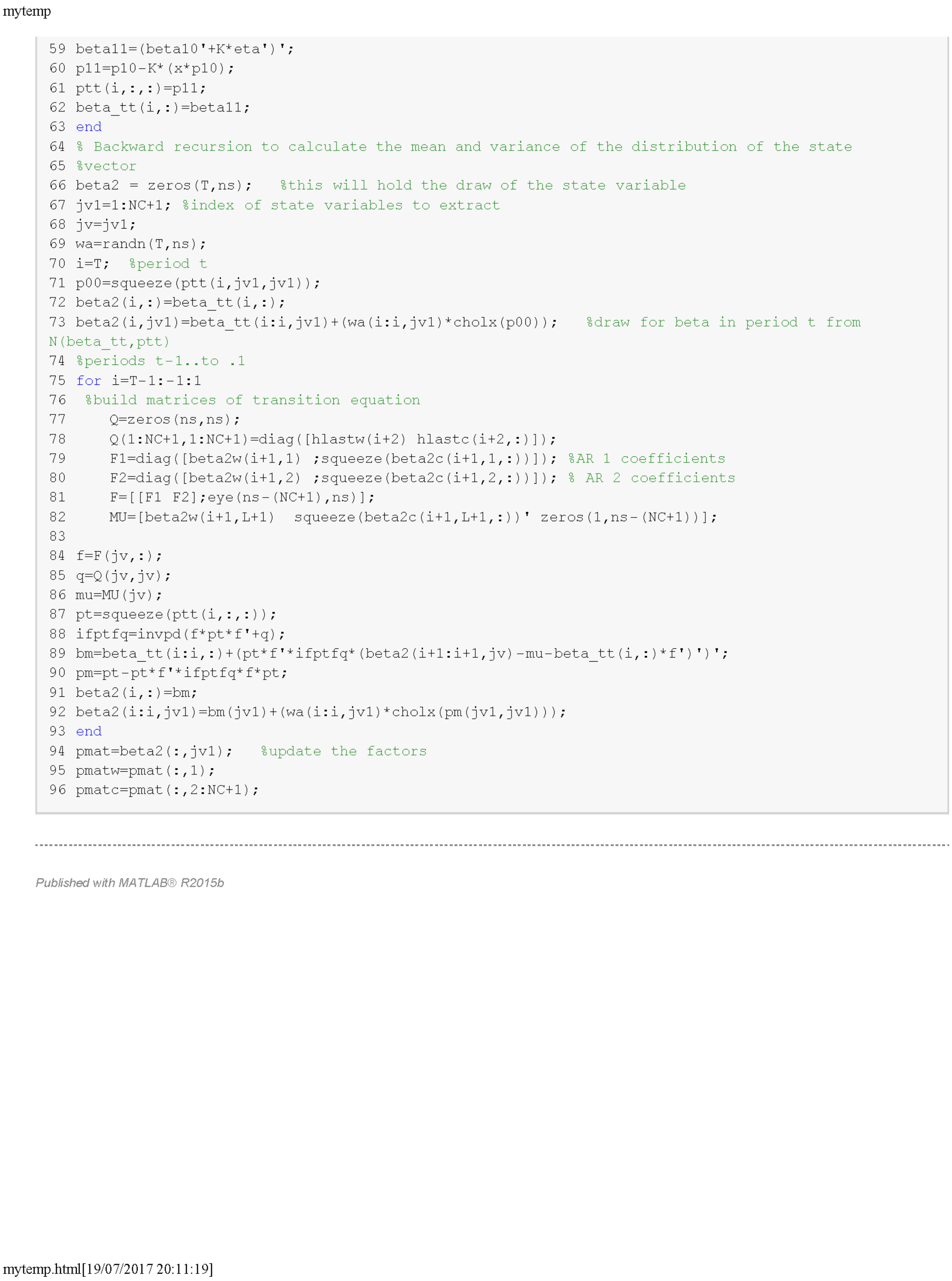


mytemp

```
%% Step 4: Carter Kohn Algorithm to sample the factors
dataF=zeros(T,NN);
for j=1:NN
    dataF(:,j)=remSC(dataS(:,j),beta2e(:,:,j));
end
dataF(1:Lx,:)=repmat(dataF(Lx+1,:),Lx,1);
%Carter and Kohn algorithm to draw the factor
ns=cols(pmat00);
beta_tt=zeros(T,ns);          %will hold the filtered state variable
ptt=zeros(T,ns,ns);    % will hold its variance
beta11=pmat00;
p11=vmat00;
for i=1:T

    %build matrices of state space as they are time-varying

    %observation equation
    H1=zeros(NN,NC+1);
    H2=H1;
    %world factor loadings
    H1(:,1)=fload(:,1);
    H2(:,1)=fload(:,1).*-squeeze(beta2e(i,:,:));
    %country factor loadings
    jj=2;
    jjj=1;
    for j=1:NC
        floadc=fload(index==idc(j),2); %country factor loadings
        tmpbeta2e=squeeze(beta2e(i,:,idc(j)==index)); %AR coefficient at time t of idiosyncratic shock
     H1(jjj:jjj+rows(floadc)-1,jj)=floadc;
     H2(jjj:jjj+rows(floadc)-1,jj)=floadc.*-tmpbeta2e;
     jj=jj+1;
     jjj=jjj+rows(floadc);
    end
    H=zeros(NN,(NC+1)*2);
    H(:,1:NC+1)=H1;
    H(:,NC+2:end)=H2;

    R=diag(hlaste(i+1,:));

    %transition equation

    Q=zeros(ns,ns);
    Q(1:NC+1,1:NC+1)=diag([hlastw(i+1) hlastc(i+1,:)]);
    F1=diag([beta2w(i,1) ;squeeze(beta2c(i,1,:))]); %AR 1 coefficients
    F2=diag([beta2w(i,2) ;squeeze(beta2c(i,2,:))]); % AR 2 coefficients
    F=[[F1 F2];eye(ns-(NC+1),ns)];
    MU=[beta2w(i,L+1)  squeeze(beta2c(i,L+1,:))' zeros(1,ns-(NC+1))];

    %Prediction
x=H;
beta10=MU+beta11*F';
p10=F*p11*F'+Q;
yhat=(x*(beta10)')';
eta= dataF(i,:)-yhat;
feta=(x*p10*x')+R;
ifeta=invpd(feta);
%updating
K=(p10*x')*ifeta;
```

mytemp.html[19/07/2017 20:11:19]

FIGURE 7. Carter Kohn Step to draw factors.



mytemp

```
beta11=(beta10'+K*eta')';
p11=p10-K*(x*p10);
ptt(i,:,:)=p11;
beta_tt(i,:)=beta11;
end
% Backward recursion to calculate the mean and variance of the distribution of the state
%vector
beta2 = zeros(T,ns);   %this will hold the draw of the state variable
jv1=1:NC+1; %index of state variables to extract
jv=jv1;
wa=randn(T,ns);
i=T;  %period t
p00=squeeze(ptt(i,jv1,jv1));
beta2(i,:)=beta_tt(i,:);
beta2(i,jv1)=beta_tt(i:i,jv1)+(wa(i:i,jv1)*cholx(p00));   %draw for beta in period t from N(beta_tt,ptt)
%periods t-1..to .1
for i=T-1:-1:1
 %build matrices of transition equation
    Q=zeros(ns,ns);
    Q(1:NC+1,1:NC+1)=diag([hlastw(i+2) hlastc(i+2,:)]);
    F1=diag([beta2w(i+1,1) ;squeeze(beta2c(i+1,1,:))]); %AR 1 coefficients
    F2=diag([beta2w(i+1,2) ;squeeze(beta2c(i+1,2,:))]); % AR 2 coefficients
    F=[[F1 F2];eye(ns-(NC+1),ns)];
    MU=[beta2w(i+1,L+1)  squeeze(beta2c(i+1,L+1,:))' zeros(1,ns-(NC+1))];

f=F(jv,:);
q=Q(jv,jv);
mu=MU(jv);
pt=squeeze(ptt(i,:,:));
ifptfq=invpd(f*pt*f'+q);
bm=beta_tt(i:i,:)+(pt*f'*ifptfq*(beta2(i+1:i+1,jv)-mu-beta_tt(i,:)*f')')';
pm=pt-pt*f'*ifptfq*f*pt;
beta2(i,:)=bm;
beta2(i:i,jv1)=bm(jv1)+(wa(i:i,jv1)*cholx(pm(jv1,jv1)));
end
pmat=beta2(:,jv1);   %update the factors
pmatw=pmat(:,1);
pmatc=pmat(:,2:NC+1);
```

Published with MATLAB® R2015b

mytemp.html[19/07/2017 20:11:19]

Figure 8. Carter Kohn Step to draw factors.

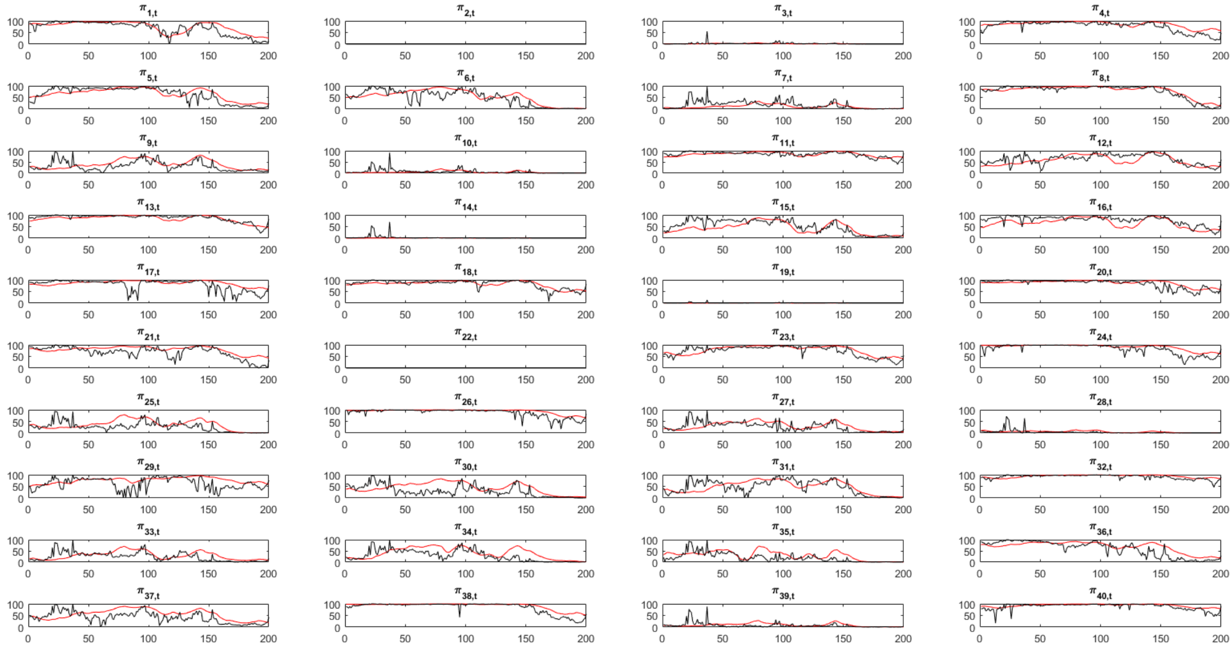


Figure 9. A comparison of the true contribution of the world factor (black line) with the estimated posterior median contribution (red line)

CHAPTER 8

Appendix: Introduction to Matlab©

1. Introduction

This appendix provides a basic introduction to Matlab and introduces the key concept needed in dealing with the codes used in this book. Note that a number of alternative guides to Matlab are available on the web and these may be used to supplement the material here.

2. Getting started

Figure 1 shows a basic screen shot of Matlab. There are two main windows: (1) the editor window which is docked on top and the (2) command window which is docked at the bottom. The editor is where we type our code. Once the code is typed it can be saved as a file which has the extension .m. The code is run by clicking on the green run button or by simply typing in the name of the program file in the command window. The command window is where the output from running the code appears. Or alternatively, each line of the code can be run by typing it in the command window and pressing enter.

In figure 2 we show how to create a generic first program called helloworld.m which prints out the words Hello World. The code simply consists of the line 'Hello World' where the single quotes signify that this a string variable as opposed to a numeric variable (or a number). By clicking on run, the output appears in the command window. Alternatively one can run the line of the program containing the code by highlighting the line and pressing F9.

3. Matrix programming language

The key data type in Matlab is a matrix and Matlab supports numerous Matrix operations (multiplication, transposes etc).

3.1. Creating matrices.

3.1.1. *Entering a matrix manually.* Suppose we want to create the following two matrices

$$X = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

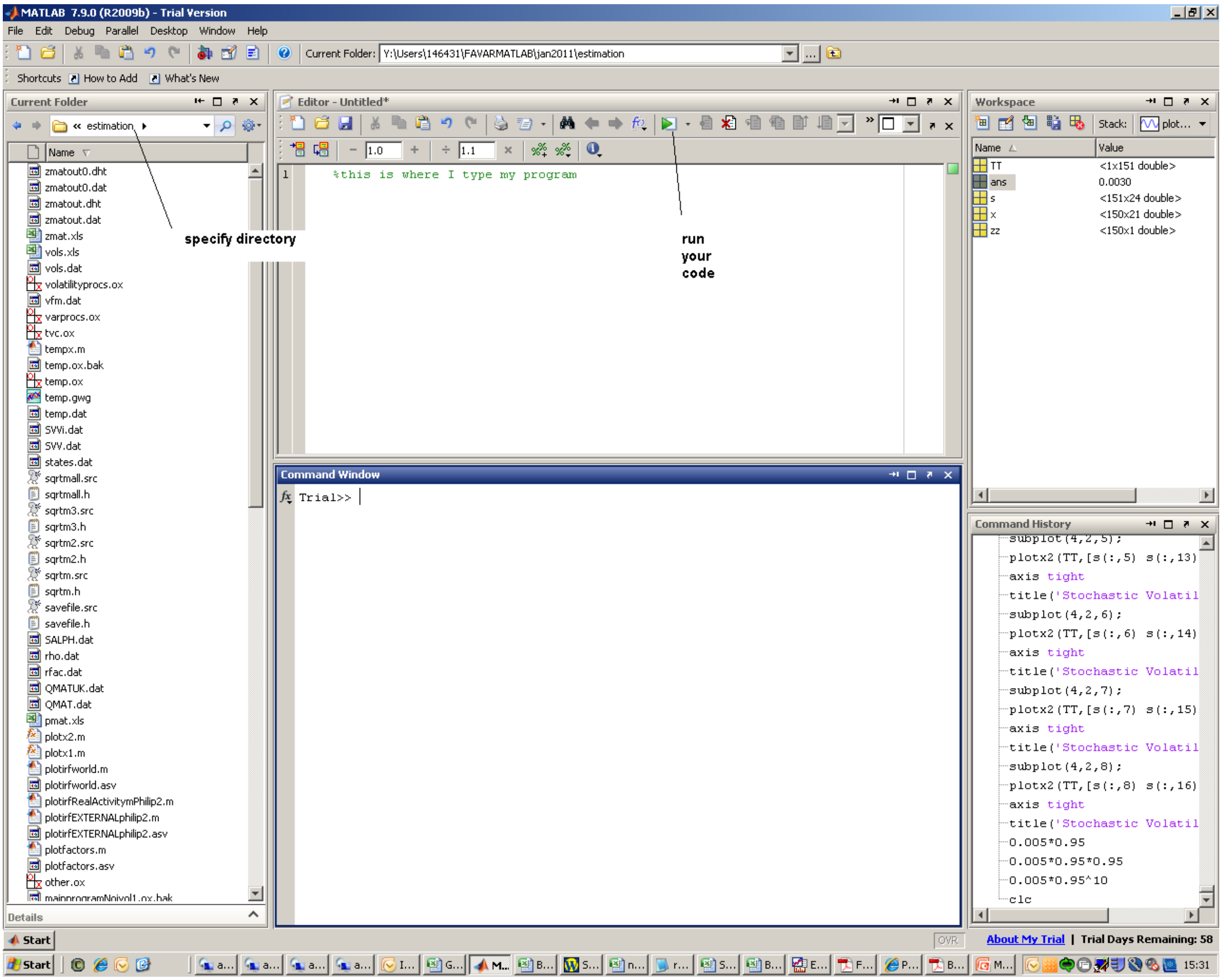
$$Z = \begin{pmatrix} 2 & 3 & 4 & 5 \\ 7 & 8 & 9 & 10 \end{pmatrix}$$

Figure 3 shows the matlab code required to do this (example1.m). Note that the numbers in the first column are line numbers which are shown here for convenience and will not appear in the actual code. Note also that any code starting with % is a comment and is ignored by matlab. Line 2 shows how X is created.[1] The numbers have square brackets around them. Each column of the matrix is seperated by a space and rows by a semi-colon. Lines 2 and 3 finish with a semi-colon. This stops Matlab from priniting out X and Z when the code is run. Instead we print the matrices by typing X and Z without a semi-colon on lines 5 and 6.

3.1.2. *Entering a matrix using built in commands.* Line 2 in figure 4 creates a 10×10 matrix of zeros (the file is example2.m). The first argument in the function zeros denotes the number of rows, the second denotes the number of columns required. Line 4 creates a 10×20 matrix of ones. Line 6 creates a 10×10 identity matrix. Line 8 creates a 10×10 matrix where each element is drawn from a N(0,1) distribution. Similarly, line 9 creates a matrix from the standard uniform distribution.

3.1.3. *Reading in data from an excel file.* In practice we will read in matrices (i.e. data) from an outside source. As an example we have stored data on UK GDP growth and inflation in an excel file called data.xls in the folder called data. Figure 5 shows the matlab code used to read the excel file. The command xlsread reads the data into a matrix called data. The variable names are read into a string variable called names. Note that text files can be read by using the command load. Suppose one wants to read in data from a text file dalled data.txt, one simply needs to type load data.txt. Type 'help load' in the command window for further information.

[1] Matlab is case sensitive. Therefore X is treated as a different variable than x.



```
1  %this is where I type my program
```

```
fx Trial>> |
```

Name	Value
TT	<1x151 double>
ans	0.0030
s	<151x24 double>
x	<150x21 double>
zz	<150x1 double>

```
subplot(4,2,5);
plotx2(TT,[s(:,5) s(:,13)
axis tight
title('Stochastic Volatil
subplot(4,2,6);
plotx2(TT,[s(:,6) s(:,14)
axis tight
title('Stochastic Volatil
subplot(4,2,7);
plotx2(TT,[s(:,7) s(:,15)
axis tight
title('Stochastic Volatil
subplot(4,2,8);
plotx2(TT,[s(:,8) s(:,16)
axis tight
title('Stochastic Volatil
0.005*0.95
0.005*0.95*0.95
0.005*0.95^10
clc
```

FIGURE 1. Matlab command window

3.2. Manipulating matrices. Lines 2 and 3 in figure 6 create two matrices X and Z shown in example 1 above. Line 5 sets the element (2,1), i.e. the element on the second row first column to 30. Line 7 creates a new 4 × 4 matrix by vertically concatenating X and Z. This done by the command M=[X;Z] where the semi-colon denotes vertical concatenation. That is, it creates

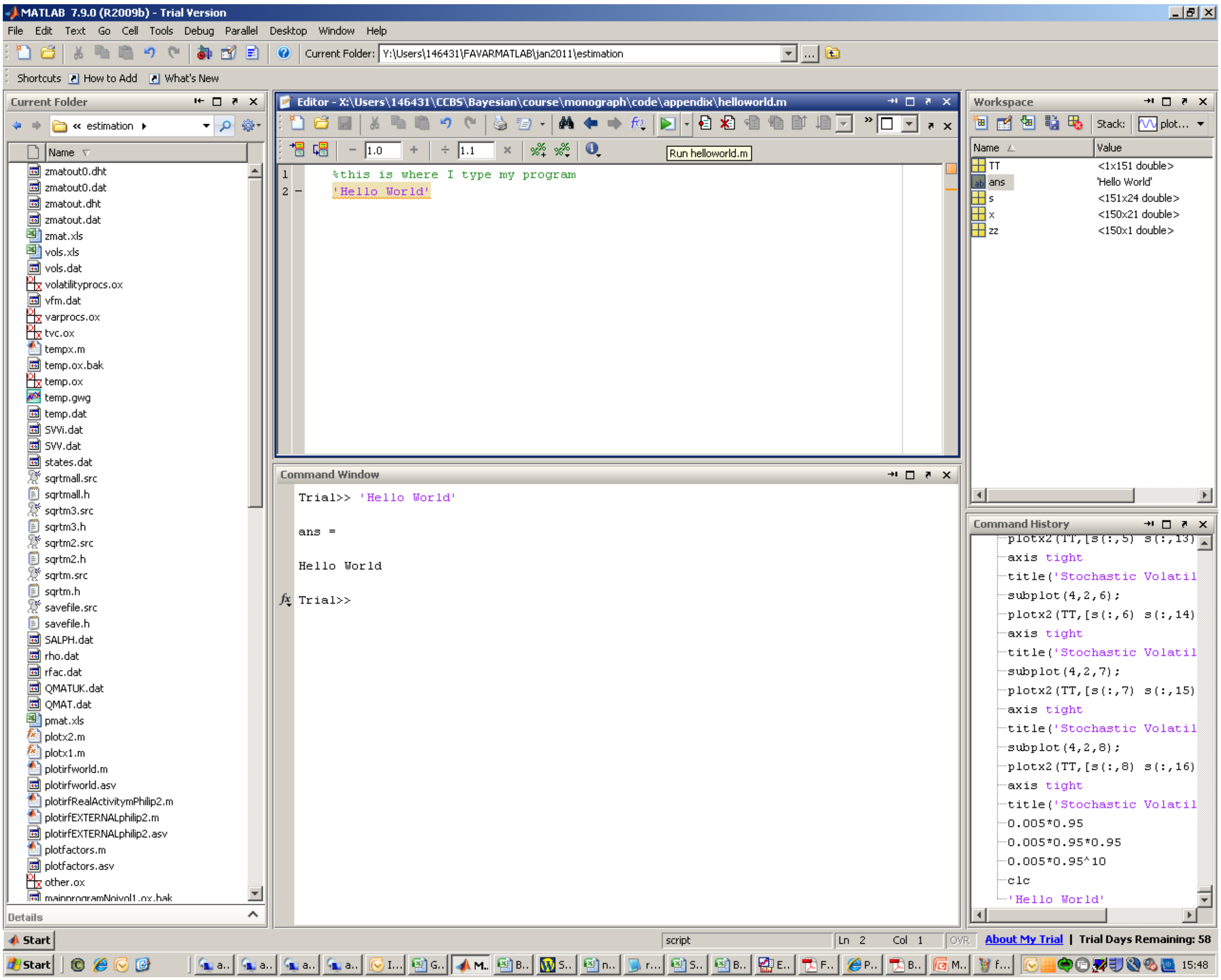
$$M=\begin{pmatrix}1&2&3&4\\30&6&7&8\\2&3&4&5\\7&8&9&10\end{pmatrix}$$

Line 9 of the code creates a new 2 × 8 matrix N by horizontally concatenating X and Z. This done by the command N=[X Z] where the space denotes horizontal concatenation. Finally Line 11 of the code shows how to set the entire second row of N equal to -10. Note that argument 1:end in N(2,1:end)=-10 selects columns 1 to 10. One can delete elements by setting them equal to [ ]. For example N(2,:)=[ ] deletes the entire second row.

3.3. Matrix Algebra. Matlab supports numerous matrix functions. We demonstrate some of these by writing code to estimate an OLS regression using data in the file data.xls used in example 3 above (example5.m). Recall the formulas for an OLS regression $Y = XB + E$ are

$$\begin{aligned}\hat{B} &= (X'X)^{-1}(X'Y)\\ VAR(E) &= S=\frac{(Y-XB)'(Y-XB)}{T-K}\\ VAR\left(\hat{B}\right) &= S\times(X'X)^{-1}\end{aligned}$$

Where T denotes the number of observations and K are the number of regressors. We assume Y is the first column of data.xls and X is the second column of data.xls and a constant term. Line 3 of the code shown in Figure 7 loads the data. Line 5 shows the function size to find the number of rows in the data. Line 7 assigns the first column of



MATLAB 7.9.0 (R2009b) - Trial Version
File Edit Text Go Cell Tools Debug Parallel Desktop Window Help
Current Folder: Y:\Users\146431\FAVARMATLAB\jan2011\estimation
Current Folder
Editor - X:\Users\146431\CCBS\Bayesian\course\monograph\code\appendix\helloworld.m
Run helloworld.m
Command Window
Workspace
Command History
Start
About My Trial | Trial Days Remaining: 58

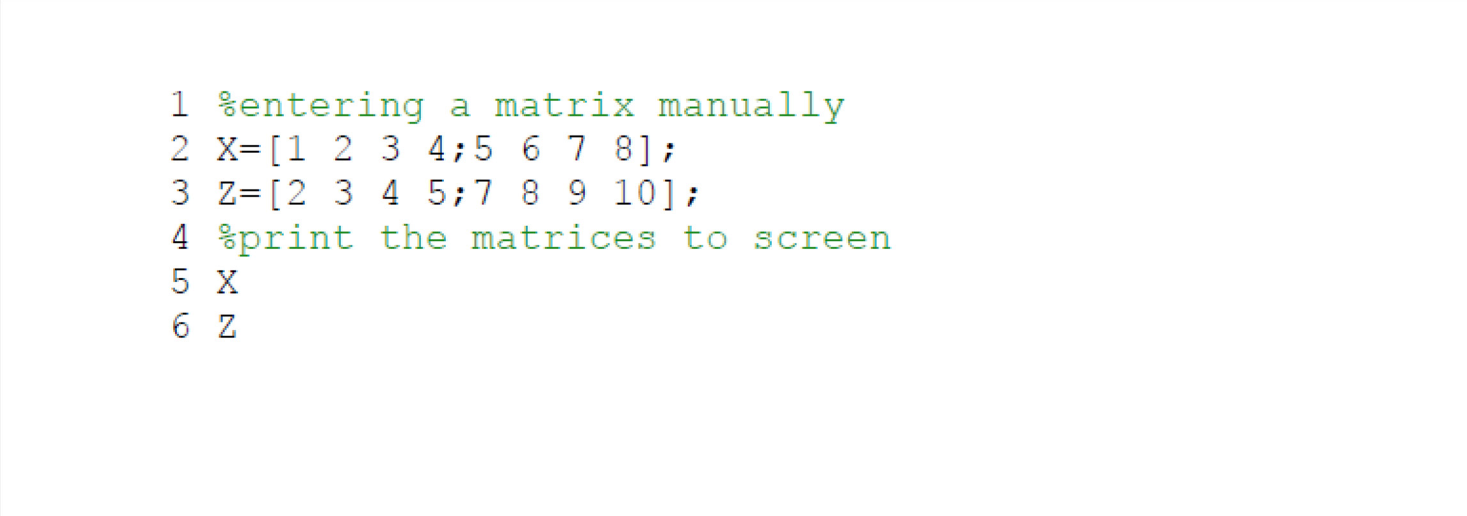
```
%this is where I type my program
'Hello World'
```

```
Trial>> 'Hello World'

ans =

Hello World

Trial>>
```

Name	Value
TT	<1x151 double>
ans	'Hello World'
s	<151x24 double>
x	<150x21 double>
zz	<150x1 double>

```
plotx2(TT,[s(:,5) s(:,13)
axis tight
title('Stochastic Volatil
subplot(4,2,6);
plotx2(TT,[s(:,6) s(:,14)
axis tight
title('Stochastic Volatil
subplot(4,2,7);
plotx2(TT,[s(:,7) s(:,15)
axis tight
title('Stochastic Volatil
subplot(4,2,8);
plotx2(TT,[s(:,8) s(:,16)
axis tight
title('Stochastic Volatil
0.005*0.95
0.005*0.95*0.95
0.005*0.95^10
clc
'Hello World'
```

Figure 2. Hello World

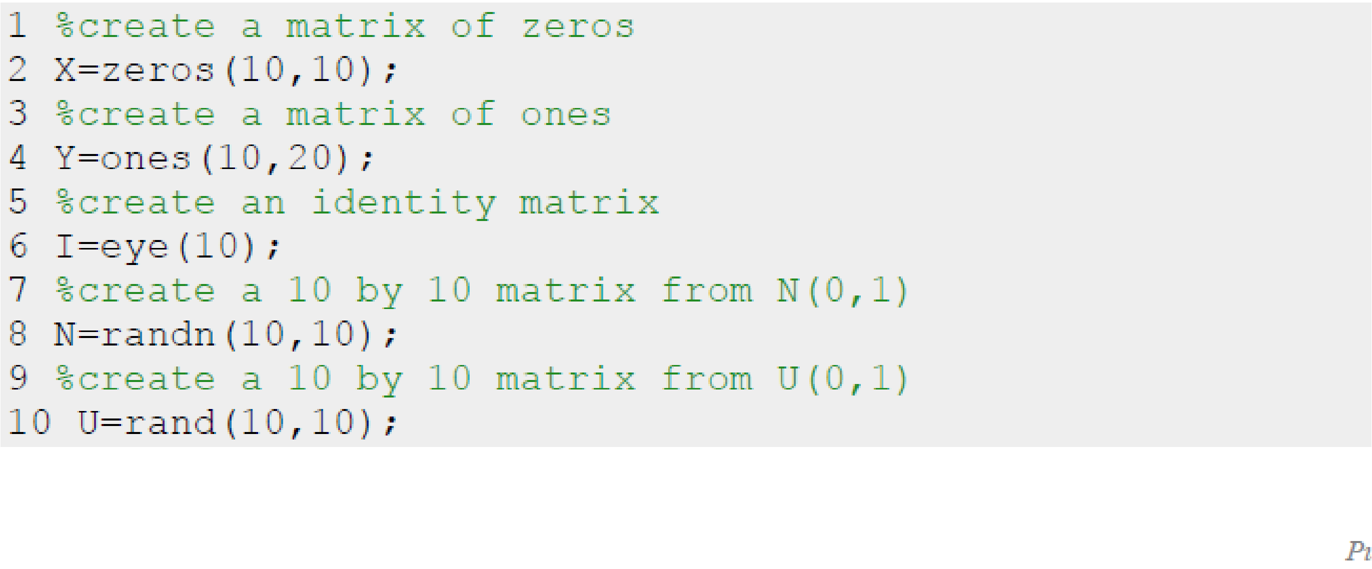


```
%entering a matrix manually
X=[1 2 3 4;5 6 7 8];
Z=[2 3 4 5;7 8 9 10];
%print the matrices to screen
X
Z
```

Figure 3. Example 1

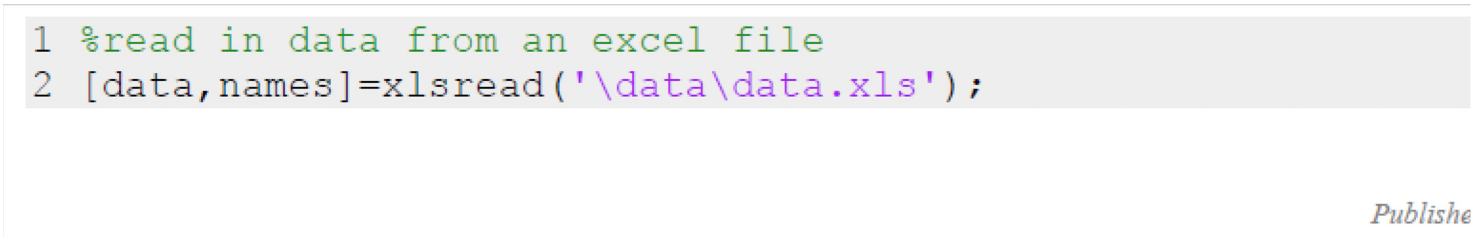
data to a $T \times 1$ matrix called Y. Line 9 creates the $T \times 2$ X matrix where the first column is the constant and the second column is the second column of data.

Line 11 calculates the OLS esimate of the coefficients using the formula $\hat{B} = (X'X)^{-1}(X'Y)$. This line shows three matrix functions. First the the transpose of X in matlab is simply X'. One can multiply conformable matrices by using the * key. The inverse of matrix is calculated in matlab using the inv function. Line 13 calculates the



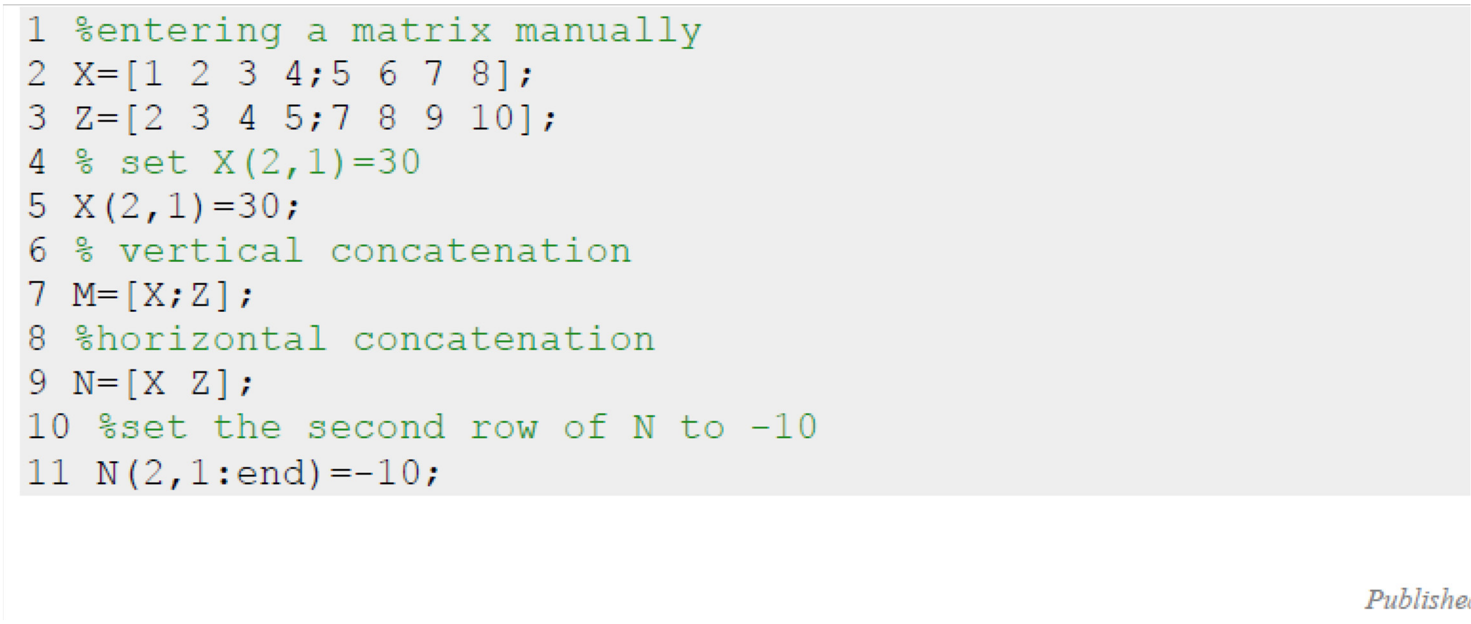
```
%create a matrix of zeros
X=zeros(10,10);
%create a matrix of ones
Y=ones(10,20);
%create an identity matrix
I=eye(10);
%create a 10 by 10 matrix from N(0,1)
N=randn(10,10);
%create a 10 by 10 matrix from U(0,1)
U=rand(10,10);
```

FIGURE 4. Example 2



```
%read in data from an excel file
[data,names]=xlsread('\data\data.xls');
```

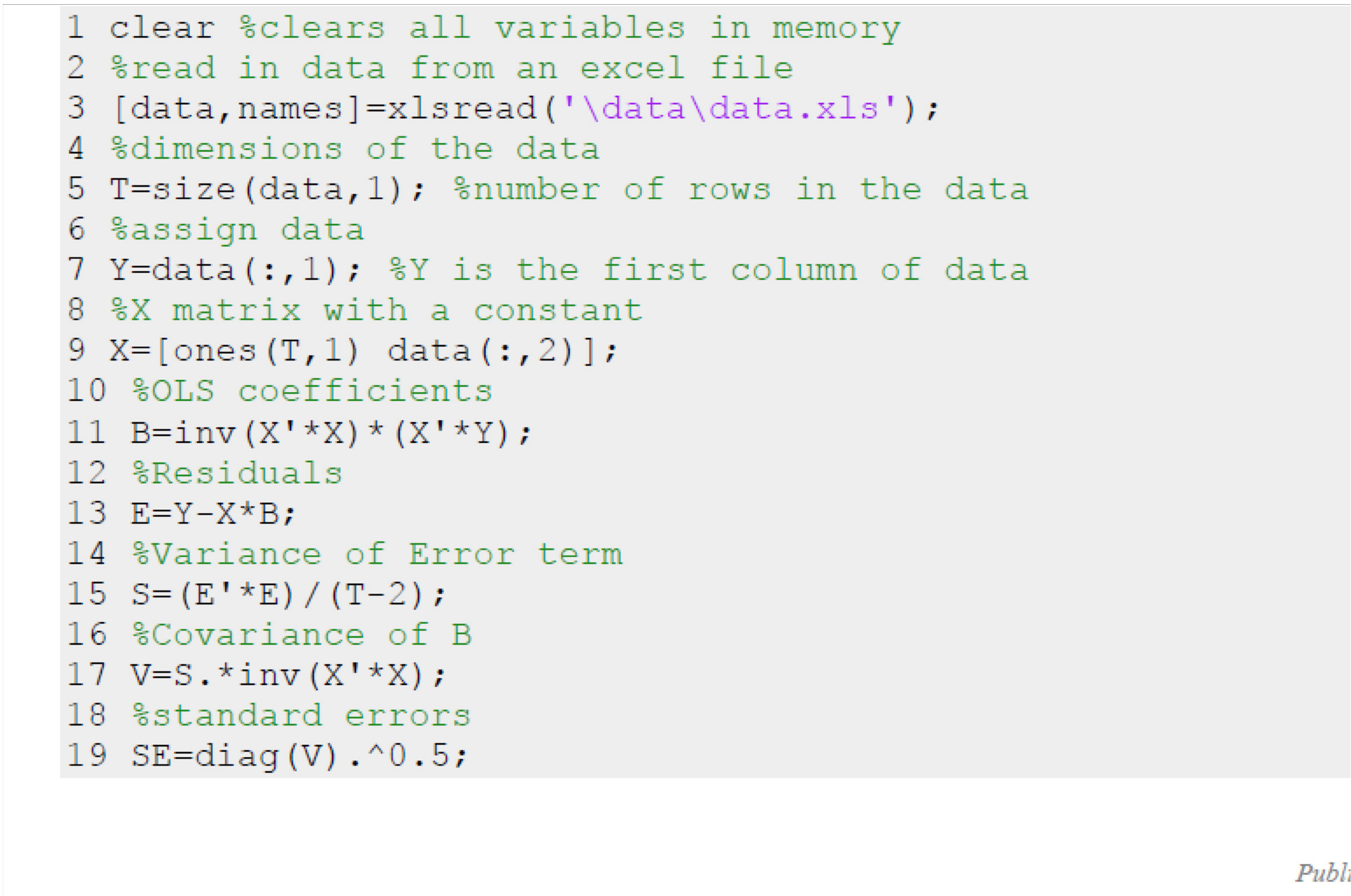
FIGURE 5. Example 3



```
%entering a matrix manually
X=[1 2 3 4;5 6 7 8];
Z=[2 3 4 5;7 8 9 10];
% set X(2,1)=30
X(2,1)=30;
% vertical concatenation
M=[X;Z];
%horizontal concatenation
N=[X Z];
%set the second row of N to -10
N(2,1:end)=-10;
```

FIGURE 6. Example 4

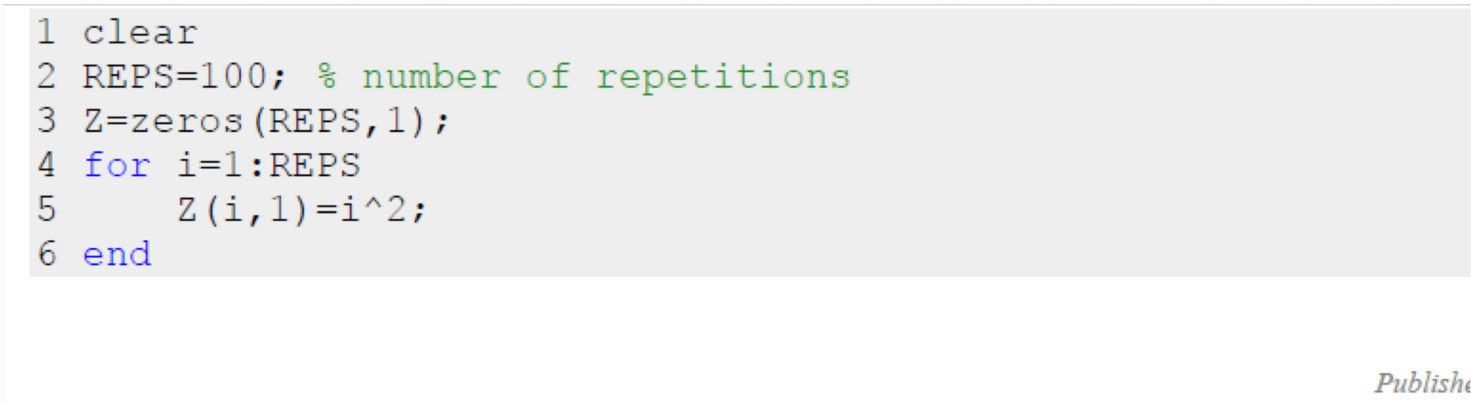
residuals while Line 15 calculates $VAR(E)$ and line 17 calculates $VAR\left(\hat{B}\right) = S \times (X'X)^{-1}$. Note that S is a scalar and $(X'X)^{-1}$ is a matrix. In general a scalar can be multiplied by each element of matrix by the '.*' key which denotes element by element multiplication. So this can also be used for element by element multiplication of two matrices. Finally, the standard errors of the coefficients are given as the square root of the diagonal of the matrix



```
1 clear %clears all variables in memory
2 %read in data from an excel file
3 [data,names]=xlsread('\data\data.xls');
4 %dimensions of the data
5 T=size(data,1); %number of rows in the data
6 %assign data
7 Y=data(:,1); %Y is the first column of data
8 %X matrix with a constant
9 X=[ones(T,1) data(:,2)];
10 %OLS coefficients
11 B=inv(X'*X)*(X'*Y);
12 %Residuals
13 E=Y-X*B;
14 %Variance of Error term
15 S=(E'*E)/(T-2);
16 %Covariance of B
17 V=S.*inv(X'*X);
18 %standard errors
19 SE=diag(V).^0.5;
```

Publi

FIGURE 7. Example 5



```
1 clear
2 REPS=100; % number of repetitions
3 Z=zeros(REPS,1);
4 for i=1:REPS
5     Z(i,1)=i^2;
6 end
```

Publishe

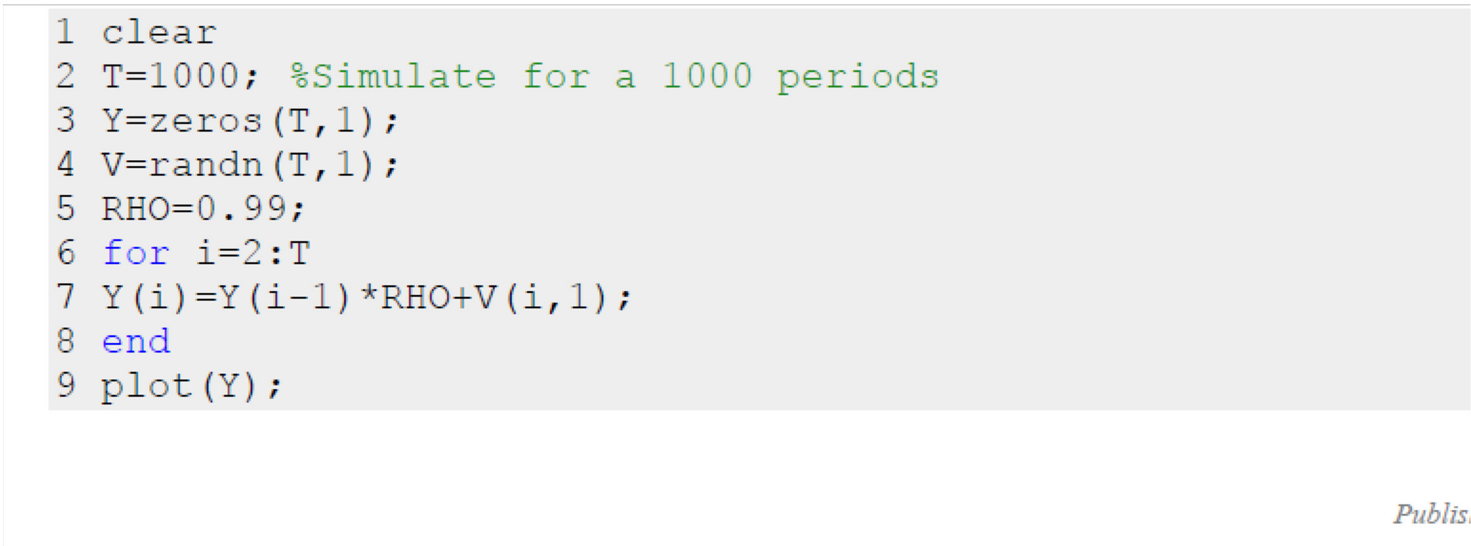
FIGURE 8. Example 6

$S \times (X'X)^{-1}$. Line 19 uses the diag command to extract the diagonal and then takes the square root by raising them to the power 0.5. This is done by the command .^ which raises each element of a matrix to a specified power.

Other useful matrix functions include the Cholesky decomposition (command chol, type help chol in the command window for details). A list of matrix functions can be seen by typing help matfun in the command window.

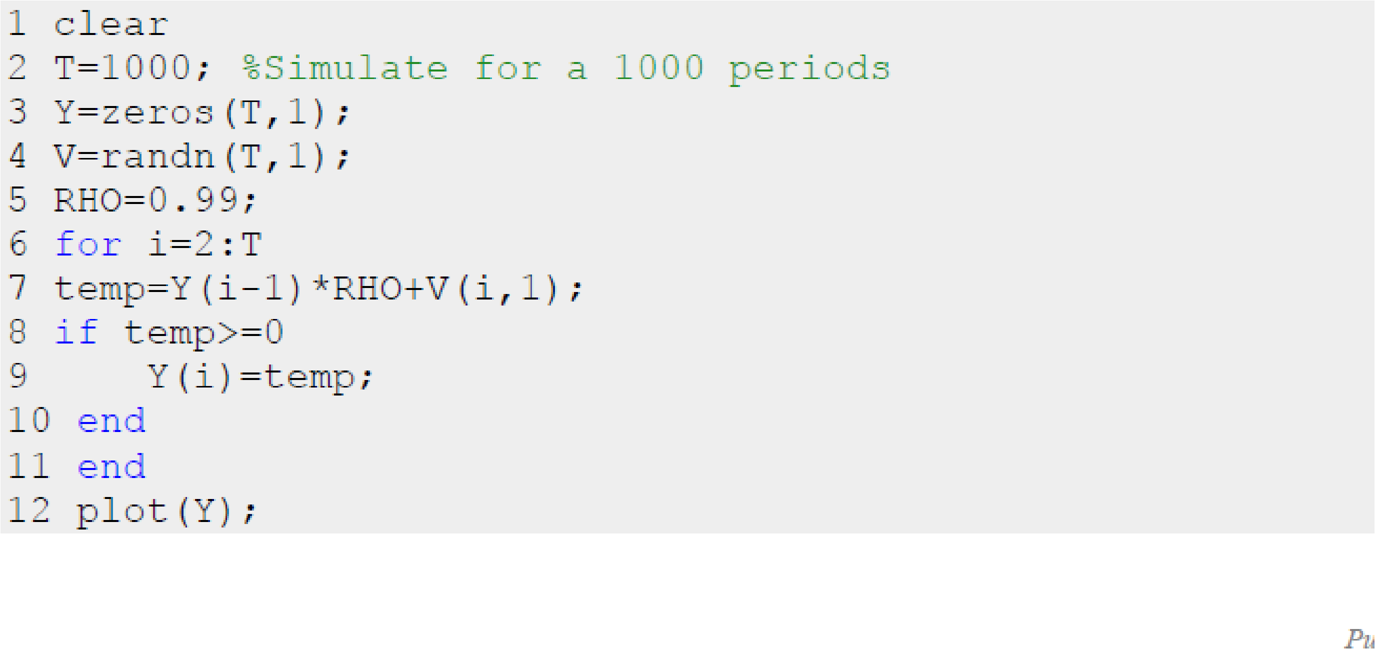
4. Program control

4.1. For Loops. One of the main uses of programming is to use the code to repeat tasks. This is used intensively in Bayesian simulation methods. The main type of loop we use is the *For loop*. To take a trivial example suppose we need to create a 100 × 1 matrix called Z where each element is equal to row number raised to power 2. So Z(1,1)=1, Z(2,1)=2^2, Z(3,1)=3^2 and so on. The code is shown in figure 8. Line 2 sets the total number of rows in the matrix Z. Line 3 creates the empty matrix Z.



```
1 clear
2 T=1000; %Simulate for a 1000 periods
3 Y=zeros(T,1);
4 V=randn(T,1);
5 RHO=0.99;
6 for i=2:T
7 Y(i)=Y(i-1)*RHO+V(i,1);
8 end
9 plot(Y);
```

FIGURE 9. Example 7



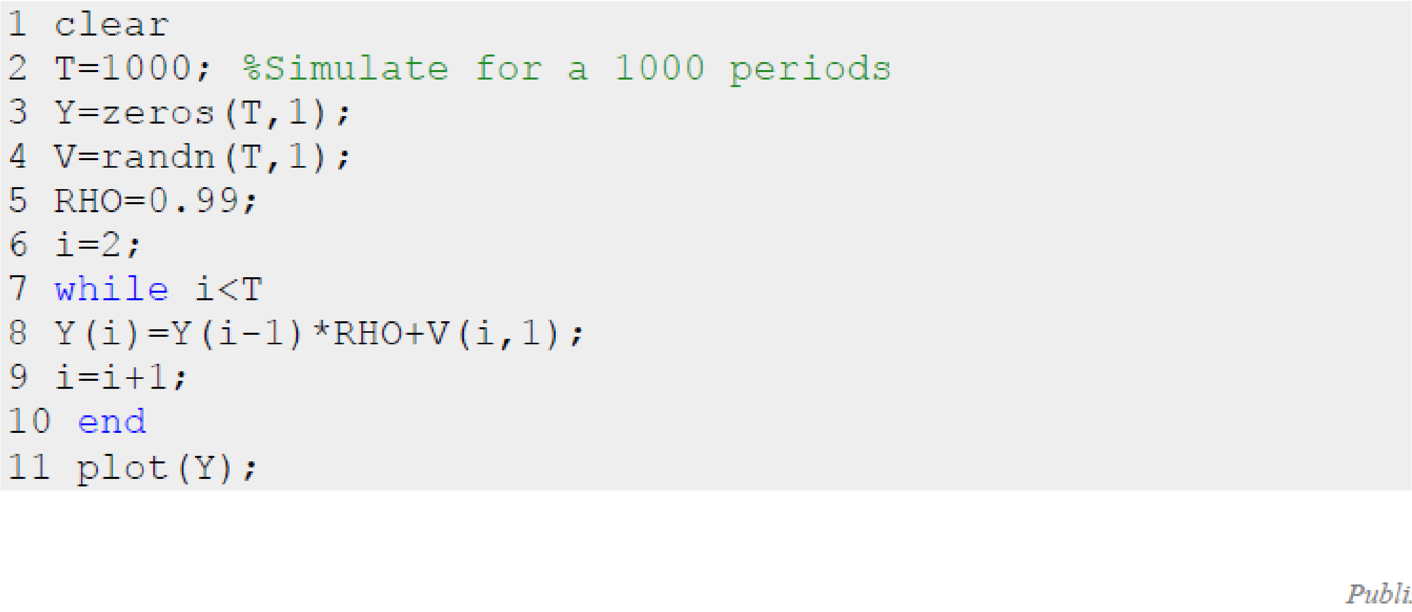
```
1 clear
2 T=1000; %Simulate for a 1000 periods
3 Y=zeros(T,1);
4 V=randn(T,1);
5 RHO=0.99;
6 for i=2:T
7 temp=Y(i-1)*RHO+V(i,1);
8 if temp>=0
9     Y(i)=temp;
10 end
11 end
12 plot(Y);
```

FIGURE 10. Example 8

Line 4 begins the for loop and instructs matlab to repeat the instruction on line 5 REPS times. That is the loop starts with i=1 and repeats the instruction below (instructions on lines before the end statement) until i=REPS. It increases i by 1 in each iteration. On line 5 the ith row of Z is set equal to i squared. Therefore when i=1, Z(1,1)=1, when i=2, Z(2,1)=2^2, when i=3, Z(3,3)=3^2 and so on. The instruction end on line 6 closes the for loop. Note if we had typed on line 4 *for i=1:-1:REPS*, i would be decreased by 1 in each iteration. If we had typed on line 4 *for i=1:2:REPS*, i would be increased by 2 in each iteration.

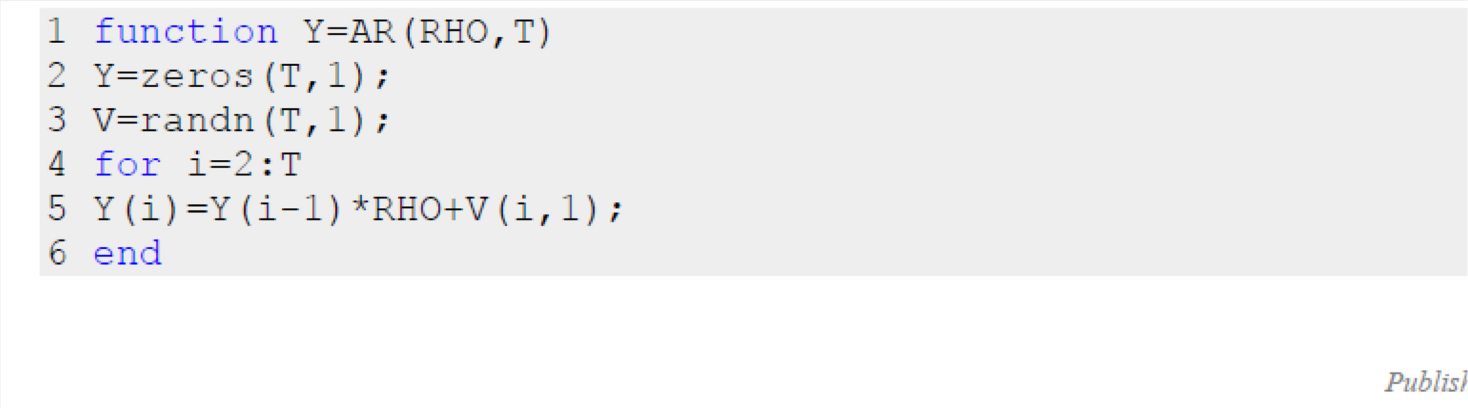
Figure 9 shows a second example where we simulate an $AR(1)$ process for 1000 periods $Y_t = \rho Y_{t-1} + V_t, t = 1...1000$. Where $V_t\tilde{}N(0,1)$. Line 3 of the code creates a $T \times 1$ matrix of zeros Y. Line 4 draws the error term from the standard normal distribution. Line 5 sets the AR coefficient $\rho = 0.99$. Line 6 starts the loop from period 2 going up to period T=1000. Line 7 simulates each value of $Y_t, t = 1....1000$ and line 8 ends the for loop. Line 9 plots the simulated series.

4.2. Conditional statements. Conditional statements instruct Matlab to carry out commands if a condition is met. Suppose, in example 7 above, we want to simulate the AR model but only want to retain values for Y which are greater than or equal to zero. We can do this by using the if statement in matlab. Figure 10 shows the matlab code. Lines 1 to 5 are exactly as before. Line 6 begins the for loop. Line 7 sets a variable $temp = \rho Y_{t-1} + V_t$. Line 8



```
1 clear
2 T=1000; %Simulate for a 1000 periods
3 Y=zeros(T,1);
4 V=randn(T,1);
5 RHO=0.99;
6 i=2;
7 while i<T
8 Y(i)=Y(i-1)*RHO+V(i,1);
9 i=i+1;
10 end
11 plot(Y);
```

Figure 11. Example 9



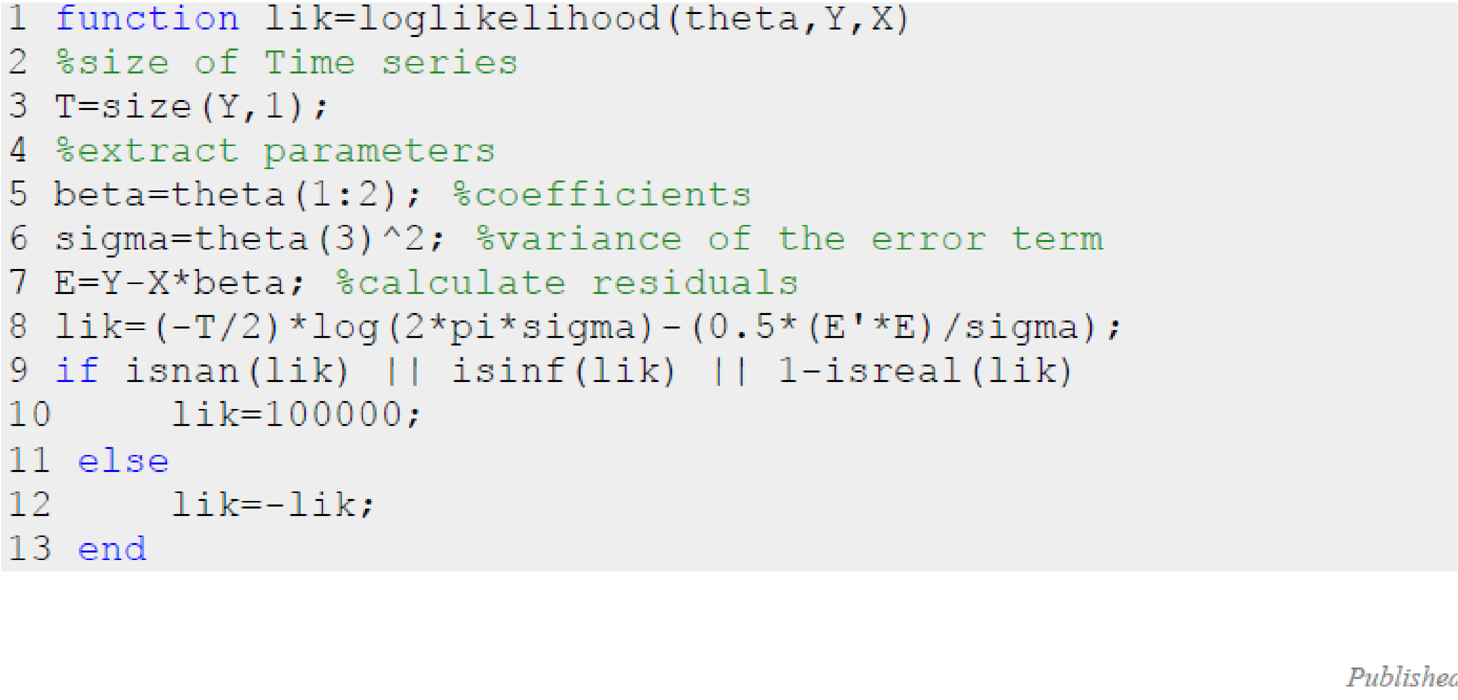
```
1 function Y=AR(RHO,T)
2 Y=zeros(T,1);
3 V=randn(T,1);
4 for i=2:T
5 Y(i)=Y(i-1)*RHO+V(i,1);
6 end
```

Figure 12. Example 10 ar.m

begins the if statement and instructs matlab to carry out the command on line 9 if the condition temp>=0 is true. Line 10 ends the if statement. Line 11 ends the for loop. If we wanted to retain negative values only, line 8 would change to *if temp<0.* If we wanted to retain values equal to zero, line 8 would change to *if temp==0.* If we wanted to retain all values not equal to zero, line 8 would change to *if temp˜=0.* If we wanted to retain values greater than 0 but less than 1, line 8 would change to if temp>0 & & temp<1. If we wanted to retain values greater than 0 or greater than 1 line 8 would change to if temp>0 || temp >1.

4.3. While Loops. While loops are loops which repeat matlab statements until a condition is met. The code in figure 11 re-formulates the for loop in example 7 using the while loop. Line 6 sets the starting point of the lopp at i=2. Line 7 starts the while loop and instructs matlab to perform tasks before the end statement until the condition i<T is true. On line 8 we simulate the AR(1) model as before. Line 9 increase the value of i by 1 in each iteration. Note that unlike the For loop, the index variable is not incremented automatically and this has to be done manually.

4.4. Functions. As our code becomes longer and more complicated, it is good practice to transfer parts of the code into seperate files called functions which can then be called from a main program in exactly the same way as built in matlab functions (like inv() etc) . Suppose we want to create a function called AR which takes as inputs the value of AR coefficient ρ and number of observations T and returns as output simulated $T \times 1$ matrix of data from this AR model. We can convert the code from example 7 into this function in a simple way. The code is shown in figure 12. The function begins with the word function. Then one specifies the output of the function (Y), the name of the function (AR) and the inputs (RHO,T). Lines 2 to 6 remain exactly the same. This function should be saved with the file name AR.m. The function (for e.g. with $\rho = 0.99, T = 100$)can be called from the command window (or from another piece of code) as Y=AR(0.99,100).



```
1  function lik=loglikelihood(theta,Y,X)
2  %size of Time series
3  T=size(Y,1);
4  %extract parameters
5  beta=theta(1:2); %coefficients
6  sigma=theta(3)^2; %variance of the error term
7  E=Y-X*beta; %calculate residuals
8  lik=(-T/2)*log(2*pi*sigma)-(0.5*(E'*E)/sigma);
9  if isnan(lik) || isinf(lik) || 1-isreal(lik)
10     lik=100000;
11 else
12     lik=-lik;
13 end
```

Figure 13. Example 11

5. Numerical optimisation

A key tool in Matlab is the ability find the maximum/minimum of a function numerically. This is important as we need these numerical tools to find the maximum of the likelihood functions. There are several built in optmising routines in Matlab. Here we focus on a minimisation routine written by Chris Sims called CSMINWEL (available from http://sims.princeton.edu/yftp/optimize/mfiles/. We have saved these files in the folder func). This routine has been known to work well for the type of models we consider in this course.

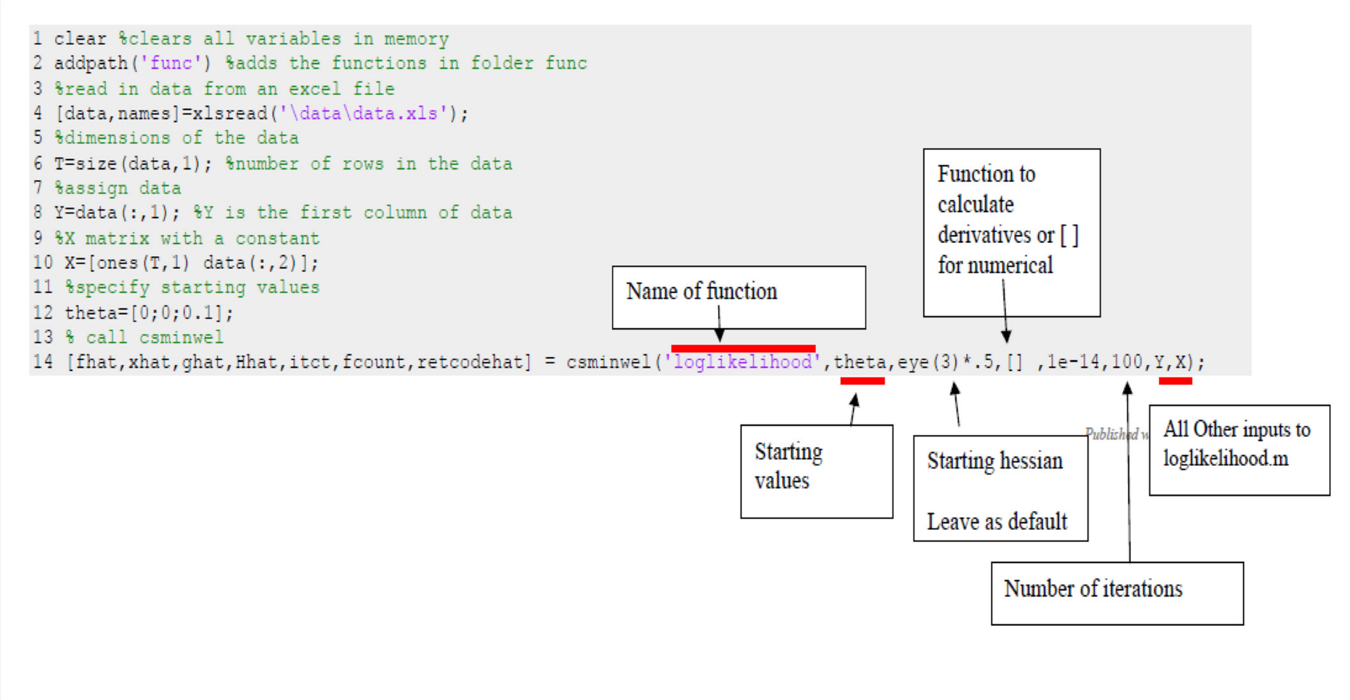
As an example we are going to maximise the likelihood function for the linear regression model considered in example 5. The log likelihood function is given by

$$\ln lik = -T/2 \ln \left(2\pi\sigma^2\right) - \frac{1}{2}\left(\frac{(Y - XB)'(Y - XB)}{\sigma^2}\right)$$

We need to maximise this with respect to B and σ^2. To use CSMINWEL we proceed in two steps

STEP1: We first need to write a function that takes in as input a set of values for B and σ^2 and returns the value of $\ln lik$ at that particular value of the parameters. The code to calculate the likelihood function is shown in 13. The function is called loglikelihood. It takes as input, the parameters theta, and the data series Y and X. The parameter vector needs to be the first argument. Line 5 extracts the regression coefficients. Line 6 extracts the standard deviation of the error term σ and squares it. Thus we optimise with respect to σ (and not σ^2). Line 6 ensures that the value of σ^2 will always be positive. Lines 7 and 8 calculate the likelihood function for a given B and σ^2. The consitional statement on line 9 checks for numerical problems. In particular, it checks if the log likelihood is not a number (isnan(lik), or it equals infinity (isinf(lik)) or is a complex number (1-isreal(lik) – isreal(lik) equals 1 if lik is real and thus 1-isreal(lik) equals 1 if lik is not a real number). In case of numerical problems, the negative of the log likelihood is set to a large number. If there are no numerical problems the function returns the negative of the calculated log likelihood. The function returns the negative of the log likelihood as CSMINWEL is a minimiser (i.e. we minimise the negative of the log likelihood and this is equivalent to maximising the log likelihood).

STEP2: We use CSMINWEL to minimise the negative log likelihood calculated by loglikelihood.m. This code can be seen in figure 14. Line 2 ensures that the files required for CSMINWEL in the folder func can be found by Matlab. Lines 4 to 10 load the data and create the Y and X matrix. Line 12 specifies the initial values of the K parameters. Line 14 calls the function csminwel. The first input argument is the name of the function that calculates the log likelihood. The second input are the starting values. The tird input is the starting value of the inverse hessian. This can be left as default as a $K \times K$ matrix with diagonal elements equal to 0.5. If the next argument is set equal to [ ] csminwel uses numerical derivitives in the optimisation. This is the default un all out applications. The next argument is the convergence tolerance which should be left as defaults. The next input argument are the number of maximum iterations. The remaining inputs are passed directly to the function loglikelihood.m after the parameter values. The function returns the minimum of the negative log likelihood in fhat, the values of the parameters at the



```
clear %clears all variables in memory
addpath('func') %adds the functions in folder func
%read in data from an excel file
[data,names]=xlsread('\data\data.xls');
%dimensions of the data
T=size(data,1); %number of rows in the data
%assign data
Y=data(:,1); %Y is the first column of data
%X matrix with a constant
X=[ones(T,1) data(:,2)];
%specify starting values
theta=[0;0;0.1];
% call csminwel
[fhat,xhat,ghat,Hhat,itct,fcount,retcodehat] = csminwel('loglikelihood',theta,eye(3)*.5,[] ,1e-14,100,Y,X);
```

Function to calculate derivatives or [ ] for numerical
Name of function
Starting values
Starting hessian
Leave as default
All Other inputs to loglikelihood.m
Number of iterations

Figure 14. Example 12

minimum in xhat and the inverse hessian as Hhat.Retcodehat=0 if convergence occurs. Running this code produce the same value of B as the OLS formula in the examples above.

Bibliography

Albert, James H. and Siddhartha Chib, 1993, Bayesian Analysis of Binary and Polychotomous Response Data, *Journal of the American Statistical Association* **88**(422), 669–679.
URL: *http://www.tandfonline.com/doi/abs/10.1080/01621459.1993.10476321*

An, Sungbae and Frank Schorfheide, 2007, Bayesian Analysis of DSGE Models, *Econometric Reviews* **26**(2-4), 113–172.
URL: *http://dx.doi.org/10.1080/07474930701220071*

Banbura, Marta, Domenico Giannone and Lucrezia Reichlin, 2007, Bayesian VARs with Large Panels, *CEPR Discussion Papers 6326*, C.E.P.R. Discussion Papers.
URL: *http://ideas.repec.org/p/cpr/ceprdp/6326.html*

Baumeister, Christiane, Philip Liu and Haroon Mumtaz, 2013, Changes in the effects of monetary policy on disaggregate price dynamics, *Journal of Economic Dynamics and Control* **37**(3), 543–560.
URL: *https://ideas.repec.org/a/eee/dyncon/v37y2013i3p543-560.html*

Bauwens, Luc, Michel Lubrano and Jean François Richard, 1999, *Bayesian inference in dynamic econometric models*, Oxford University Press.

Benati, Luca and Haroon Mumtaz, 2006, US evolving Macroeconomic Dynamics: A structural investigation., Mimeo European Central Bank.

Bernanke, Ben, Jean Boivin and Piotr S. Eliasz, 2005, Measuring the Effects of Monetary Policy: A Factor-augmented Vector Autoregressive (FAVAR) Approach, *The Quarterly Journal of Economics* **120**(1), 387–422.
URL: *http://ideas.repec.org/a/tpr/qjecon/v120y2005i1p387-422.html*

Bianchi, Francesco, Haroon Mumtaz and Paolo Surico, 2009, The great moderation of the term structure of UK interest rates, *Journal of Monetary Economics* **56**(6), 856–871.
URL: *https://ideas.repec.org/a/eee/moneco/v56y2009i6p856-871.html*

Canova, Fabio, 2007, *Methods for Applied Macroeconomic Research*, Princeton University Press, Princeton.

Canova, Fabio and Luca Sala, 2009, Back to square one: Identification issues in DSGE models, *Journal of Monetary Economics* **56**(4), 431–449.
URL: *https://ideas.repec.org/a/eee/moneco/v56y2009i4p431-449.html*

Carriero, Andrea, George Kapetanios and Massimiliano Marcellino, 2010, Forecasting Government Bond Yields with Large Bayesian VARs, *Working Papers 662*, Queen Mary, University of London, School of Economics and Finance.
URL: *http://ideas.repec.org/p/qmw/qmwecw/wp662.html*

Carter, C. K. and R. Kohn, 1994, On Gibbs sampling for state space models, *Biometrika* **81**(3), 541–553.
URL: *http://biomet.oxfordjournals.org/content/81/3/541.abstract*

Casella, George and Edward I. George, 1992, Explaining the Gibbs Sampler, *The American Statistician* **46**(3), pp. 167–174.
URL: *http://www.jstor.org/stable/2685208*

Chen, Cathy W. S. and Jack C. Lee, 1995, BAYESIAN INFERENCE OF THRESHOLD AUTOREGRESSIVE MODELS, *Journal of Time Series Analysis* **16**(5), 483–492.
URL: *http://dx.doi.org/10.1111/j.1467-9892.1995.tb00248.x*

Chib, Siddhartha, 1993, Bayes regression with autoregressive errors : A Gibbs sampling approach, *Journal of Econometrics* **58**(3), 275–294.
URL: *http://ideas.repec.org/a/eee/econom/v58y1993i3p275-294.html*

Chib, Siddhartha, 1995, Marginal Likelihood from the Gibbs Output, *Journal of the American Statistical Association* **90**(432), 1313–1321.
URL: *http://www.jstor.org/stable/2291521*

Chib, Siddhartha, 1996, Calculating posterior distributions and modal estimates in Markov mixture models, *Journal of Econometrics* **75**(1), 79 – 97.
URL: *http://www.sciencedirect.com/science/article/pii/0304407695017704*

Chib, Siddhartha, 1998, Estimation and comparison of multiple change-point models, *Journal of Econometrics* **86**(2), 221 – 241.
URL: *http://www.sciencedirect.com/science/article/pii/S0304407697001152*

Chib, Siddhartha and Srikanth Ramamurthy, 2010, Tailored randomized block MCMC methods with application to DSGE models, *Journal of Econometrics* **155**(1), 19–38.
URL: *http://ideas.repec.org/a/eee/econom/v155y2010i1p19-38.html*

Cogley, Timothy and Thomas J. Sargent, 2002, Evolving Post-World War II U.S. Inflation Dynamics, *NBER Macroeconomics Annual 2001, Volume 16*, NBER Chapters, National Bureau of Economic Research, Inc, pp. 331–388.
URL: *http://ideas.repec.org/h/nbr/nberch/11068.html*

Doan, Thomas, Robert B. Litterman and Christopher A. Sims, 1983, Forecasting and Conditional Projection Using Realistic Prior Distributions, *NBER Working Papers 1202*, National Bureau of Economic Research, Inc.
URL: *http://ideas.repec.org/p/nbr/nberwo/1202.html*

Ellis, Colin, Haroon Mumtaz and Pawel Zabczyk, 2014, What Lies Beneath? A Time-varying FAVAR Model for the UK Transmission Mechanism, *The Economic Journal* **124**(576), 668–699.
URL: *http://dx.doi.org/10.1111/ecoj.12147*

Filardo, Andrew J. and Stephen F. Gordon, 1998, Business cycle durations, *Journal of Econometrics* **85**(1), 99–123.
URL: *https://ideas.repec.org/a/eee/econom/v85y1998i1p99-123.html*

Fry, Renee and Adrian Pagan, 2007, Some Issues in Using Sign Restrictions for Identifying Structural VARs, *NCER Working Paper Series 14*, National Centre for Econometric Research.
URL: *http://ideas.repec.org/p/qut/auncer/2007-8.html*

Gelfand, A. E. and D. K. Dey, 1994, Bayesian Model Choice: Asymptotics and Exact Calculations, *Journal of the Royal Statistical Society. Series B (Methodological)* **56**(3), pp. 501–514.
URL: *http://www.jstor.org/stable/2346123*

Gelman, Andrew and Donald B. Rubin, 1992, Inference from Iterative Simulation Using Multiple Sequences, *Statistical Science* **7**(4), pp. 457–472.
URL: *http://www.jstor.org/stable/2246093*

Geweke, John, 1991, Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments, *Technical report*.

Geweke, John, 1998, Using simulation methods for Bayesian econometric models: inference, development, and communication, *Technical report*.

Hamilton, J. D., 1994, *Time series analysis*, Princeton University Press, Princeton.

Herbst, E. and F. Schorfheide, 2015, *Bayesian Estimation of DSGE Models*, Princeton University Press, Princeton.

Jacquier, E, N Polson and P Rossi, 2004, Bayesian analysis of stochastic volatility models, *Journal of Business and Economic Statistics* **12**, 371–418.

Kadiyala, K Rao and Sune Karlsson, 1997, Numerical Methods for Estimation and Inference in Bayesian VAR-Models, *Journal of Applied Econometrics* **12**(2), 99–132.
URL: *http://ideas.repec.org/a/jae/japmet/v12y1997i2p99-132.html*

Kim, C-J. and C. R. Nelson, 1999, *State-space models with regime switching*, MIT Press, Cambridge, Massachusetts.

Kim, Sangjoon, Neil Shephard and Siddhartha Chib, 1998, Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models, *Review of Economic Studies* **65**(3), 361–93.
URL: *http://ideas.repec.org/a/bla/restud/v65y1998i3p361-93.html*

Koop, Gary, 2003, *Bayesian Econometrics*, Wiley.

Liu, Philip, Haroon Mumtaz and Angeliki Theophilopoulou, 2014, The transmission of international shocks to the UK. Estimates based on a time-varying factor augmented VAR, *Journal of International Money and Finance* **46**(C), 1–15.
URL: *https://ideas.repec.org/a/eee/jimfin/v46y2014icp1-15.html*

Lopes, Hedibert F. and Esther Salazar, 2006, Bayesian Model Uncertainty In Smooth Transition Autoregressions, *Journal of Time Series Analysis* **27**(1), 99–117.
URL: *http://dx.doi.org/10.1111/j.1467-9892.2005.00455.x*

Lubik, Thomas A. and Frank Schorfheide, 2007, Do central banks respond to exchange rate movements? A structural investigation, *Journal of Monetary Economics* **54**(4), 1069 – 1087.
URL: *http://www.sciencedirect.com/science/article/pii/S0304393206002108*

Mumtaz, Haroon and Konstantinos Theodoridis, 2017, Common and country specific economic uncertainty, *Journal of International Economics* **105**, 205 – 216.
URL: *http://www.sciencedirect.com/science/article/pii/S0022199617300090*

Mumtaz, Haroon and Paolo Surico, 2012, EVOLVING INTERNATIONAL INFLATION DYNAMICS: WORLD AND COUNTRYSPECIFIC FACTORS, *Journal of the European Economic Association* **10**(4).

Negro, Marco Del and Christopher Otrok, 2008, Dynamic factor models with time-varying parameters: measuring changes in international business cycles, *Technical report*.

Negro, Marco Del and Frank Schorfheide, 2004, Priors from General Equilibrium Models for VARS, *International Economic Review* **45**(2), 643–673.
URL: *http://ideas.repec.org/a/ier/iecrev/v45y2004i2p643-673.html*

Primiceri, G, 2005, Time varying structural vector autoregressions and monetary policy, *The Review of Economic Studies* **72**(3), 821–852.

Ramirez, Juan Rubio, Daniel Waggoner and Tao Zha, 2010, Structural Vector Autoregressions: Theory of Identification and Algorithms for Inference, *Review of Economic Studies* **77**(2), 665–696.
URL: *http://ideas.repec.org/a/bla/restud/v77y2010i2p665-696.html*

Robertson, John C. and Ellis W. Tallman, 1999, Vector autoregressions: forecasting and reality, *Economic Review* (Q1), 4–18.
URL: *http://ideas.repec.org/a/fip/fedaer/y1999iq1p4-18nv.84no.1.html*

Robertson, John C, Ellis W Tallman and Charles H Whiteman, 2005, Forecasting Using Relative Entropy, *Journal of Money, Credit and Banking* **37**(3), 383–401.
URL: *http://ideas.repec.org/a/mcb/jmoncb/v37y2005i3p383-401.html*

Ruge-Murcia, Francisco J., 2007, Methods to estimate dynamic stochastic general equilibrium models, *Journal of Economic Dynamics and Control* **31**(8), 2599 – 2636.
URL: *http://www.sciencedirect.com/science/article/pii/S0165188906001758*

Schorfheide, Frank and Dongho Song, 2015, Real-Time Forecasting With a Mixed-Frequency VAR, *Journal of Business & Economic Statistics* **33**(3), 366–380.
URL: *https://ideas.repec.org/a/taf/jnlbes/v33y2015i3p366-380.html*

Sims, Christopher A., 2002, Solving Linear Rational Expectations Models, *Computational Economics* **20**(1), 1–20.
URL: *http://dx.doi.org/10.1023/A:1020517101123*

Sims, Christopher A., Daniel F. Waggoner and Tao Zha, 2008, Methods for inference in large multiple-equation Markov-switching models, *Journal of Econometrics* **146**(2), 255–274.
URL: *https://ideas.repec.org/a/eee/econom/v146y2008i2p255-274.html*

Sims, Christopher A and Tao Zha, 1998, Bayesian Methods for Dynamic Multivariate Models, *International Economic Review* **39**(4), 949–68.
URL: *http://ideas.repec.org/a/ier/iecrev/v39y1998i4p949-68.html*

Sims, Christopher A. and Tao Zha, 1999, Error Bands for Impulse Responses, *Econometrica* **67**(5), 1113–1156.
URL: *https://ideas.repec.org/a/ecm/emetrp/v67y1999i5p1113-1156.html*

STOCK, JAMES H. and MARK W. WATSON, 2007, Why Has U.S. Inflation Become Harder to Forecast?, *Journal of Money, Credit and Banking* **39**, 3–33.
URL: *http://dx.doi.org/10.1111/j.1538-4616.2007.00014.x*

Villani, Mattias, 2009, Steady-state priors for vector autoregressions, *Journal of Applied Econometrics* **24**(4), 630–650.
URL: *http://dx.doi.org/10.1002/jae.1065*

Waggoner, Daniel F. and Tao Zha, 1997, Normalization, probability distribution, and impulse responses, *Technical report*.

Waggoner, Daniel F. and Tao Zha, 1999, Conditional Forecasts In Dynamic Multivariate Models, *The Review of Economics and Statistics* **81**(4), 639–651.
URL: *http://ideas.repec.org/a/tpr/restat/v81y1999i4p639-651.html*

Waggoner, Daniel F. and Tao Zha, 2003, Likelihood preserving normalization in multiple equation models, *Journal of Econometrics* **114**(2), 329 – 347.
URL: *http://www.sciencedirect.com/science/article/pii/S0304407603000873*

Zellner, Arnold, 1971, *An introduction to Bayesian Inference in Econometrics*, Wiley.