User Guide to MATLAB Bayesian Estimation Routines

Timothy Kam, Kirdan Lees and Philip Liu

February 7, 2006

1 Introduction

This set of MATLAB codes implement a Bayesian estimation procedure for a medium-scale dynamic stochastic general equilibrium (DSGE) model for a small open economy with optimal monetary policy. The user needs to specify the linear (or linearized) structure of the DSGE model and also the central bank's objective function, which is restricted to a quadratic form.

We have applied these routines to time series data for three small open economies: New Zealand, Canada and Australia. These country models are estimated independently, assuming that they are small open economies that take the evolution of foreign variables as exogenous. The details of our example model are in Kam, Lees and Liu (2006).

2 Computation of Bayesian posterior distributions

Each model can be thought of as summarized by the set of parameters, $\theta \in \Theta$ where $\Theta \subset \mathbb{R}^k$. In reality, on a computer, Θ is a countable and finite set. Given a model, M, defined by its parameters θ, we can compute the likelihood of the data being generated by this model as $L(y|\theta, M)$. By using a Bayesian perspective, we weigh this likelihood with our own prior beliefs about where the model should "locate", using a prior density on the parameters $p(\theta)$. By Bayes' rule, we have the posterior density on the model as

$$p(\theta|y) = \frac{L(y|\theta, M)\, p(\theta)}{p(y)} \tag{1}$$

Since the Bayesian take the data as fixed, we can express this as

$$p(\theta|y) \propto L(y|\theta, M)\, p(\theta) \tag{2}$$

We compute the posterior density, $p(\theta|y)$, using a Markov Chain Monte Carlo (MCMC) method.

2.1 Existence and stability of an invariant posterior distribution

There is a connection between the existence and uniqueness of a posterior distribution for θ and stability of a sequence of candidate θ's constructed as a Markov Chain. This

also implies that the idea of using a Markov Chain Monte Carlo (MCMC) method to compute the posterior density is a theoretically good one. The following discussion and analysis follows Norris (1997) closely. For a more practical "cookbook" approach, see Koop (2003).

We want to work with the approximation to $p(\theta|y)$ using the invariant distribution of a sequence $\{\theta_n\}_{n\geq 0}$ constructed as a Markov chain. In practice, a parameter region located at site m, Θ_m, as represented on the computer is finite. We will thus be analyzing the algorithm in a MCMC state space in product form defined as

$$I = \prod_{m\in\Lambda} \Theta_m$$

where we also assume that Λ is finite. We can partition I into m disjoint subsets Θ_m. We can think of the set of candidate parameters, $X := \{\theta(m) : m \in \Lambda\}$ as a random variable, where $X \in I$, and for every site $m \in \Lambda$, $\theta(m) \in \Theta_m$.

We want an approximate distribution to the distribution constructed from the unobserved posterior density $p(\theta|y)$. Define the unknown distribution as $\pi = \{\pi_i : i \in I\}$, so that we can compute sample moments of interest from this approximate distribution as:

$$\sum_{i\in I} \pi_i f(\theta(i)) \tag{3}$$

However, the problem is that typically, Λ is a very large set, so then the product space I will be very large! In such cases, computing the sum (3) above, site by site, becomes infeasible. A sampling approach would be to simulate a long sequence of random variables, $\theta_1, ..., \theta_N$, that would live in the state space I, where each sequence has distribution π, so then we can approximate (3) with the sample counterpart, $N^{-1}\sum_{n=1}^{N} f(\theta_n)$, where we know by the strong law of large numbers,

$$\frac{1}{N}\sum_{n=1}^{N} f(\theta_n) \to \sum_{i\in I} \pi_i f(\theta(i))$$

with probability 1. Even, then sampling from the distribution π can also be difficult when π does not have product form: Norris (1997).

We can exploit sampling using a MCMC approach. The idea is to simulate a Markov chain $\{\theta_n\}_{n\geq 1}$ which by construction would have an invariant distribution π. This is much easier than simulating the whole distribution π since the state space for the Markov chain is the product space I. That is, the chain $X := \{\theta_n\}_{n\geq 1}$ evolves by changing its components one site m at a time. When a site m is chosen, we simulate a new random vector $\theta_{n+1}(m) \in \Theta_m$. Thus at each stage of the chain, we only simulate one component of X at a time in the smaller space Θ_m rather than the whole state space I. By doing so, we can construct, step by step, the whole random variable X with distribution π.

Following Norris (1997) we define $i \overset{m}{\sim} j$ if i and j agree, except possibly at site m. We require that a new value $\theta_{n+1}(m)$ to be generated at site m follow the law described

by a stochastic matrix $P(m)$ where $p_{ij}(m) = 0$ unless $i \overset{m}{\sim} j$. Thus, for every site m, there is a $P(m)$, and thus, a family of such transitional probability matrices can be constructed, one $P(m)$ for each site m.

A sufficient condition for π to be an invariant distribution for $P(m)$ is to have, for all $i, j \in I$,

$$\pi_i p_{ij}(m) = \pi_j p_{ji}(m)$$

These are known as detailed balance equations. (See Lemma 1.9.2 in Norris (1997).)

For any candidate stochastic matrix $R(m)$ with elements $r_{ij}(m) = 0$ unless $i \overset{m}{\sim} j$, we can uncover $P(m)$ by:

$$\pi_i p_{ij}(m) = \pi_i r_{ij}(m) \wedge \pi_j r_{ji}(m), \forall i \neq j$$

and then

$$p_{ii}(m) = 1 - \sum_{j \neq i} p_{ij}(m) \geq 0.$$

The interpretation of the above law is as follows, which is known as a Hastings algorithm. If $\theta_n = \theta(i)$, we simulate a new random vector $\tilde{\theta}_{n+1} = \theta(j)$ with probability $r_{ij}(m)$. If $\tilde{\theta}_{n+1} = \theta(j)$ we set

$$\theta_{n+1} = \begin{cases} \tilde{\theta}_{n+1} & \text{with probability } (\pi_j r_{ji}(m) / \pi_i r_{ij}(m)) \wedge 1 \\ \theta_n & \text{otherwise} \end{cases}$$

A special case takes $r_{ij}(m) = r_{ji}(m)$ for all i and j and defines $P(m)$ by

$$p_{ij}(m) = ((\pi_j / \pi_i) \wedge 1) r_{ij}(m)$$

for $i \overset{m}{\sim} j$ and $i \neq j$. This is called the Metropolis algorithm. The idea is to pick a random value $\theta_{n+1} = \theta(j_m)$ at site m. If the posterior probability associated with the value $\theta(j_m)$ is greater than the posterior probability associated with $\theta(i)$, $\pi_j > \pi_i$, the chain is instructed to move to the new value, $\theta_{n+1} = \theta(j_m)$ with probability 1, otherwise, we adopt the new value with probability $\pi_j / \pi_i < 1$ (i.e. we reject the new value with probability $1 - \pi_j / \pi_i$ and set $\theta_{n+1} = \theta_n$).

One example, which we use, is the random walk Metropolis algorithm, where:

$$\theta_{n+1} = \theta_n + z_{n+1}; z \sim \varphi; \tag{4}$$

where we assumed φ to be a jointly normal density, $N(0, V_k)$. Further, the posterior probabilities, π_i, π_j are computed using the Bayes rule (2) which will be dependent on the model (θ_n, M).

Since $\{\theta_n\}_{n \geq 0}$ visits a site randomly at each stage in the random walk Metropolis algorithm, $\{\theta_n\}_{n \geq 0}$ is itself Markov-(P, λ), where the stochastic matrix is given as

$$P = |\Lambda|^{-1} \sum_{m \in \Lambda} P(m).$$

and λ is the initial distribution of the chain.

By construction, P will be irreducible. We also know that

$$\pi_i p_{ij}(m) = \pi_j p_{ji}(m), \forall m, \forall i, j$$

and it follows that

$$\pi_i p_{ij} = \pi_j p_{ji}, \forall i, j.$$

So π is the unique invariant measure for P. Thus, by the ergodicity theorem (see e.g. Theorem 1.10.2 in Norris (1997)) of an irreducible Markov chain, we have

$$\Pr\left(\frac{1}{N}\sum_{n=1}^{N} f(\theta_n) \to \sum_{i\in I} \pi_i f(\theta(i)), n \to +\infty\right) = 1.$$

where $f : I \to R$ is a bounded function.

In summary, by simulating a long enough Markov chain in the parameter state space I, we can approximate the moments of the posterior distribution π of these parameters with the distribution of the random variable $X; = \{\theta_n\}_{n\geq 0}$ generated by the MCMC algorithm.

2.2 The MATLAB files

The format for naming files are as follows. We have countries defined by the collection `COUNTRY` which contains `aus`, `nzd`, and `can`, respectively for Australia, New Zealand and Canada.

The master script is called `COUNTRY_estimate.m`. Purpose:

- Estimation of DGSE model using the Random Walk Metropolis-Hasting Markov Chain Monte Carlo method.
- Approximates prior and posterior densities of parameter vector.

`COUNTRY_estimate.m` calls the following functions:

- `aus_datadoc.m` to manipulate and plot data series. Custom functions called:
  - `multiplot2D.m` creates figures with subplots of raw and filtered/detrended data.
  - `suptitle2.m` create a master title in the figures with subplots instead of titles for each current subplot.
  - `hpfilter.m` calculates the Hodrick-Prescott filtered trend from a time series.
  - `ols.m` is James P. LeSage's object-oriented code for running OLS regressions. You need to install the public domain Econometrics Toolbox from LeSage's website: http://www.spatial-econometrics.com/.

- `model_likelihood.m` for evaluating the log-likelihood value of the model given the data matrix, and vector of model parameters. Functions called:
  - `model_solve.m` Solves for the rational expectations equilibrium and expresses this as a state-space form stochastic dynamical system. You need to specify the structure of your own model here. Also, we have called two other functions, `discretion.m` and `commitment.m`, from within `model_solve.m` to compute a policy-maker's optimal policy under either commitment (set option `POLICY`= 1) or discretion (set option `POLICY`= 0). `model_solve.m` also calls `quadpref.m` to input the quadratic preferences of the policy maker.
  - `doublek.m` is Sargent and Hansen's doubling algorithm used to compute the steady state Kalman gain and MSE covariance matrix. This will be used as initial objects for starting the Kalman filter likelihood computation of the state-space model solution, each time `model_likelihood.m` is called.
- `priorln.m` constructs the prior densities of the vector of parameters. depending on your specification of the prior densities of your parameters, this function calls the following density functions:
  - `lpdfbeta.m` computes the value of the log-Beta density.
  - `lpdfig1.m` computes the value of the log-inverse-Gamma density (Type 1).
  - `lpdfgamma.m` computes the value of the log-Gamma density.
  - `lpdfnorm.m` computes the value of the log-Normal density.
- `genTheta.m` generates new parameter vectors, θ, using the random walk Metropolis algorithm. You can manipulate the incremental step of the θ random walk by drawing from a more dispersed density φ.

References

Kam, Timothy, Kirdan Lees and Philip Liu (2006). What do central banks care about? A tale of three small open economies. Available from: www.rbnz.govt.nz

Koop, Gary (2003). Bayesian Econometrics. Wiley & Sons.

Norris, James R. (1997). Markov Chains. Cambridge Series in Probabilistic and Statistical Mathematics. Cambridge University Press.