Chapter 4

Nonlinear Regression: Introduction to Metropolis-Hastings

4.1 Theory

As mentioned earlier, the linear model with Normal errors has two very convenient (and closely related) properties:

1. It has a set of sufficient statistics
2. The likelihood can be inverted easily to form a kernel for the density of the parameters.

Because it has that set of sufficient statistics (the crossproduct matrix), you can do even millions of simulations in a very modest amount of time, because each simulation involves a set of calculations with matrices of modest size (number of regressors). There aren't any calculations the size of the data set in the inner loop.

In general, a non-linear regression model has neither of these nice properties. If

$$y_i = f(X_i, \gamma) + \varepsilon_i, \varepsilon_i \sim N\left(0, h^{-1}\right) i.i.d.$$

the log likelihood is

$$-\frac{N}{2}\log 2\pi + \frac{N}{2}\log h - \frac{h}{2}\sum_{i=1}^{N}\left(y_i - f\left(X_i, \gamma\right)\right)^2$$

While that last term has a $\mathbf{Y}'\mathbf{Y}$ component, that won't help much. Now, there is a class of special cases which at least has sufficient statistics. These are *non-linear in parameters* (NLIP) models. This is short for "linear in the data, non-linear in the parameters". For these models,

$$f(X_i, \gamma) = X_i g(\gamma)$$

The final term in the log-likelihood for this is now:

$$-\frac{h}{2}\sum_{i=1}^{N}\left(y_i - X_i g\left(\gamma\right)\right)^2 = -\frac{h}{2}\left(\mathbf{Y}'\mathbf{Y} - 2\mathbf{Y}'\mathbf{X}g\left(\gamma\right) + g\left(\gamma\right)'\mathbf{X}'\mathbf{X}g\left(\gamma\right)\right)$$

so we have the same set of sufficient statistics as before. This can be a big time saver if it's available.

The lack of sufficient statistics mainly affects the time required to do a certain number of simulations. The lack of an easily identifiable kernel for the posterior is a more serious problem. We were able to do Gibbs sampling in the linear model with independent Normal-gamma prior because we had that. Assume that we again have an independent Normal-gamma prior.

$$p(\gamma, h) \propto \exp\left(-\frac{1}{2}(\gamma-\gamma)' \mathbf{H} (\gamma-\gamma)\right) \times h^{(v/2)-1} \exp\left(-\frac{h\nu}{2s^{-2}}\right)$$

The conditional posterior for $h|\gamma$ is exactly the same as before:

$$p(h|\gamma, y) \propto h^{(N+v)/2-1} \exp\left(-\frac{h}{2}\left(\nu s^2 + \varepsilon(\gamma)' \varepsilon(\gamma)\right)\right)$$

where

$$\varepsilon(\gamma) = y - f(X, \gamma)$$

The difference is that, unless we have an NLIP model, we'll have to run a loop over observations to compute $\varepsilon(\gamma)' \varepsilon(\gamma)$.

It's $\gamma|h$ that causes the problems. Getting rid of factors that don't depend upon γ, we get

$$p(\gamma|h, y) \propto \exp\left(-\frac{1}{2}(\gamma-\gamma)' \mathbf{H} (\gamma-\gamma)\right) \exp\left(-\frac{h}{2}\varepsilon(\gamma)' \varepsilon(\gamma)\right) \tag{4.1}$$

While this can be computed easily enough (again, requiring a loop over observations to evaluate the second factor), it can't be written in the form of a known density.

Unlike the Gibbs sampler in the linear model, anything we do here is going to require at least some ingenuity or at least some experimentation. The most straightforward way to handle this is to use a more advanced form of MCMC called Metropolis-Hastings (or more correctly here, Metropolis within Gibbs, since we're using this for just part of a Gibbs sampler, with standard methods taking care of draws for $h|\gamma$).

The idea behind this is as follows: suppose that our current draw for γ is called $\gamma^{(i-1)}$. We want to generate a new draw $\gamma^{(i)}$ from $p(\gamma|h, y)$, but there's no known procedure for drawing from that. Instead, let's choose γ^* from a more convenient density $q(\gamma)$ (often called the *proposal density*). We can evaluate both the density $p(\gamma^*|h, y)$ and the density $q(\gamma^*)$. Compute

$$\alpha = \frac{p(\gamma^*|h, y)}{p(\gamma^{(i-1)}|h, y)} \times \frac{q\left(\gamma^{(i-1)}\right)}{q(\gamma^*)} \tag{4.2}$$

With probability α, we accept the new draw and make $\gamma^{(i)} = \gamma^*$, otherwise we stay with our previous value and make $\gamma^{(i)} = \gamma^{(i-1)}$. Note that it's possible to have $\alpha > 1$, in which case, we just accept the new draw.

The first ratio in (4.2) makes perfect sense. We want, as much as possible, to have draws where the posterior density is high, and not where it's low. The

second counterweights (notice that it's the ratio in the opposite order) for the probability of drawing a given value. Another way of looking at the ratio is

$$\alpha = \frac{p(\gamma^*|h,y)/q\left(\gamma^*\right)}{p(\gamma^{(i-1)}|h,y)/q\left(\gamma^{(i-1)}\right)}$$

p/q is a measure of the relative desirability of a draw. The ones that really give us a strong "move" signal are where the target density (p) is high and the proposal density (q) is low; we may not see those again, so when we get a chance we should move. Conversely, if p is low and q is high, we might as well stay put; we may revisit that one at a later time. Note that $1/\alpha$ is the probability of moving back if we took $\gamma^*$and drew $\gamma^{(i-1)}$ as a candidate.

This is known as *Independence Chain* M-H, where the candidates are drawn independently of the previous value. In greater generality, suppose that proposal density is dependent upon the previous value $\gamma^{(i-1)}$: $q\left(\gamma^*|\gamma^{(i-1)}\right)$. In that case α takes the form:

$$\alpha = \frac{p(\gamma^*|h,y)/q\left(\gamma^*|\gamma^{(i-1)}\right)}{p(\gamma^{(i-1)}|h,y)/q\left(\gamma^{(i-1)}|\gamma^*\right)}$$

Note that if it's easier to go from $\gamma^{(i-1)} \rightarrow \gamma^*$ than vice versa, (top q bigger than bottom q), we'll tend to stay put. A special case of this is *Random Walk* M-H, where the proposal density is centered at $\gamma^{(i-1)}$. When $q\left(\gamma^*|\gamma^{(i-1)}\right) = q\left(\gamma^{(i-1)}|\gamma^*\right)$, which will be the case for multivariate Normal or t with a covariance matrix that's independent of γ, the formula for α simplifies to

$$\alpha = \frac{p(\gamma^*|h,y)}{p(\gamma^{(i-1)}|h,y)}$$

We can just as easily move from one to the other, so we don't need the counterweighting for the draw density.

A few things to note about this:

1. We don't need p and q to be complete densities. Because they always are used in ratios, any common constants of integration will drop out.
2. Keep the calculations in logs as long as possible. In high dimensional problems, it's possible for the pieces to over- and underflow.

In implementing a chain, we need to decide upon the method and the proposal density. It's often easier to pick a reasonable proposal for Independence Chain M-H. With that, we're trying to pick from a density which comes fairly close to mimicking the shape of the posterior itself. If we have a respectable amount of data, we can use the asymptotic theory of non-linear regression to get an approximate density for γ as

$$\gamma \sim N\left(\hat{\gamma}, \hat{s}^2\left(\sum_{i=1}^{N} \frac{\partial f(X_i,\gamma)}{\partial \gamma}' \frac{\partial f(X_i,\gamma)}{\partial \gamma}\right)^{-1}\right) \tag{4.3}$$

which would be the output from a conventional non-linear least squares optimization. If the prior is fairly flat, we can get by with that. If not, we can combine this with the posterior using the Normal prior-Normal data to get an approximate Normal posterior. If the prior might very well dominate the data for some parameters, you may need to maximize the posterior (4.1) directly (you would maximize the log of this) and take a Normal approximation from that optimization.

One of these Normal approximations might work just fine, but it's often a good idea to "fatten" up the tails a bit. If we look at the desirability measure:

$$p(\gamma^*|h, y)/q\left(\gamma^*\right)$$

it can be quite high even for candidates with a fairly low value of p, if q is relatively lower than that. Normals have fairly thin tails, so if we manage to stumble across a tail γ where the p function is fatter than the Normal, we can stay at that one spot for quite a while. Having overly thick tails in the proposal density isn't as big a problem because, if the tails of q are too fat, there will be places to which we won't move easily, but we won't get stuck anywhere.

However you do it, it's important to monitor how frequently you accept new draws. If that's low (below 20%), you need a better proposal density. With Independence Chain M-H, it's hard for an acceptance ratio to be too *high*. After all, if you hit the posterior density spot on ($q = p$), $\alpha = 1$ for all draws.

Independence Chain is generally the better procedure if the posterior is fairly well-behaved, in the sense that it has a single maximum and generally falls off from there in all directions. If the posterior is multi-modal, or has directions in which it falls off very slowly, it won't work as well, since a single (convenient) density will have a hard time matching those features. That's where Random Walk M-H becomes useful. Picking the proposal density there, however, is quite a bit more challenging. You don't want to be drawing from a distribution based upon the overall spread of the posterior. If you do that, then you won't really have a chance to explore those odd shapes in the posterior. Once you locate one, if you use too wide a spread, the next draw will likely be rejected (since you're in a small region with a high posterior); if it's accepted, you'll probably be moving somewhere else, perhaps with a small probability of going back. There's quite a bit more tuning involved with this, since a low acceptance rate can mean that you need to spread your draw more (you've found one mode and can't move off of it because your draws don't span to the other one), or can mean that you need to spread your draw less. With Random Walk M-H, a high acceptance rate is also not good. It probably means you have too small a spread and are not moving very far. Somewhere in the range of .25 to.50 is generally best.

The same matrix as generated above in (4.3) can be useful to give some idea of the relative scales of the parameters. Taking the diagonal elements only (and possibly scaling down a bit) isn't a bad starting place.

4.2 Calculations

Our model is

$$y_i = f(X_i, \gamma) + \varepsilon_i, \varepsilon_i \sim N\left(0, h^{-1}\right) i.i.d.$$

For purposes of drawing from the posterior for γ, it's most convenient to have γ represented as a VECTOR. For writing the f function, however, it's usually most convenient to use descriptive variables (alpha, sigma, etc.). That makes it easier to change the representation, by adding or deleting parameters. Fortunately, it's relatively easy to switch between the two.

The example used is of a CES production function:

$$y_i = \gamma_1 + \left(\gamma_2 x_{1i}^{\gamma 4} + \gamma_3 x_{2i}^{\gamma 4}\right)^{1/\gamma_4} + \varepsilon_i$$

We'll set up the function using the descriptive variable names gamma1, etc.

```
nonlin(parmset=cesparms) gamma1 gamma2 gamma3 gamma4
*
compute gamma4=1.0
compute gamma1=0.0
compute gamma2=gamma3=0.5
*
frml ces y = gamma1+(gamma2*x1^gamma4+gamma3*x2^gamma4)^(1/gamma4)
nlls(frml=ces,parmset=cesparms)
```

To estimate this model successfully with **NLLS**, we need a positive guess value for γ_4 (otherwise the function can't even be computed) and for γ_2 or γ_3 (otherwise, γ_4 drops out of the function). Note that we used a named PARMSET. This is important so we can set the values of all the parameters from a VECTOR.

The setup for the prior is the same (other than numbers) as it is for the linear model:

```
compute s2prior=1.0/10.0
compute nuprior=12.0
*
compute [vector] bprior=||1.0,1.0,1.0,1.0||
compute [symm]   vprior=.25*%identity(4)
compute [symm]   hprior =inv(vprior)
```

We'll again start at the least-squares estimates and set up a SERIES[VECT] and SERIES to collect the draws. In this example, we're doing 25000 draws with 5000 burn-ins. We have a new addition to this: we also need to keep a count of the number of acceptances.

```
compute bdraw =%beta
compute s2draw=%seesq
compute nburn =5000
compute ndraws=25000
dec series[vect] bgibbs
dec series       hgibbs
gset bgibbs 1 ndraws = %zeros(%nreg,1)
set  hgibbs 1 ndraws = 0.0
compute accept=0
```

Example **4.1** does Random Walk M-H. This is using a Normal with mean zero and covariance matrix taken from **NLLS** as the increment in the proposal density. Since the covariance matrix is fixed across draws, we take its Choleski factorization outside the loop. (Given the level of effort in the other calculations, this doesn't matter that much, but it's a simple enough optimization).

```
compute fxx=%decomp(%seesq*%xx)
```

With the set of options used in **NLLS**, %XX does not have the scale factor for the residual variance incorporated; hence the scaling by %SEESQ. If you use ROBUSTERRORS on **NLLS**, you would skip that, because the relationship with the residuals is built into the calculation.

The draw for the precision of the residuals is pretty much the same as with the linear model, other than the need to calculate the sum of squared residuals. Note the use of the %PARMSPOKE function. This is why we assigned a named PARMSET to the parameters. See this chapter's *Tips and Tricks* (Section 4.3) for more on this function.

When you use a FRML like CES in an expression, it does not include the dependent variable – it computes the right-hand side of $y = f(X, \gamma)$, not the residuals.

```
compute %parmspoke(cesparms,bdraw)
sstats / (y-ces(t))^2>>rssbeta
compute rssplus=nuprior*s2prior+rssbeta
compute hdraw  =%ranchisqr(nuprior+%nobs)/rssplus
```

We now draw a candidate based upon the previous value of bdraw and evaluate the sum of squared residuals at that.

```
compute btest=bdraw+%ranmvnormal(fxx)
compute %parmspoke(cesparms,btest)
sstats / (y-ces(t))^2>>rsstest
```

We next compute the (logs) of the posterior densities and exp up to get the acceptance probability.

```
compute logptest=-.5*hdraw*rsstest-.5*%qform(hprior,btest-bprior)
compute logplast=-.5*hdraw*rssbeta-.5*%qform(hprior,bdraw-bprior)
compute alpha   =exp(logptest-logplast)
```

This is the code for the randomized acceptance. If we accept, we replace `bdraw` with the test vector and increment `accept`.

```
if alpha>1.0.or.%uniform(0.0,1.0)<alpha
   compute bdraw=btest,accept=accept+1
```

The remainder of the loop is the same as with the linear model, as is the post-processing for finding the mean and variance of the posterior. The one addition is that we need to display the fraction of acceptances:

```
disp "Acceptance Rate" accept/(ndraws+nburn+1.0)
```

Independence Chain M-H has the additional requirement that we compute the (kernel) of the proposal density. Fortunately, RATS is set up to handle that automatically. If you arrange the calculation so the draw is done in a single function call, you can fetch the log of the kernel of that previous draw using the function `%RANLOGKERNEL()`.

Since `%RANLOGKERNEL()` only has the correct value until the next set of draws, we need to keep track of it in case we hang onto the previous draw. If your initializer for the Gibbs sampler is the mean of your proposal density (from a Normal or t), you can start with the log kernel of 0.

The setup is the same until we get to:

```
compute fxx     =%decomp(%seesq*%xx)
compute nuxx    =10
compute logqlast=0.0
```

We're fattening the tails by using a t with 10 degrees of freedom. `logqlast` will be used to hold the q value for the last draw; we'll use `logqtest` for the q value for the test draw.

Most of the internals of the draw loop are the same. `compute btest` is different because it is centered at `%beta` rather than `bdraw`; we add the `compute logqtest` to grab the log kernel while it's available; `alpha` is computed with the more complicated expression and, if we accept, we hang onto the value of log q.

```
compute btest=%beta+%ranmvt(fxx,nuxx)
compute logqtest=%ranlogkernel()
compute %parmspoke(cesparms,btest)
sstats / (y-ces(t))^2>>rsstest
compute logptest=-.5*hdraw*rsstest-.5*%qform(hprior,btest-bprior)
compute logplast=-.5*hdraw*rssbeta-.5*%qform(hprior,bdraw-bprior)
compute alpha  =exp(logptest-logplast+logqlast-logqtest)
if alpha>1.0.or.%uniform(0.0,1.0)<alpha {
   compute bdraw=btest,accept=accept+1
   compute logqlast=logqtest
}
```

4.3 RATS Tips and Tricks

Using PARMSETS

A PARMSET is an object which organizes a set of parameters for non-linear estimation. In most cases where you do non-linear optimization (with instructions like **NLLS** or **MAXIMIZE**), you don't need a "named" parameter set. The **NONLIN** instruction without a PARMSET option creates the default unnamed parameter set which is what will be used on the non-linear estimation instruction, if *it* doesn't have a PARMSET option:

```
nonlin gamma1 gamma2 gamma3 gamma4
nlls(frml=ces)
```

The named PARMSET is useful when you need to be able to work with the parameters as a VECTOR. The functions for this are %PARMSPEEK(parmset) and %PARMSPOKE(parmset,vector). The first of these returns the current settings as a VECTOR; the second (which we used in this chapter), resets the parameters based upon the value in the VECTOR.

```
nonlin(parmset=cesparms) gamma1 gamma2 gamma3 gamma4
compute %parmspoke(cesparms,||0.0,0.5,0.5,1.0||)
compute gamma4=0.7
compute v=%parmspeek(cesparms)
```

Here, the %PARMSPOKE will make $\gamma_1 = 0, \gamma_2 = .5$, etc. At the end, V will be $[0, .5, .5, .7]$.

PARMSETS are also helpful if you would like to break a larger parameter set up into more manageable pieces. The following defines one PARMSET for the parameters governing the mean and another for the GARCH parameters. Those are "added" (combined) on the **MAXIMIZE** instruction to form the full parameter set for the estimation:

```
linreg us3mo
# constant us3mo{1}
frml(lastreg,vector=ar1) meanf
nonlin(parmset=meanparms) ar1
*
nonlin(parmset=garchparms) gamma alpha beta lambda
maximize(parmset=meanparms+garchparms) logl 2 *
```

The instruction FIND

In Example **4.2**, we do independence M-H using the coefficients and a (fattened up) asymptotic covariance matrix from an **NLLS** to provide the mean and covariance matrix for the proposal density. That works reasonably well since the prior is fairly loose; as a result, the posterior density (which is what we're trying to match) is dominated by the data.

Suppose instead that we have a more informative prior. The **NLLS** information might no longer be a good match for the posterior. What we can do instead is to find the *posterior mode*: the coefficient vector which maximizes the (log) posterior density. We can use that, plus the Hessian from that optimization (possibly with some fattening) to provide a proposal density which might better match the posterior. To compute the log posterior density, start by using the "evaluation" method for the instruction (see appendix A). For **NLLS**, this is METHOD=EVALUATE. This will give you the log likelihood. Add to that the log density for the prior. If the prior is in several independent pieces, you'll have to add each of those in. This code fragment estimates the posterior mode for the model used in this chapter. Note that you need to use the METHOD=BFGS and STDERRS options on the **FIND** if you want to get the Hessian. (The default estimation method for **FIND** is METHOD=SIMPLEX, which is derivative-free).

```
compute [vector] bprior=||1.0,1.0,1.0,0.5||
compute [symm]   vprior=.09*%identity(4)
compute [symm]   hprior =inv(vprior)
*
declare real logpostdensity
find(parmset=cesparms,method=bfgs,stderrs) max logpostdensity
   nlls(frml=ces,parmset=cesparms,method=evaluate,noprint)
   compute logpostdensity=%logl+$
     %logdensity(vprior,%parmspeek(cesparms))
end find
```

Example 4.1 Non-linear Regression: Random Walk MH

```
open data cesdata.xls
data(format=xls,org=columns) 1 123 y x1 x2
*
nonlin(parmset=cesparms) gamma1 gamma2 gamma3 gamma4
*
compute gamma4=1.0
compute gamma1=0.0
compute gamma2=gamma3=0.5
*
frml ces y = gamma1+(gamma2*x1^gamma4+gamma3*x2^gamma4)^(1.0/gamma4)
*
nlls(frml=ces,parmset=cesparms)
*
* Prior for variance.
*
compute s2prior=1.0/10.0
compute nuprior=12.0
*
* Prior mean and variance. Because the variance of the prior is now
* independent of the variance of the residuals, the <<hprior>> can be
* taken by just inverting the <<vprior>> without the degrees of freedom
* adjustment.
*
compute [vector] bprior=||1.0,1.0,1.0,1.0||
compute [symm]   vprior=.25*%identity(4)
compute [symm]   hprior =inv(vprior)
*
compute bdraw =%beta
compute s2draw=%seesq
*
compute nburn =5000
compute ndraws=25000
*
dec series[vect] bgibbs
dec series       hgibbs
gset bgibbs 1 ndraws = %zeros(%nreg,1)
set  hgibbs 1 ndraws = 0.0
compute accept=0
*
compute fxx=%decomp(%seesq*%xx)
do draw=-nburn,ndraws
   *
   *   Draw residual precision conditional on current bdraw
   *
   compute %parmspoke(cesparms,bdraw)
   sstats / (y-ces(t))^2>>rssbeta
   compute rssplus=nuprior*s2prior+rssbeta
   compute hdraw  =%ranchisqr(nuprior+%nobs)/rssplus
   *
   * Random walk MC
   *
   compute btest=bdraw+%ranmvnormal(fxx)
```

```
   compute %parmspoke(cesparms,btest)
   sstats / (y-ces(t))^2>>rsstest
   compute logptest=-.5*hdraw*rsstest-.5*%qform(hprior,btest-bprior)
   compute logplast=-.5*hdraw*rssbeta-.5*%qform(hprior,bdraw-bprior)
   compute alpha  =exp(logptest-logplast)
   if alpha>1.0.or.%uniform(0.0,1.0)<alpha
      compute bdraw=btest,accept=accept+1

   if draw<=0
      next
   *
   *   Do the bookkeeping here.
   *
   compute bgibbs(draw)=bdraw
   compute hgibbs(draw)=hdraw
end do draw
*
disp "Acceptance Rate" accept/(ndraws+nburn+1.0)
*
@mcmcpostproc(ndraws=ndraws,mean=bmean,stderrs=bstderrs,$
     cd=bcd,nse=bnse) bgibbs
report(action=define)
report(atrow=1,atcol=1,align=center) "Variable" "Coeff" "Std Error" $
     "NSE" "CD"
do i=1,%nreg
   report(row=new,atcol=1) "Gamma"+i bmean(i) bstderrs(i) bnse(i) bcd(i)
end do i
report(action=format,atcol=2,tocol=3,picture="*.###")
report(action=format,atcol=4,picture="*.##")
report(action=show)
```

Example 4.2 Non-linear Regression: Independence MH

```
open data cesdata.xls
data(format=xls,org=columns) 1 123 y x1 x2
*
nonlin(parmset=cesparms) gamma1 gamma2 gamma3 gamma4
*
compute gamma4=1.0
compute gamma1=0.0
compute gamma2=gamma3=0.5
*
frml ces y = gamma1+(gamma2*x1^gamma4+gamma3*x2^gamma4)^(1.0/gamma4)
*
nlls(frml=ces,parmset=cesparms)
*
* Prior for variance.
*
compute s2prior=1.0/10.0
compute nuprior=12.0
*
* Prior mean and variance. Because the variance of the prior is now
* independent of the variance of the residuals, the <<hprior>> can be
```

```
* taken by just inverting the <<vprior>> without the degrees of freedom
* adjustment.
*
compute [vector] bprior=||1.0,1.0,1.0,1.0||
compute [symm]   vprior=.25*%identity(4)
compute [symm]   hprior =inv(vprior)
*
compute bdraw =%beta
compute s2draw=%seesq
*
compute nburn =5000
compute ndraws=25000
*
dec series[vect] bgibbs
dec series       hgibbs
gset bgibbs 1 ndraws = %zeros(%nreg,1)
set  hgibbs 1 ndraws = 0.0
compute accept=0
*
* Draw from multivariate t centered at the NLLS estimates. In order to do
* this most conveniently, we set up a VECTOR into which we can put the
* draws from the standardized multivariate t.
*
compute fxx =%decomp(%seesq*%xx)
compute nuxx=10
dec vector tdraw(%nreg)
*
* Since we're starting at %BETA, the kernel of the proposal density is 1.
*
compute logqlast=0.0
*
do draw=-nburn,ndraws
   *
   *   Draw residual precision conditional on current bdraw
   *
   compute %parmspoke(cesparms,bdraw)
   sstats / (y-ces(t))^2>>rssbeta
   compute rssplus=nuprior*s2prior+rssbeta
   compute hdraw  =%ranchisqr(nuprior+%nobs)/rssplus
   *
   * Independence chain MC
   *
   compute btest=%beta+%ranmvt(fxx,nuxx)
   compute logqtest=%ranlogkernel()
   compute %parmspoke(cesparms,btest)
   sstats / (y-ces(t))^2>>rsstest
   compute logptest=-.5*hdraw*rsstest-.5*%qform(hprior,btest-bprior)
   compute logplast=-.5*hdraw*rssbeta-.5*%qform(hprior,bdraw-bprior)
   compute alpha  =exp(logptest-logplast+logqlast-logqtest)
   if alpha>1.0.or.%uniform(0.0,1.0)<alpha {
      compute bdraw=btest,accept=accept+1
      compute logqlast=logqtest
   }
   if draw<=0
```

```
      next
   *
   *   Do the bookkeeping here.
   *
   compute bgibbs(draw)=bdraw
   compute hgibbs(draw)=hdraw
end do draw
*
disp "Acceptance Rate" accept/(ndraws+nburn+1.0)
*
@mcmcpostproc(ndraws=ndraws,mean=bmean,stderrs=bstderrs,$
     cd=bcd,nse=bnse) bgibbs
report(action=define)
report(atrow=1,atcol=1,align=center) "Variable" "Coeff" "Std Error" $
     "NSE" "CD"
do i=1,%nreg
   report(row=new,atcol=1) "Gamma"+i bmean(i) bstderrs(i) bnse(i) bcd(i)
end do i
report(action=format,atcol=2,tocol=3,picture="*.###")
report(action=format,atcol=4,picture="*.##")
report(action=show)
```