# EXERCISE ON MCMC PARTICLE FILTER

In this exercise you will conduct inference on the model

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + \sigma_t \varepsilon_t \tag{1}$$

$$\log \sigma_t = \beta_0 + \beta_1 \log \sigma_{t-1} + \nu_t \tag{2}$$

$$\varepsilon_t, \nu_t \text{ i.i.d. across } t \tag{3}$$

$$\begin{bmatrix} \varepsilon_t \\ \nu_t \end{bmatrix} \sim N\left(0, \begin{bmatrix} 1 & 0 \\ 0 & \tau^2 \end{bmatrix}\right) . \tag{4}$$

The observables are the time series $\{y_t\}$. $\sigma_t$ is unobservable. Given two initial values of $y$ and a distribution for the initial values of $\sigma_t$, the particle filter, or possibly even simple sequential importance sampling, can be used to evaluate the likelihood.

Despite the fact that the particle filter gives only a more or less noisy measure of the likelihood, its likelihood values can be used in the accept-reject rule of an MCMC chain to provide accurate draws from the posterior on the parameters — here the $\alpha_i$'s, $\beta_i$'s, and $\tau^2$.

You are to carry out particle-filter-based MCMC posterior simulation on this model, using as $y_t$ the quarterly inflation rate, i.e. the quarterly change in the log price level, with the price data being the same as that used in the previous exercise on unemployment and inflation.

Assume stationarity of $\sigma_t$, so that parameter draws that imply non-stationarity are rejected. (Note that for valid MCMC, this has to be an MCMC rejection — repeat previous draw. Otherwise the region of the parameter space bordering the non-stationary region is undersampled by MCMC.) As the initial distribution for $\log \sigma_0$, therefore, you can use the steady-state unconditional distribution implied by the parameters. This ignores the fact that, in observing the initial $y$ values on which we are conditioning, we should in principle get some information about the initial $\sigma$.

As an initial value for the six free parameters, you can try

| a0 | a1 | tau | b0 | b1 | b2 |
|---|---|---|---|---|---|
| -0.75 | 0.87 | 0.29 | 0.00 | 0.53 | 0.33 |

These were reached after 40,000 iterations from a less-well-fitting starting point, but the algorithm was showing a lot of serial dependence, so even starting here you may need 100,000's of draws to get reasonable convergence, if it is even possible. Make a reasonable effort to get convergence, and be prepared to discuss the issue at the class presentation of the exercise.

For the jump distribution you can try the one in the supplied `pfexjump()` function. This builds in the expected negative correlation between constant and sum of right-hand-side coefficients in both equations, and tries to scale the jumps to a reasonable size. Note that the $\tau$ parameter is modeled as lognormal, so it will stay positive. (And note also that this preserves the $q(\theta' \mid \theta) = q(\theta \mid \theta')$ requirement on the jump density.)

---

*Date*: December 7, 2015.

There is working code for this problem on the course web site. As I've already noted, it generates quite serially correlated draws, so you may be able to improve it. If you write your own code, be particularly careful to handle the problem of "bad" parameter draws. Since we are using the steady-state distribution of the $\sigma_t$ process, we need $|\alpha_1| < 1$. And it is possible in rare instances to get a likelihood increment so small that the weight calculation generates NaN's. It can be discouraging to have the algorithm stop for this reason after tens of thousands of draws. The supplied code does try to handle these cases correctly. When you have converged results, or an unconverged result that is the best you can do in a reasonable time, discuss whether it implies a forecasting rule much different from what you would get by OLS estimates of an AR(2).

Try to use your results to generate a time series of expected values of $\sigma_t$, given the whole sample and conditioning on a parameter vector near the posterior mode. Note that the filtered $\sigma_t$ draws from the particle filter are at each date reflecting information in the data up to that date. To condition on the full sample, you will need to run one or more particle filters with large $N$, and then use the `ancestry` matrix.