ECON 634 Problem Set 7

Mingyang Li

December 10, 2017

1. • The transition function g is

   $$X_t = g(X_{t-1}, X_{t-2}, \varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}; \theta) = \rho_1 X_{t-1} + \rho_2 X_{t-2} + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + \varepsilon_t,$$

   where $\varepsilon_t \sim \mathrm{N}(0, \sigma_x^2)$.

   • The observation equation h is

   $$\begin{pmatrix} A_t \\ B_t \end{pmatrix} = \begin{pmatrix} h_A(X_t, v_t^A; \theta) \\ h_B(X_t, v_t^B; \theta) \end{pmatrix} = \begin{pmatrix} \exp(X_t + \nu_t^A) \\ \beta X_t^2 + \nu_t^B \end{pmatrix},$$

   where $\nu_t^A \sim \mathrm{N}(0, \sigma_A^2)$ and $\nu_t^B \sim \mathrm{N}(0, \sigma_B^2)$.

   • The state S_t is $S_t = \{X_{t-1}, X_{t-2}, \varepsilon_{t-1}, \varepsilon_{t-2}\}$.

   • The observables Y_t is $Y_t = \begin{pmatrix} A_t & B_t \end{pmatrix}'$.

   • The shocks W_t to state variables is $W_t = \begin{pmatrix} \varepsilon_t \end{pmatrix}$.

   • The shocks V_t to observables is $V_t = \begin{pmatrix} \nu_t^A & \nu_t^B \end{pmatrix}'$.

   • The parameter vector θ is

   $$\theta = \begin{pmatrix} \rho_1 & \rho_2 & \phi_1 & \phi_2 & \beta & \sigma_x & \sigma_A & \sigma_B \end{pmatrix}',$$

   where ρ_1, ρ_2, ϕ_1 and ϕ_2 are the ARMA(2,2) coefficients, β is the coefficient of X_t^2 in the observation equation h_B, σ_x is the standard deviation of the normal distribution to which ε_t belongs, and σ_A and σ_B are the standard deviations of the normal distributions to which the measurement errors ν_t^A and ν_t^B belong, respectively.

2. Please see Appendix B for the code.

3. Please see Appendix A for the code. The acceptance rate is about 1.7%. The distributions of parameters are as follows.

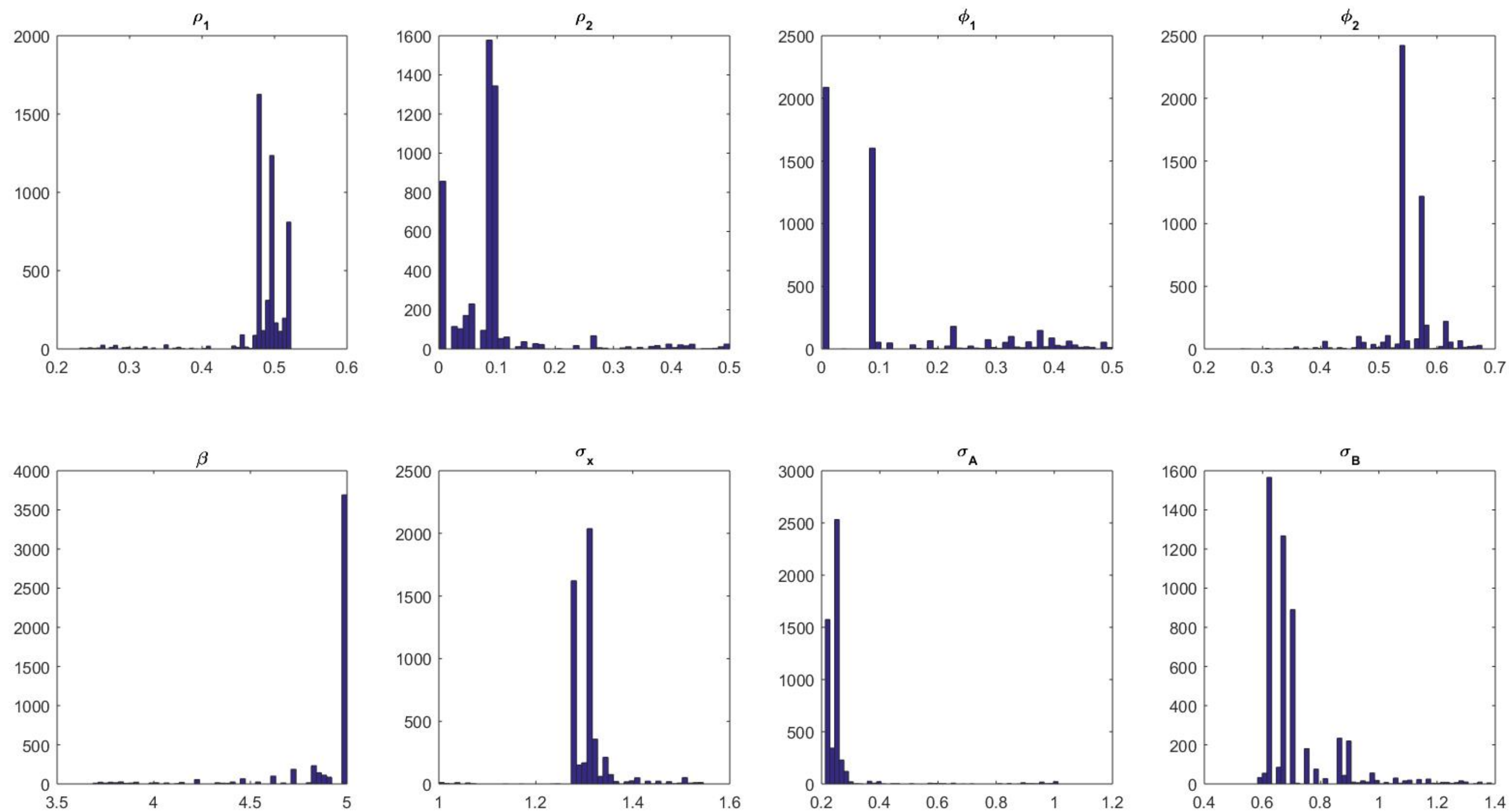

Figure 1: Posterior Distribution of θ

Appendices

A The main program PS7_McMc_sampler.m

```
%%%% Extremely simple sample application of a Metropolis hastings algorithm
%%%% (this script) and a particle filter to approximate the likelihood
%%%% function empirically ('PS7_model_llh.m').
%%%% Instructions: Load 'PS7_data.mat'. Specify priors, step sizes and number of
%%%% particles below, and run this script. Major credit goes to Prof. Kuhn.
clear, clc, close all
tic
% likelihood simulation parameters:
N = 1000; % number of particles
T = 400; % length of time series (given by data)

% data needs to be provided:
% data = cell2mat(struct2cell(load('PS7_data.mat')));
load data

% priors:
prior.rho_1 = @(x) normpdf(x, 0, 3);
prior.rho_2 = @(x) lognpdf(x, 0, 3);
prior.phi_1 = @(x) lognpdf(x, 0, 3);
prior.phi_2 = @(x) lognpdf(x, 0, 3);
prior.beta = @(x) 1;
prior.sigma_x = @(x) lognpdf(x, -1/2, 2);
prior.sigma_A = @(x) lognpdf(x, -1/2, 2);
prior.sigma_B = @(x) lognpdf(x, -1/2, 2);
prior.all = @(p) log(prior.rho_1(p(1))) + log(prior.rho_2(p(2))) + ...
    log(prior.phi_1(p(3))) + log(prior.phi_2(p(4))) + ...
    log(prior.beta(p(5))) + log(prior.sigma_x(p(6))) + ...
    log(prior.sigma_A(p(7))) + log(prior.sigma_B(p(8)));

% proposals according to random walk with parameter sd's:
prop_sig.rho_1 = 1;
prop_sig.rho_2 = 1;
prop_sig.phi_1 = 1;
prop_sig.phi_2 = 1;
prop_sig.beta = 1;
prop_sig.sigma_x = 1;
prop_sig.sigma_A = 1;
prop_sig.sigma_B = 1;
prop_sig.all = .04 * [prop_sig.rho_1 prop_sig.rho_2 prop_sig.phi_1 ...
```

```
    prop_sig.phi_2 prop_sig.sigma_x prop_sig.beta prop_sig.sigma_A ...
    prop_sig.sigma_B];

% initial values for parameters
init_params = [.5 .5 .5 .5 4 1 1 1];

% length of sample
M = 5000;
% M_burnin = 1000;
acc_rate = zeros(M, 8);

llhs = zeros(M, 1);
parameters = zeros(M, 8);
parameters(1, :) = init_params;

% evaluate model with initial parameters
log_prior = prior.all(parameters(1, :));
llh = PS7_model_llh(parameters(1, :), data, N, T);
llhs(1) = log_prior + llh;

% sample:
rng(1)
oneatatime = 0;
proposal_chance = log(rand(M, 1));
prop_step = randn(M, 8);
for m = 2:M
    % proposal draw:
    prop_param = parameters(m-1, :);
    vary_param = mod(m, 8) + 1;
    if oneatatime
        prop_param(vary_param) = prop_param(vary_param) + ...
            prop_step(m, vary_param) .* prop_sig.all(vary_param);
    else
        prop_param = prop_param + prop_step(m, :) .* prop_sig.all(vary_param);
    end

    % evaluate prior and model with proposal parameters:
    prop_prior = prior.all(prop_param);
    if prop_prior > -Inf % theoretically admissible proposal
        prop_llh = PS7_model_llh(prop_param, data, N, T);
        llhs(m) = prop_prior + prop_llh;
        if llhs(m) - llhs(m-1) > proposal_chance(m)
            accept = 1;
        else
            accept = 0;
        end
```

```
    else % reject proposal since disallowed by prior
        accept = 0;
    end

    % update parameters (or not)
    if accept
        parameters(m, :) = prop_param;
        acc_rate(m, :) = 1;
    else
        parameters(m, :) = parameters(m-1, :);
        llhs(m) = llhs(m-1);
    end

    waitbar(m / M)
end

toc

acc = sum(acc_rate) / M
str={'\rho_1', '\rho_2', '\phi_1', '\phi_2', ...
     '\beta', '\sigma_x', '\sigma_A','\sigma_B'};
for i=1:8
    subplot(2, 4, i)
    hist(parameters(:, i), 50);
    title(str{i});
end
```

B The function file PS7_model_llh.m

```
% Auxiliary function to 'PS7_McMc_sampler.m'. Credit goes to Prof. Kuhn.
function [LLH] = PS7_model_llh(params, data, N, T)
p.rho_1 = params(1);
p.rho_2 = params(2);
p.phi_1 = params(3);
p.phi_2 = params(4);
p.beta = params(5);
p.sigma_x = params(6);
p.sigma_A = params(7);
p.sigma_B = params(8);

T = min(T, length(data));
data_logA = log(data(:, 1));
data_B = data(:, 2);

% What's the long run distribution over state (X(t), X(t-1), X(t-2))?
```

```
rng(0)
lr_sim = 5000;
x_distn = zeros(lr_sim + 3, 1);
distn_shocks = p.sigma_x + randn(lr_sim + 3, 1);
for t = 3:lr_sim + 3
    x_distn(t) = p.rho_1 * x_distn(t-1) + p.rho_2 * x_distn(t-2) + ...
        p.phi_1 * distn_shocks(t-1) + p.phi_2 * distn_shocks(t-2);
end

particles = zeros(T, N, 6);
llhs = zeros(T, 1);
init_sample = randsample(lr_sim, N);
particles(1, :, 1) = x_distn(init_sample + 2);
particles(1, :, 2) = x_distn(init_sample + 1);
particles(1, :, 3) = x_distn(init_sample);
particles(1, :, 4) = distn_shocks(init_sample + 2);
particles(1, :, 5) = distn_shocks(init_sample + 1);
particles(1, :, 6) = distn_shocks(init_sample);
likelihoods = normpdf(data_logA(1), particles(1, :, 1), p.sigma_A) .* ...
    normpdf(data_B(1), p.beta * particles(1, :, 1) .^ 2, p.sigma_B);
llhs(1) = log(sum(likelihoods)) - log(N);

% predict, filter, update particles and collect the likelihood
for t = 2:T
    %%% Prediction:
    shocks = p.sigma_x * randn(1,N);
    particles(t, :, 1) = p.rho_1 * particles(t-1, :, 1) + p.rho_2 * ...
        particles(t-1, : , 2) + p.phi_1 * particles(t-1, :, 4) + p.phi_2...
        * particles(t-1, :, 5) + shocks;

    %%% Filtering:
    likelihoods = normpdf(data_logA(t), particles(t, :, 1), p.sigma_A) .* ...
        normpdf(data_B(t), p.beta * particles(t, :, 1) .^ 2, p.sigma_B);
    sampling_weights = exp(log(likelihoods) - log(sum(likelihoods)));
    if sum(likelihoods) == 0
        sampling_weights(:) = 1 / length(sampling_weights);
    end
    % store the log(mean likelihood)
    llhs(t) = log(sum(likelihoods)) - log(N);

    %%% Sampling:
    samples = randsample(N, N, true, sampling_weights);
    particles(t, :, :) = particles(t, samples, :);

end
LLH = sum(llhs);
```