ECON 6130 - Problem Set # 4

Julien Manuel Neves

December 15, 2017

Part (1)

For this problem set, we use the Adaptive Random-Walk Chain Metropolis-Hastings algorithm as defined in the notes with the following proposal density

$$q(\Theta^* \mid \Theta^{(s-1)}) = N(\Theta^{(s-1)}, \Sigma_{s-1})$$

where $\Sigma_{s-1} = 0.003 \cdot \Sigma_{\Theta s-1}$.

The prior distributions chosen for Θ are

$$\begin{aligned}
\sigma &\sim N(2, 0.1) \\
\beta &\sim B(.99, 0.05) \\
\phi &\sim N(4, 0.1) \\
\phi_\pi &\sim N(1.5, 0.1) \\
\phi_y &\sim N(0.125, 0.1) \\
\theta &\sim B(0.75, 0.1) \\
\alpha &\sim B\left(\frac{1}{3}, 0.1\right)
\end{aligned}$$

and improper distributions for the rest of Θ.

These prior distributions usually come from micro data and stylized facts. Technically, we could use uninformative distributions for every parameters, but since we have more information about certain specific parameters it is better to incorporate this into our MCMC algorithm.

Note that the chosen prior distributions specific parameter values comes from the class notes.

N.B. I believe Prof. Nimark's code for the prior distributions is incorrect. There is a mismatch between the distributions and Θ. This could explain the disparity between my results and the ones shown in class.

Part (2)

With our choice of parameters for the Adaptive Random-Walk Chain Metropolis-Hastings algorithm, the acceptance rate is equal to roughly 20%. This is roughly the sweet spot between too small steps/high acceptance rate and too big steps/low acceptance rate.

The raw MCMC for Θ is plotted in Fig. 1.

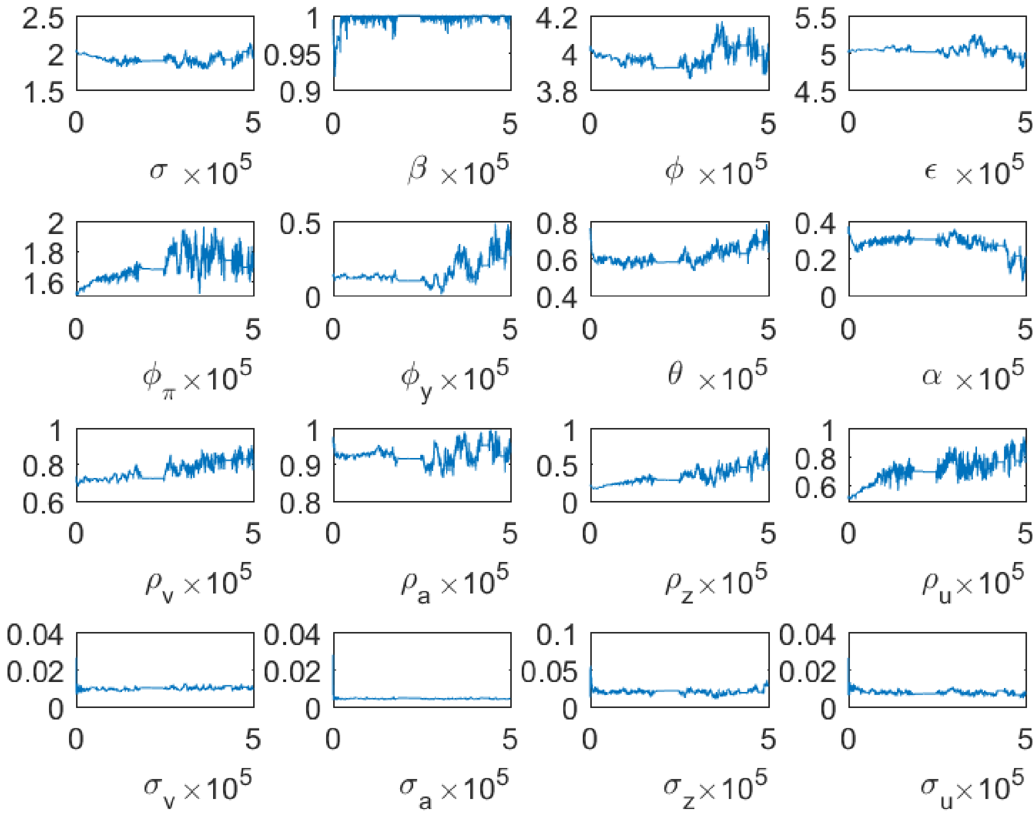

Figure 1: Raw MCMC of Θ

It usually a good idea to discard the first part of the Markov Chain since it takes some time before the algorithm reaches the stationary distribution. The raw MCMC for Θ with a burn-in sample of 20% is plotted in Fig. 2.

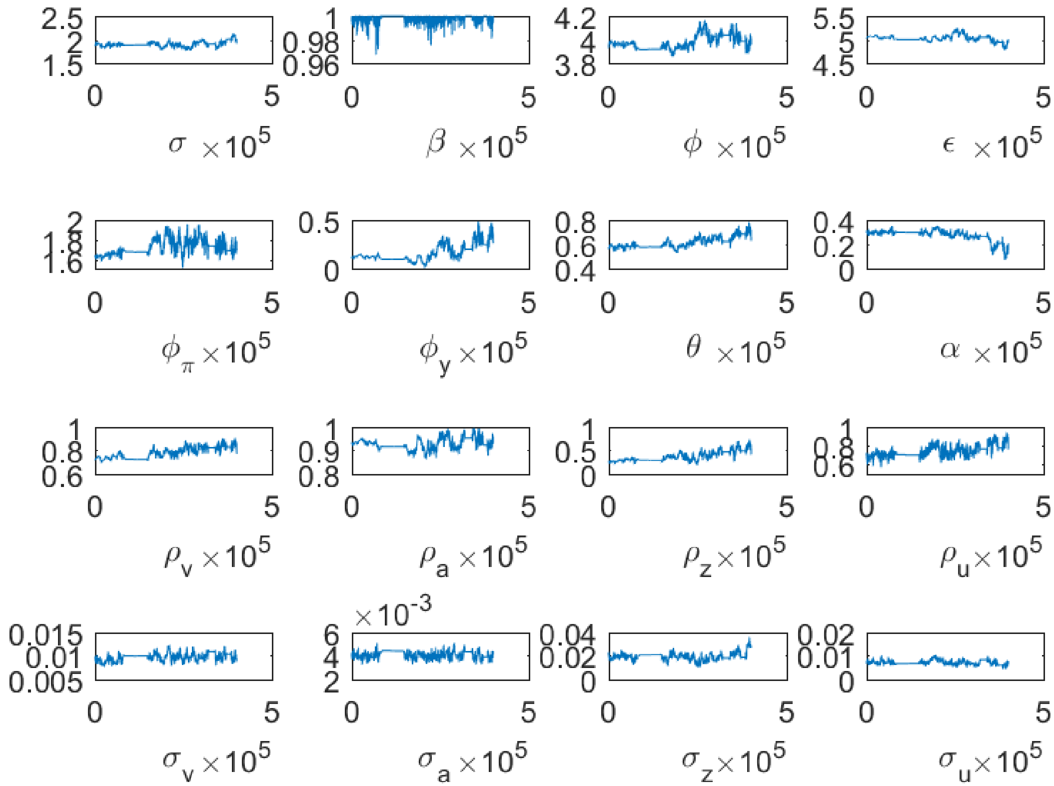

Figure 2: MCMC of Θ with discarded burn-in sample

Finally, to check for convergence, it is easier to look at the cumulative mean and see if it stabilizes. The cumulative mean of the MCMC for Θ is plotted in Fig. 3. It is straightforward to see that the cumulative mean stabilizes for every single component of Θ, hence we have convergence.

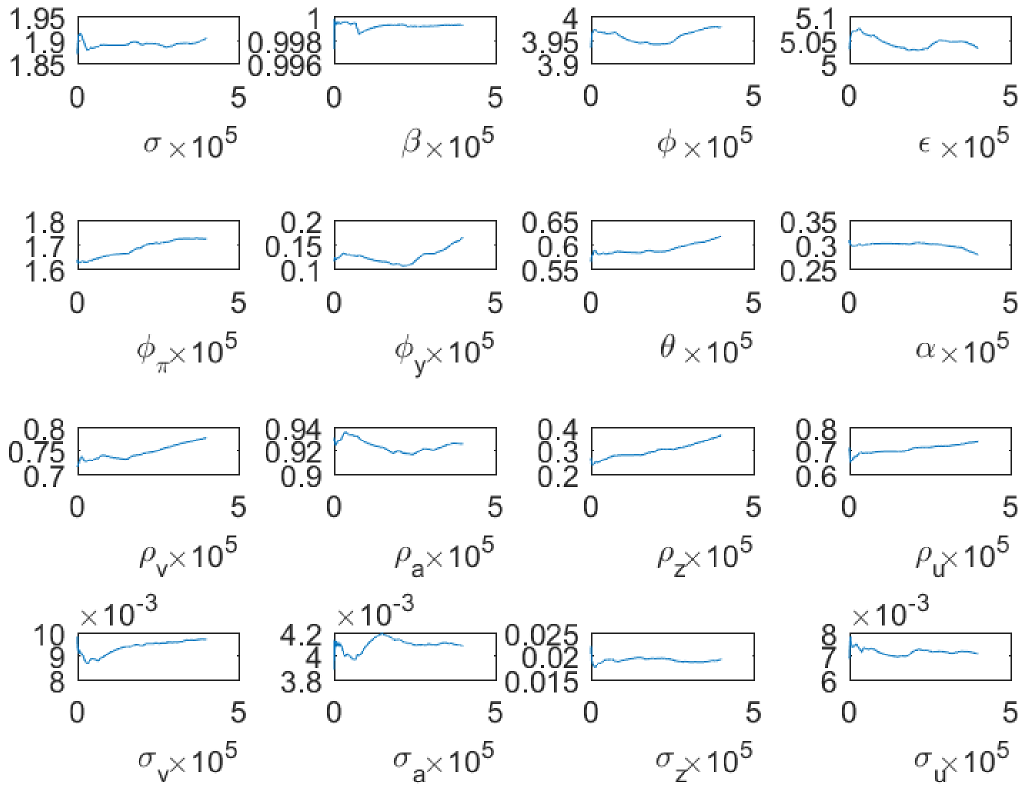

Figure 3: Cumulative mean of Θ

Part (3)

The posterior distribution of Θ is plotted in Fig. 4.

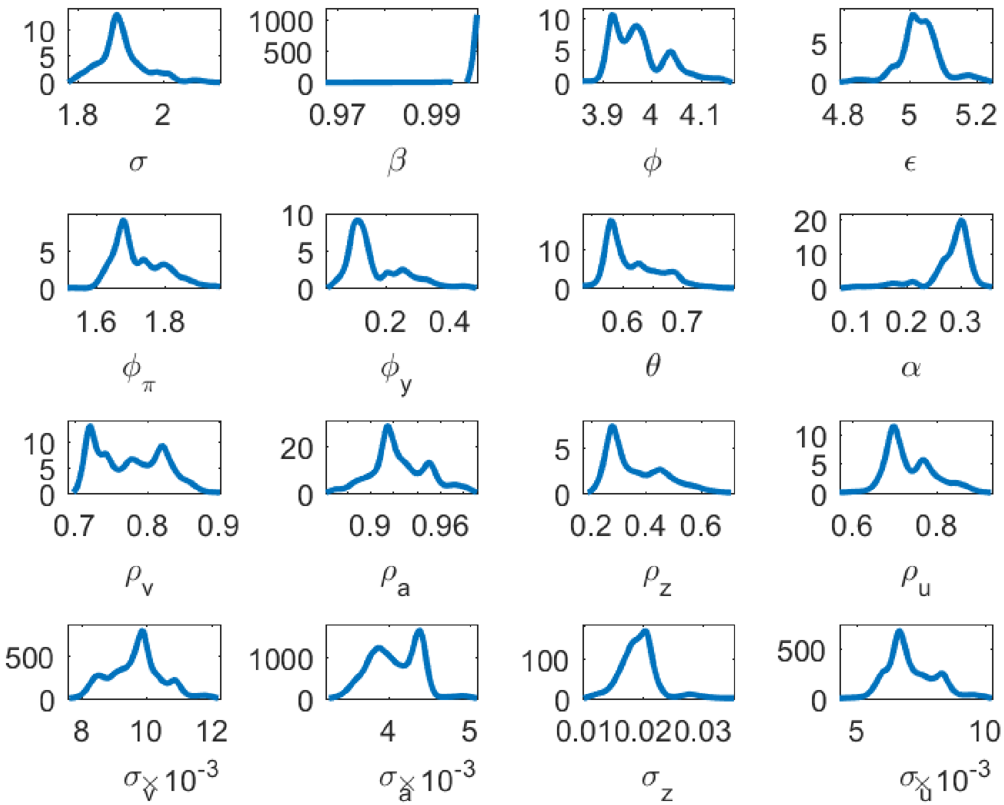

Figure 4: Posterior distribution of Θ

Part (4)

The impulse responses are plotted in Fig.5, Fig.6, Fig.7 and Fig.8.

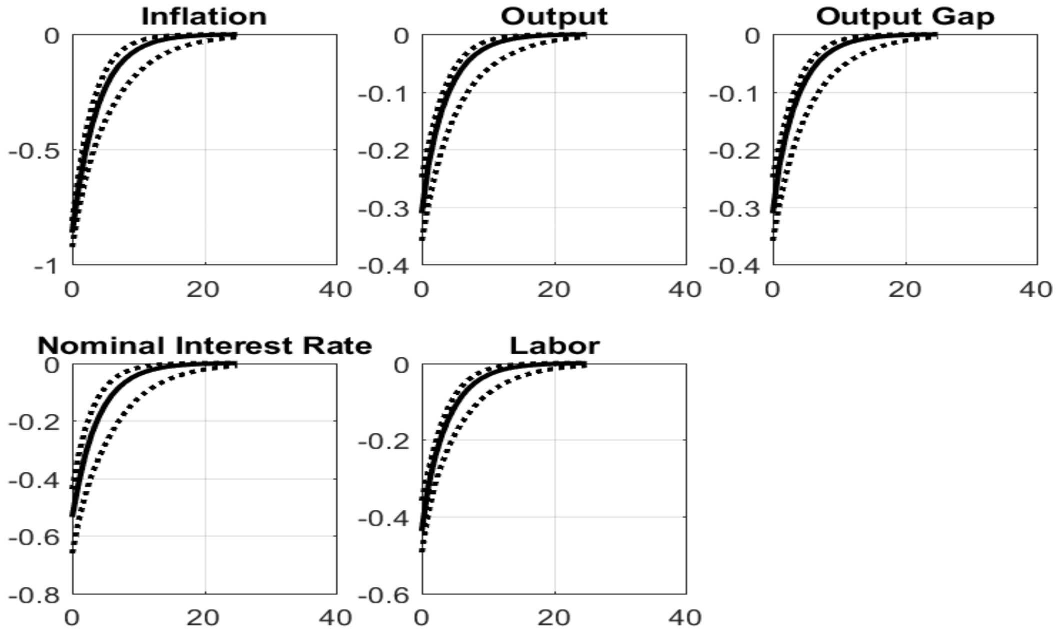

Figure 5: Impulse response to monetary shocks

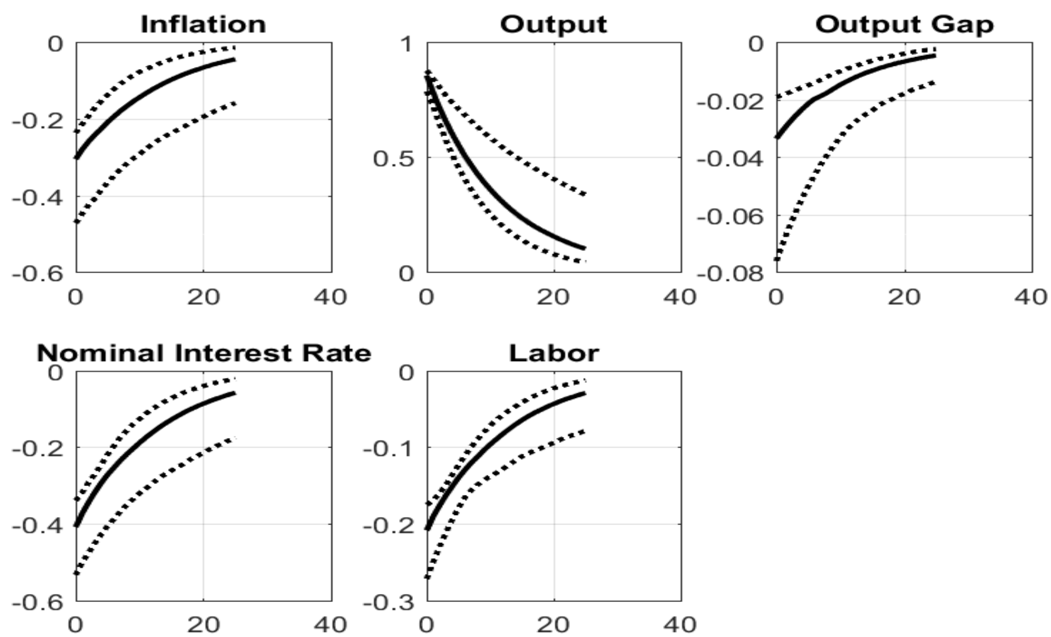

Figure 6: Impulse response to productivity shocks

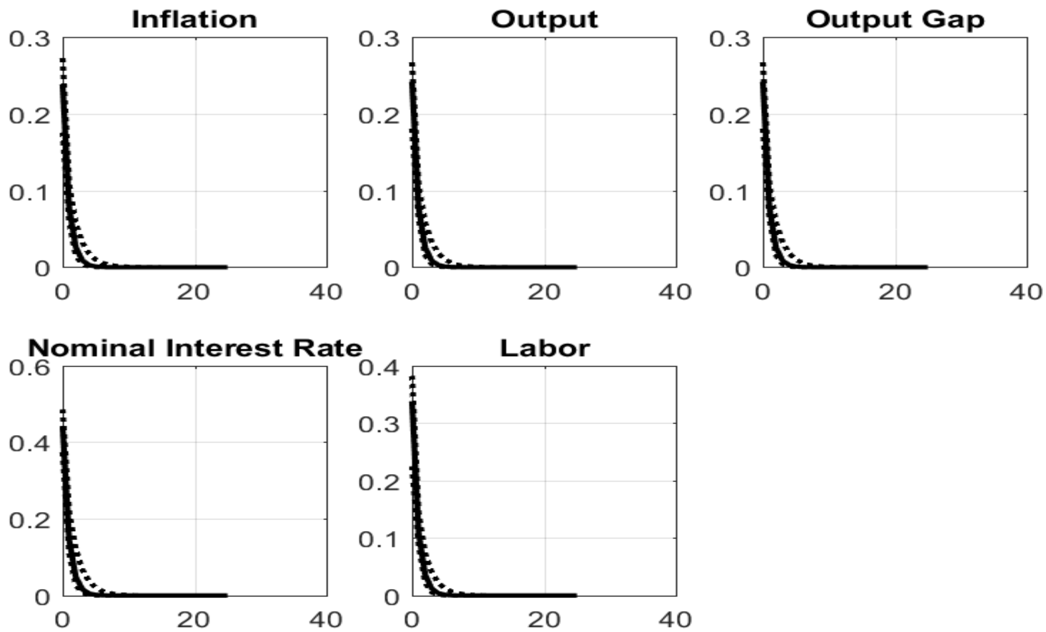

Figure 7: Impulse response to demand shocks

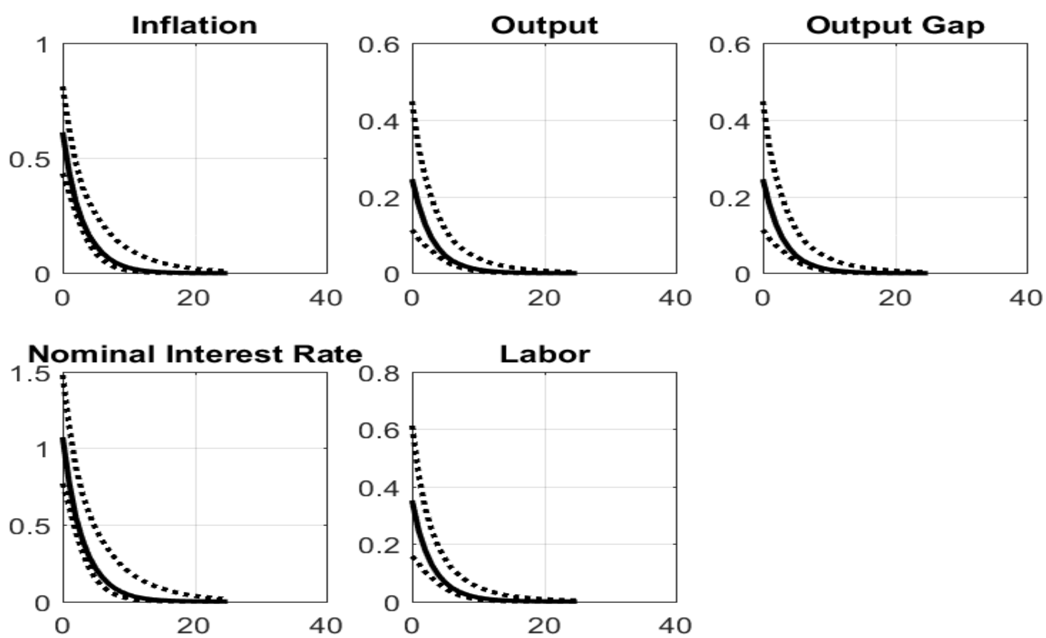

Figure 8: Impulse response to cost-push shocks

Part (5)

The distribution of the variance decomposition of each observable variable are plotted in Fig.9, Fig.10, Fig.11 and Fig.12.

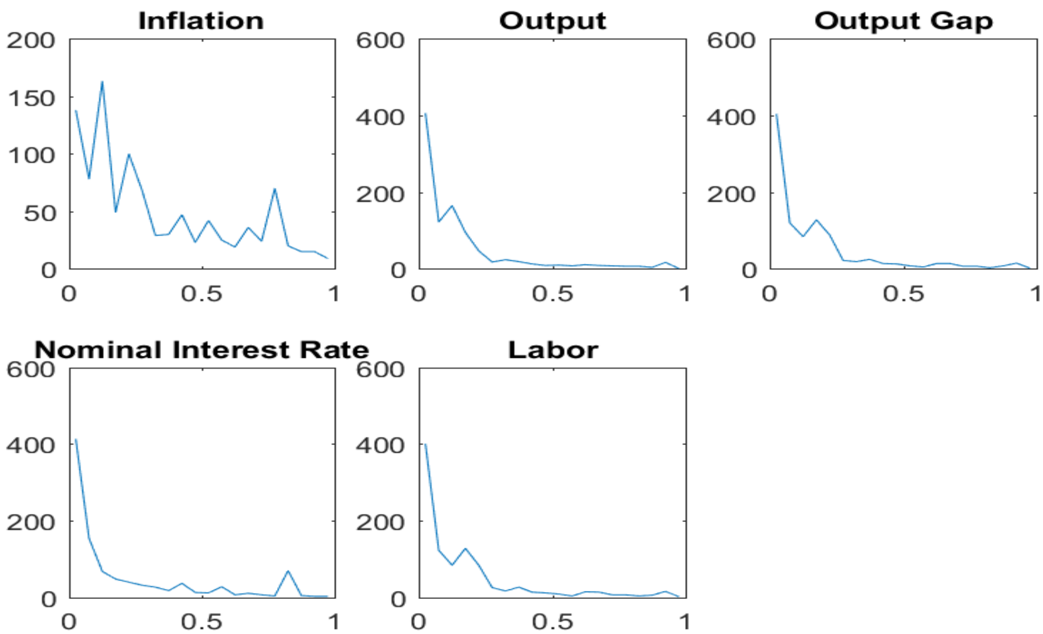

Figure 9: Variance decomposition - Monetary shocks

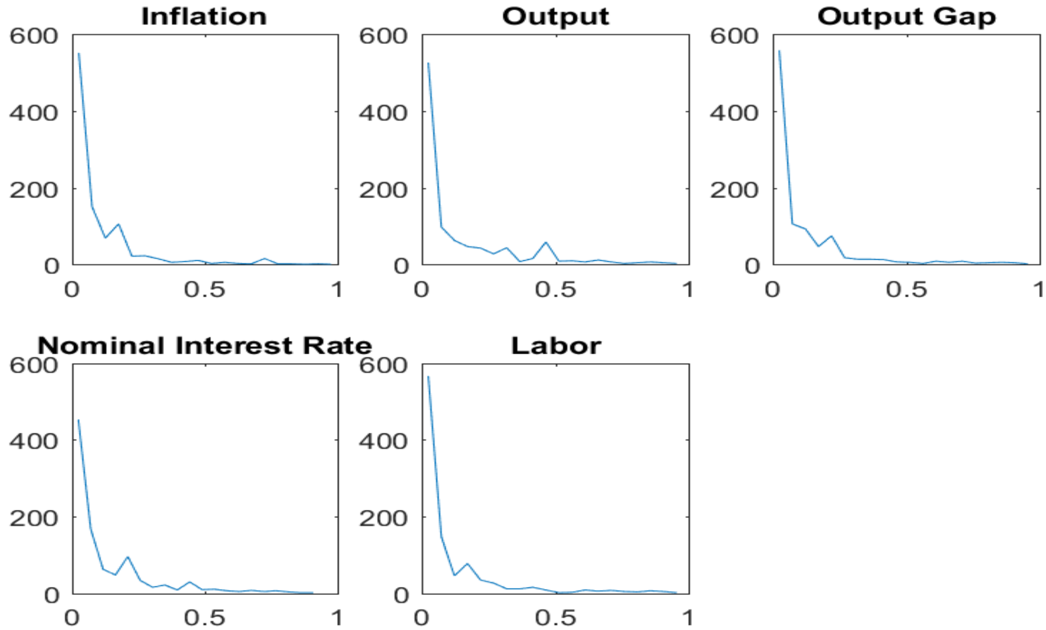

Figure 10: Variance decomposition - Productivity shocks

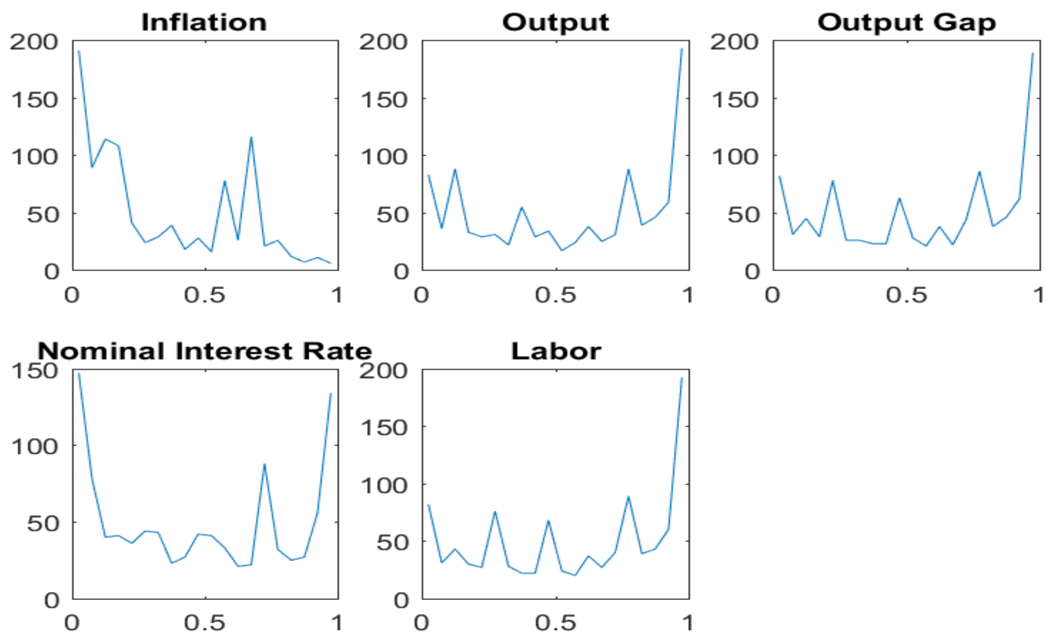

Figure 11: Variance decomposition - Demand shocks

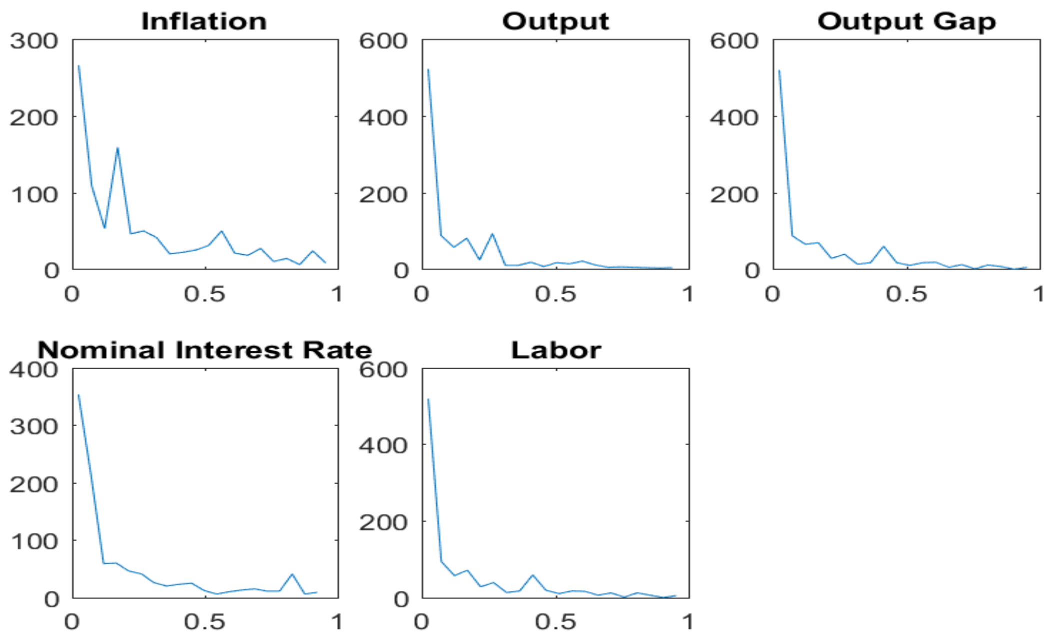

Figure 12: Variance decomposition - Cost-push shocks

Part (6)

Using a sample of 1000 points from our Markov Chain, we compute the posterior probability that labor inputs fall for 8 consecutive quarters after a positive productivity shock

$$P(\text{labor inputs in 8 consecutive quarters} < 0 \mid Z) = 1$$

i.e. it always falls.

Code

Main (main.m)

```
%% New Keynesian Model - Simulated Annealing
% Based of Kris Nimark's code.
% Modified by Julien Neves

%% Housekeeping
close all;
warning off all;

%% Part (1)
global Z
```

```
% Set handle for FRED data
url = 'https://fred.stlouisfed.org/';
c = fred(url);

% Set dates for sample period
startdate = '01/01/1983';
enddate = '12/01/2007';

CPI = fetch(c,'USACPIALLQINMEI','09/01/1982',enddate); % fetch CPI
    from FRED
GDP = fetch(c,'GDPC1',startdate,enddate); % fetch GDP from FRED
INT = fetch(c,'IRSTFR01USQ156N',startdate,enddate); % fetch rate
    from FRED
UNR = fetch(c,'LRUN64TTUSQ156S',startdate,enddate); % fetch rate
    from FRED

pi_data = diff(log(CPI.Data(:,2))); % log[price(t)] - log[price(t
    -1)]
gdp_data = log(GDP.Data(:,2)); % log of GDP
i_data = log(1+INT.Data(:,2)/100); % log of nominal interest rate
n_data = log(1-UNR.Data(:,2)/100); % log of employment rate

[~, pi_data] = hpfilter(pi_data,1600); % extract cyclical
    component of pi
[~,gdp_data] = hpfilter(gdp_data,1600); % extract cyclical
    component of y
[~,i_data] = hpfilter(i_data,1600); % extract cyclical component
    of i
[~,n_data] = hpfilter(n_data,1600); % extract cyclical component
    of n

% Combine data
Z = [pi_data';gdp_data'; i_data'; n_data'];

% Calibration
sigma     = 2; % CRRA parameter.
beta      = 0.99; % discount factor
phi       = 4; % inverse of elastiicity of labor supply
eps       = 5; % elasticity of substitution between goods i and j
phi_pi    = 1.5; % taylor rule parameter
phi_y     = 0.125; % taylor rule parameter
theta     = 0.75; % degree of price stickiness
alpha     = 0.33; % production function parameter
rho_v     = 0.7; % persistence parameter
rho_a     = 0.95; % persistence parameter
```

```
rho_z     = 0.2; % persistence parameter
rho_u     = 0.5; % persistence parameter
sigma_v   = 0.01; % standard deviation
sigma_a   = 0.008; % standard deviation
sigma_z   = 0.008; % standard deviation
sigma_u   = 0.01; % standard deviation

% Set starting value
THETA = [sigma; beta; phi; eps; phi_pi; phi_y; theta; alpha;
    rho_v; rho_a; rho_z; rho_u; sigma_v; sigma_a; sigma_z; sigma_u];

LB= [0  0 1  1  1 0 0 0 0 0 0 0 0 0 0 0]'; % lower bound
UB= [10 1 10 25 5 5 1 1 1 1 1 1 10 10 10 10]'; % upper bound

x=THETA;

%Number of draws
S=5e5;%Number of draws in MCMC
burnin=0.2*S;% Fraction of MCMC disregarded as "burnin sample"
J=1000;%Number of draws from MCMC used to simulate posterior functions of theta
epseye=1e-6; %scale up or down to tune acceptance ratio
adaptive=1; %set to 1 to use adaptive proposal density

% -----------------------------------------------------------------------------------------
% Initializes the MH algorithm
% -----------------------------------------------------------------------------------------

%Initializes the proposal variance.
%Set up so that the first candidate draw is always accepted
lpostdraw = -9e+200;
bdraw=x;%Initial draw
vscale=diag(abs(theta))*1d-4+1e-5*eye(length(x));%Initial covariacne of proposal density (not really important)

thetaMCMC=zeros(length(x),S);%Store all draws in thetaMCMC

%Matrices that keep track of switches and drwas outside LB and UB
OutsideProp=zeros(S,1);%Keep track of proprtion of draws outside parmater bounds
```

```
SwitchesProp=zeros(S,1);%Keep track of proprtion of switches (accepted draws)
%Number of draws outside parameter boundaries
q=0;
%Number of switches (acceptances)
pswitch=0;
%Iteration counter
iter=0;

%-------------------------------------------------------------------------------------------
% MH algorithm starts here
%-------------------------------------------------------------------------------------------

tic
for iter=1:S
    % Draw from proposal density Theta*_{t+1} ~ N(Theta_{t},vscale)
    bcan = bdraw + norm_rnd(vscale);

    if min(bcan > LB)==1 && min(bcan < UB)==1
        lpostcan = log_prior_DSGE(bcan)+LLDSGE(bcan);
        laccprob = lpostcan-lpostdraw;
    else
        laccprob=-9e+200;%assign very low value to reject
        q=q+1;
    end

    %Accept candidate draw with log prob = laccprob, else keep old draw
    if log(rand)<laccprob
        lpostdraw=lpostcan;
        bdraw=bcan;
        pswitch=pswitch+1;
    end

    thetaMCMC(:,iter)=bdraw; %add accepted new draw (or the previous value if candidate was rejected) to chain

    OutsideProp(iter)=q/iter;
    SwitchesProp(iter)=pswitch/iter;

```

```
    if mod(iter,10000)==0 && iter > 10000 %do this every 10000 draws if iter > 10000
        disp(['iter: ',num2str(iter)]);
        disp(['acceptance rate: ',num2str(SwitchesProp(iter))]);
        if adaptive==1
            vscale=3d-3*cov(thetaMCMC(:,1000:iter)')+1e-10*eye(16);%update the covariance of proposal density (the second component is just to avoid possible singualrites o
        end
    end
end
toc

thetaMCMC_raw=thetaMCMC;
thetaMCMC=thetaMCMC(:,burnin:end); % disregard burnin sample and keep only evey 100th draw

%% Part (2)
% Plot raw data
plotraw(thetaMCMC_raw,1);
plotraw(thetaMCMC,0);

% Plot cumulative mean
convcheck(thetaMCMC);

%% Part (3)
plotpost(thetaMCMC,0);
close all

%% Impulse resonses / Variance decomposition
time = 0:25;
ra = max(size(thetaMCMC));
FF = ceil(ra.*rand(J,1));

upper=ceil(J*.95);
med=ceil(J*.5);
lower=ceil(J*.05);

% Compute impulse responses
for j = 1:J
theta=thetaMCMC(:,FF(j,1));

% Compute Neq Keynesian Model with estimated theta
[A, C, D ,~, T] = NKBC_model(theta, 'impulse');
```

```
col = T;    % start impulse matrix
sigma = D*dlyap(A,C*C')*D';

for s=1:length(time)
    resp(:,:,s)=D*col; % compute observations
    col=A*col;  % compute next period states
end

for i = 1:4
    % Extract impulse responses for observations
    resp_pi(:,i,j)=squeeze(resp(1,i,:));
    resp_y(:,i,j)=squeeze(resp(2,i,:));
    resp_yg(:,i,j)=squeeze(resp(3,i,:));
    resp_i(:,i,j)=squeeze(resp(4,i,:));
    resp_n(:,i,j)=squeeze(resp(5,i,:));

    sigma_i = D*dlyap(A,C(:,i)*C(:,i)')*D';
    var_decomp(:,i,j)=diag(sigma_i)./diag(sigma);
end
end

for i = 1:4
    % Extract impulse responses for observations
    resp_pi(:,i,:)=sort(resp_pi(:,i,:),3);
    resp_y(:,i,:)=sort(resp_y(:,i,:),3);
    resp_yg(:,i,:)=sort(resp_yg(:,i,:),3);
    resp_i(:,i,:)=sort(resp_i(:,i,:),3);
    resp_n(:,i,:)=sort(resp_n(:,i,:),3);

    % Plot Impulse Responses
    figure(i)
    subplot(2,3,1);
    plot(time,resp_pi(:,i,upper),':','color','black','LineWidth'
        ,2); hold on;
    plot(time,resp_pi(:,i,med),'color','black','LineWidth',2);
        hold on;
    plot(time,resp_pi(:,i,lower),':','color','black','LineWidth'
        ,2); hold on;
    title('Inflation'); grid on;
    subplot(2,3,2);
    plot(time,resp_y(:,i,upper),':','color','black','LineWidth',2)
        ; hold on;
    plot(time,resp_y(:,i,med),'color','black','LineWidth',2); hold
         on;
```

```
        plot(time,resp_y(:,i,lower),':','color','black','LineWidth',2); hold on;
        title('Output'); grid on;
        subplot(2,3,3);
        plot(time,resp_yg(:,i,upper),':','color','black','LineWidth',2); hold on;
        plot(time,resp_yg(:,i,med),'color','black','LineWidth',2); hold on;
        plot(time,resp_yg(:,i,lower),':','color','black','LineWidth',2); hold on;
        title('Output Gap'); grid on;
        subplot(2,3,4);
        plot(time,resp_i(:,i,upper),':','color','black','LineWidth',2); hold on;
        plot(time,resp_i(:,i,med),'color','black','LineWidth',2); hold on;
        plot(time,resp_i(:,i,lower),':','color','black','LineWidth',2); hold on;
        title('Nominal Interest Rate'); grid on;
        subplot(2,3,5);
        plot(time,resp_n(:,i,upper),':','color','black','LineWidth',2); hold on;
        plot(time,resp_n(:,i,med),'color','black','LineWidth',2); hold on;
        plot(time,resp_n(:,i,lower),':','color','black','LineWidth',2); hold on;
        title('Labor'); grid on;
        hold off;

        figure(i+4)
        subplot(2,3,1); [y, x] = hist(squeeze(var_decomp(1,i,:)),20); plot(x,y); axis([0 1 0 500 ]); axis 'auto y'; title('Inflation');
        subplot(2,3,2); [y, x] = hist(squeeze(var_decomp(2,i,:)),20); plot(x,y); axis([0 1 0 500 ]); axis 'auto y'; title('Output');
        subplot(2,3,3); [y, x] = hist(squeeze(var_decomp(3,i,:)),20); plot(x,y); axis([0 1 0 500 ]); axis 'auto y'; title('Output Gap');
        subplot(2,3,4); [y, x] = hist(squeeze(var_decomp(4,i,:)),20); plot(x,y); axis([0 1 0 500 ]); axis 'auto y'; title('Nominal Interest Rate');
        subplot(2,3,5); [y, x] = hist(squeeze(var_decomp(5,i,:)),20); plot(x,y); axis([0 1 0 500 ]); axis 'auto y'; title('Labor');
```

```
end


%% Part (4)
% Print Impulse Responses - Monetary Shock
figure(1)
print('impulse_monetary','-dpng')
% Print Impulse Responses - Productivity Shock
figure(2)
print('impulse_prod','-dpng')
% Print Impulse Responses - Demand Shock
figure(3)
print('impulse_demand','-dpng')
% Print Impulse Responses - Cost-push Shock
figure(4)
print('impulse_cost','-dpng')


%% Part (5)
% Print Variance Decomposition - Monetary Shock
figure(5)
print('var_monetary','-dpng')
% Print Variance Decomposition - Productivity Shock
figure(6)
print('var_prod','-dpng')
% Print Variance Decomposition - Demand Shock
figure(7)
print('var_demand','-dpng')
% Print Variance Decomposition - Cost-push Shock
figure(8)
print('var_cost','-dpng')


%% Part (6)
disp(['Probability(Labor<0 in period 8): ', num2str(mean(resp_n
    (8,2,:)<0))]);
```

New Keynesian Model (nkbc_model.m)

```
function [A, C, D, eu, R] = NKBC_model( THETA, type)
%NKBC_model New Keynesian Model with cost-push shocks
%    State space model:
%    X(t) = A*X(t-1) + C*eps(t)
%    Z(t) = D*X(t)
%    By Julien Neves
```

```
%% Matrix A and C
% Calibration
sigma     = THETA(1); % CRRA parameter.
beta      = THETA(2); % discount factor
phi       = THETA(3); % inverse of elastiicity of labor supply
eps       = THETA(4); % elasticity of substitution between goods i and j
phi_pi    = THETA(5); % taylor rule parameter
phi_y     = THETA(6); % taylor rule parameter
theta     = THETA(7); % degree of price stickiness
alpha     = THETA(8); % production function parameter
rho_v     = THETA(9); % persistence parameter
rho_a     = THETA(10); % persistence parameter
rho_z     = THETA(11); % persistence parameter
rho_u     = THETA(12); % persistence parameter
sigma_v   = THETA(13); % standard deviation
sigma_a   = THETA(14); % standard deviation
sigma_z   = THETA(15); % standard deviation
sigma_u   = THETA(16); % standard deviation

% Compute the coefficients
rho = -log(beta);
lambda = (1-theta)*(1-beta*theta)*(1-alpha)/(theta*(1-alpha+alpha*eps));
kappa = lambda*(sigma + (phi+alpha)/(1-alpha));
psi_ya = (1+phi)/(sigma*(1-alpha)+phi+alpha);

% State: 'y','x'; 'pi'; 'r^e'; 'i'; 'v'; 'a'; 'z'; 'u'; 'E(x)'; 'E(pi)'
Gamma0 = [kappa -kappa 0 0 0 0 0 0 -1 0 0;
    0 -kappa 1 0 0 0 0 0 -1 0 -beta;
    0 1 0 -1/sigma 1/sigma 0 0 0 0 -1 -1/sigma;
    -phi_y 0 -phi_pi 0 1 -1 -phi_y*psi_ya 0 0 0 0;
    0 0 0 1 0 0 sigma*(1-rho_a)*psi_ya -(1-rho_z) 0 0 0;
    0 0 0 0 0 1 0 0 0 0 0;
    0 0 0 0 0 0 1 0 0 0 0;
    0 0 0 0 0 0 0 1 0 0 0;
    0 0 0 0 0 0 0 0 1 0 0;
    0 1 0 0 0 0 0 0 0 0 0;
    0 0 1 0 0 0 0 0 0 0 0];

Gamma1 = [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 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_v 0 0 0 0 0;
            0 0 0 0 0 0 rho_a 0 0 0 0;
            0 0 0 0 0 0 0 rho_z 0 0 0;
            0 0 0 0 0 0 0 0 rho_u 0 0;
            0 0 0 0 0 0 0 0 0 1 0;
            0 0 0 0 0 0 0 0 0 0 1];

Psi = [ 0 0 0 0 0 1 0 0 0 0 0;
        0 0 0 0 0 0 1 0 0 0 0;
        0 0 0 0 0 0 0 1 0 0 0;
        0 0 0 0 0 0 0 0 1 0 0;]';

Pi = [0 0 0 0 0 0 0 0 0 1 0;
      0 0 0 0 0 0 0 0 0 0 1]';

Cons = [0 0 0 0 0 0 0 0 0 0 0]';

% Solve New Keynesian Model
[A,~,R,~,~,~,~,eu,~]=gensys(Gamma0,Gamma1,Cons,Psi,Pi);

C = R*[sigma_v 0 0 0;
       0 sigma_a 0 0;
       0 0 sigma_z 0;
       0 0 0 sigma_u];

%% Matrix D
% Set up measurement matrix Z(t)=D*S(t)
if strcmp(type, 'data')
    % State: 'y','x'; 'pi'; 'r^e'; 'i'; 'v'; 'a'; 'z'; 'u'; 'E(x)'; 'E(pi)'
    D = [0 0 1 0 0 0 0 0 0 0 0; % inflation
         1 0 0 0 0 0 psi_ya 0 0 0 0; % output
         0 0 0 0 1 0 0 0 0 0 0;  % nominal interest rate
         1/(1-alpha) 0 0 0 0 0 -(1-psi_ya)/(1-alpha) 0 0 0 0]; % labor

elseif strcmp(type, 'impulse')
    % State: 'y','x'; 'pi'; 'r^e'; 'i'; 'v'; 'a'; 'z'; 'u'; 'E(x)'; 'E(pi)'
    D = [0 0 1 0 0 0 0 0 0 0 0; % inflation
         1 0 0 0 0 0 psi_ya 0 0 0 0; % output
         1 0 0 0 0 0 0 0 0 0 0;  % output gap
         0 0 0 0 1 0 0 0 0 0 0;  % nominal interest rate
```

```
        1/(1-alpha) 0 0 0 0 0 -(1-psi_ya)/(1-alpha) 0 0 0 0]; % labor
else
    warning('Measurement matrix missing')
end

end
```

Loglikelihood - Data (LLDSGE.m)

```
function [ logL ] = LLDSGE( THETA )
%loglikelihood_DSGE Likelihood function for State space model
%   State space model:
%   X(t) = A*X(t-1) + C*eps(t)
%   Z(t) = D*X(t)
%   By Julien Neves
global Z

% Solve New Keynesian Model
[A,C,D,eu] = NKBC_model( THETA, 'data');

% Set starting values
X0 = zeros(size(A,2),1);     % set starting X
P0 = dlyap(A,C*C'); % set starting for the variance

% Compute the Kalman Filter
[~, ~, ~, Z_tilde, Omega] = kfilter(Z, A, C, D, 0, X0, P0 );

% Initialize loglikelihood
T = length(Z);
logL = -T/2*log(2*pi)*size(Z,1);

for t = 1:T
    % Update loglikelihood
    logL = logL - 1/2*log(det(Omega{t})) - 1/2* Z_tilde{t}'/Omega{t}* Z_tilde{t};
end
% if imaginary parts or not identified model set likelihood to small value
if (imag(logL)~=0)||(min(eu)==0)
    logL=-9e+200;
end

end
```

Loglikelihood - Prior (log_prior_DSGEmodel.m)

```
function LP=log_prior_DSGE(theta)

LP=0;

% %(1) sigma  - risk aversion
pmean=2; pstdd=.1;
LP = LP + lpdfNormal(theta(1),pmean,pstdd);

%(2) beta - discount factor
pmean=0.99; pstdd=0.05;
a = (1-pmean)*pmean^2/pstdd^2 - pmean;
b = a*(1/pmean - 1);
LP = LP + lpdfBeta(theta(2),a,b);

% %(3) phi - inverse of elasticity of labor supply
pmean=4; pstdd=.1;
LP = LP + lpdfNormal(theta(3),pmean,pstdd);

%(4) phi_pi - coefficient on inflation in Taylor rule
pmean=1.5; pstdd=0.1;
LP = LP + lpdfNormal(theta(5),pmean,pstdd) ;

%(5) phi_y - coefficient on output in Taylor rule
pmean=.125; pstdd=0.1;
LP = LP + lpdfNormal(theta(6),pmean,pstdd) ;

%(6) theta - price stickiness
pmean=0.75; pstdd=0.1;
a = (1-pmean)*pmean^2/pstdd^2 - pmean;
b = a*(1/pmean - 1);
LP = LP + lpdfBeta(theta(7),a,b);

%(7) a - labor share in GDP
pmean=1/3; pstdd=0.1;
a = (1-pmean)*pmean^2/pstdd^2 - pmean;
b = a*(1/pmean - 1);
LP = LP + lpdfBeta(theta(8),a,b);
```

Plot - Raw Data (plotraw.m)

```
function x=plotraw(MCMC,raw)
[n,m] = size(MCMC);
sqrn=n^.5;

```

```
figure('Name','MCMC')
for j=1:n;
%       subplot(ceil(sqrn),ceil(sqrn),j);
subplot(ceil(sqrn),ceil(sqrn),j);
    plot(MCMC(j,:));
     if j==1;
          xlabel({'\sigma '});
     end;
     if j==2;
          xlabel('\beta ');
     end;
     if j==3;
          xlabel('\phi ');
     end;
     if j==4;
          xlabel('\epsilon ');
     end;
     if j==5;
          xlabel('\phi_{\pi} ');
     end;
     if j==6;
          xlabel('\phi_{y} ');
     end;
     if j==7;
          xlabel('\theta ');
     end;
     if j==8;
          xlabel('\alpha ');
     end;
     if j==9;
          xlabel('\rho_{v} ');
     end;
     if j==10;
          xlabel('\rho_{a} ');
     end;
     if j==11;
          xlabel('\rho_{z}');
     end;
     if j==12;
          xlabel('\rho_{u}');
     end;
     if j==13;
          xlabel('\sigma_{v} ');
     end;
     if j==14;
```

```
        xlabel('\sigma_{a} ');
    end;
    if j==15;
        xlabel('\sigma_{z}');
    end;
    if j==16;
        xlabel('\sigma_{u}');
    end;
end
if raw == 1
    print('raw_MCMC','-dpng')
else
    print('burn_MCMC','-dpng')
end
end
```

Plot - Cumulative Mean (convcheck.m)

```
function x=convcheck(MCMC)
[n,m] = size(MCMC);
sqrn=n^.5;

MCMC = cumsum(MCMC,2)./repmat(1:m,n,1);

figure('Name','MCMC')
for j=1:n;
%      subplot(ceil(sqrn),ceil(sqrn),j);
subplot(ceil(sqrn),ceil(sqrn),j);
   plot(MCMC(j,:));
    if j==1;
        xlabel({'\sigma '});
    end;
    if j==2;
        xlabel('\beta ');
    end;
    if j==3;
        xlabel('\phi ');
    end;
    if j==4;
        xlabel('\epsilon ');
    end;
    if j==5;
        xlabel('\phi_{\pi} ');
    end;
    if j==6;
```

```
            xlabel('\phi_{y} ');
        end;
        if j==7;
            xlabel('\theta ');
        end;
        if j==8;
            xlabel('\alpha ');
        end;
        if j==9;
            xlabel('\rho_{v} ');
        end;
        if j==10;
            xlabel('\rho_{a} ');
        end;
        if j==11;
            xlabel('\rho_{z}');
        end;
        if j==12;
            xlabel('\rho_{u}');
        end;
        if j==13;
            xlabel('\sigma_{v} ');
        end;
        if j==14;
            xlabel('\sigma_{a} ');
        end;
        if j==15;
            xlabel('\sigma_{z}');
        end;
        if j==16;
            xlabel('\sigma_{u}');
        end;
end

print('conv_MCMC','-dpng')

end
```

Plot - Distribution (plotpost.m)

```
function plotpost(bb_,addplot)
%%
ysca=max(size(bb_));
Q=50; %HP smoothing parameter
% figure('yscaame','Calvo and Indexing','yscaumberTitle','off');
```

```
figure('Name','MCMC Distribution')
for j=1:length(bb_(:,1));
    hold on;
    [n,xout] = hist(bb_(j,:),100);
    n(1,1)=0;n(end,1)=0;
    xsca=(max(xout)-min(xout))/100;
    nn=hpfilter(n,Q);
    subplot(4,4,j);
    if addplot==1;
        AXX=axis;
        AX=[ min([min(xout),AXX(1,1)]) max([max(xout),AXX(1,2)]) 0 max([(max((nn*1.1))/(ysca*xsca)),AXX(1,4)])];
         plot(xout,nn/(ysca*xsca),'linewidth',2);axis(AX);
    else;
         plot(xout,nn/(ysca*xsca),'linewidth',2);axis([min(xout) max(xout) 0 max((nn*1.1))/(ysca*xsca)]);
    end

    if j==1;
         xlabel({'\sigma '});
    end;
    if j==2;
         xlabel('\beta ');
    end;
    if j==3;
         xlabel('\phi ');
    end;
    if j==4;
         xlabel('\epsilon ');
    end;
    if j==5;
         xlabel('\phi_{\pi} ');
    end;
    if j==6;
         xlabel('\phi_{y} ');
    end;
    if j==7;
         xlabel('\theta ');
    end;
    if j==8;
         xlabel('\alpha ');
    end;
    if j==9;
         xlabel('\rho_{v} ');
    end;
```

```
    if j==10;
        xlabel('\rho_{a} ');
    end;
    if j==11;
        xlabel('\rho_{z}');
    end;
    if j==12;
        xlabel('\rho_{u}');
    end;
    if j==13;
        xlabel('\sigma_{v} ');
    end;
    if j==14;
        xlabel('\sigma_{a} ');
    end;
    if j==15;
        xlabel('\sigma_{z}');
    end;
    if j==16;
        xlabel('\sigma_{u}');
    end;
end
print('dist_MCMC','-dpng')
end
```

Kalman Filter (kfilter.m)

```
function [ X_post, P_post, X_prior, Z_tilde, Omega] = kfilter(Z, A
    , C, D, S_vv, X0, P0 )
%KFILTER Compute the Kalman Filter for a VAR(1) process
%       X[t+1] = AX[t] + Cu[t+1]
%   Z[t+1] = DX[t+1] + v[t+1]
%   By Julien Neves

% Get size of observations
T = length(Z);

% Allocate space for estimates
X_prior = cell(1,T);
X_post = cell(1,T+1);
K = cell(1,T);
Z_tilde = cell(1,T);
Omega = cell(1,T);
P_post = cell(1,T+1);

```

```
% Set starting values
X_post{1} = X0;
P_post{1} = P0;

for t = 1:T
    % Compute X_{t+1|t}
    X_prior{t+1}  = A*X_post{t} ;

    % Compute P_{t+1|t}
    P_prior = A * P_post{t} * A' + C * C';

    % Innovations
    Z_tilde{t} = Z(:,t) - D*X_prior{t+1};
    Omega{t} = D*P_prior'*D'+S_vv ;

    % Compute K_{t+1}
    K{t} = P_prior'*D'/(Omega{t});

    % Compute X_{t+1|t+1}
    X_post{t+1}  = X_prior{t+1} + K{t}  * Z_tilde{t};

    % Compute P_{t+1|t+1}
    P_post{t+1} = P_prior- K{t}*D*P_prior;
end

% Convert post and prior estimates to matrices
X_prior = cell2mat(X_prior);
X_prior(:,1) = [];
X_post = cell2mat(X_post);
X_post(:,1) = [];
end
```