Table of Contents

Dynamic AS-AD model

Author: José M. Gaspar, CEF.UP and School of Economics and Management, University of Porto. Email: jgaspar.phd@fep.up.pt


```
function ASADdynamic

clc; clear; close all;
global c0 i0 G b k d  h tau alpha beta mdot yn a0 a1 a2
disp('-------------------------');
disp('Dynamic AS-AD model'),
disp('-------------------------');

options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';

------------------------
Dynamic AS-AD model
------------------------
```

Defining variables and parameters.

```
prompt = {'Autonomous consumption c_0=','Autonomous investment
 i_0=',...
    'Government spending G=','Propensity to consume b=',...
    'Real money balances demand sens. to real output k=',...
    'Real money balances demand sens. to nominal interest rate d=',...
    'Investment sens. to real interest rate h=','Tax rate \tau='};
name = 'AS-AD model exogenous variables and parameters (1)';
numlines = 1;
defaultanswer = {'10','5','5','0.8','0.05',...
    '0.05','0.1','0.3'};    %Default values
answer1 = inputdlg(prompt,name,numlines,defaultanswer,options);

if isempty(answer1)     %Leave program.
    clc;
```

```
    return
end

c0=str2double(answer1{1});
i0=str2double(answer1{2});
G=str2double(answer1{3});
b=str2double(answer1{4});
k=str2double(answer1{5});
d=str2double(answer1{6});
h=str2double(answer1{7});
tau=str2double(answer1{8});


prompt={'Parameter dy/dt=\alpha(A+a_1*y(t)-hr(t)  \alpha=',...
    'Parameter dr/dt=\beta(-M/P+L+ky(t)-ur(t))  \beta=',...
    'Money supply growth rate dm/dt=',...
    'Natural output (in logs) y_n='};
name='AS-AD model exogenous variables and parameters';
numlines=1;
defaultanswer={'0.1','1','0.01','1'};
answer2=inputdlg(prompt,name,numlines,defaultanswer,options);       %
 ""

if isempty(answer2)     %""
    clc;
    return
end

alpha=str2double(answer2{1});
beta=str2double(answer2{2});
mdot=str2double(answer2{3});
yn=str2double(answer2{4});

%Output must be non-negative
if yn<0
    errordlg('Equilibrium output is negative','Error');
end

%Initial conditions
prompt={'Real output initial value  y(0)=',...
    'Expected inflation initial value  \pi^e(0)=',...
    'initial time  t_0=','final time  t_f='};
name='Initial conditions';
numlines=1;
defaultanswer={'0.05','0.15','0','100'};        %Default values
answer3=inputdlg(prompt,name,numlines,defaultanswer,options);       %
 ""

if isempty(answer3)     % ""
    clc;
    return
end

y0=str2double(answer3{1});
```

```
pie0=str2double(answer3{2});
t0=str2double(answer3{3});
tf=str2double(answer3{4});

a0=(c0+i0+G)./(1-b.*(1-tau)+((h.*k)./d));       %
a1=(h./d)/(1-b.*(1-tau)+((h.*k)./d));              %
a2=h/(1-b.*(1-tau)+((h.*k)./d));
```

Used parameters and values

Display chosen values

```
fprintf(['\nAS-AD dynamic model:\n                    '...
    'dy/dt=a1*dm/dt-alpha(a1-a2*beta)*(y-yn)-a1*pie\n         '...
    'dpie/dt=beta*alpha(y-yn)\n']);
fprintf(['\nAS and AD curves:\n                     '...
    '(AD)y=dm/dt-alpha(a1-a2*beta*pie                  '...
    '(Long run AS) y=yn ']);
fprintf(['\ny: Real output\npie: Expected inflation\na0='...
    '(c0+i0+G)/(1-b*(1-tau)+((h*k)/d) = %g\n'...
    'a1:(h/d)/(1-b*(1-tau)+((h*k)/d)) =  %g\n'...
    'a2: h/(1-b*(1-tau)+((h*k)/d)) = %g\n'],...
    a0,a1,a2);
fprintf(['\nParameter values:\nc0 = %g; i0 =  %g; G = %g;'...
    'b = %g; k = %g; d = %g; h = %g; tau = %g;\n'],...
    c0,i0,G,b,k,d,h,tau);
fprintf('alpha = %g; beta = %g; mdot = %g; yn = %g\n',...
    alpha,beta,mdot,yn);
fprintf(['\nInitial conditions:\ny0 = %g; pie0 =  %g\n----------'...
    '-------------------------------------------------\n'],...
    y0,pie0);
```

```
AS-AD dynamic model:
                    dy/dt=a1*dm/dt-alpha(a1-a2*beta)*(y-yn)-a1*pie
         dpie/dt=beta*alpha(y-yn)

AS and AD curves:
                    (AD)y=dm/dt-alpha(a1-a2*beta*pie
 (Long run AS) y=yn
y: Real output
pie: Expected inflation
a0=(c0+i0+G)/(1-b*(1-tau)+((h*k)/d) = 37.037
a1:(h/d)/(1-b*(1-tau)+((h*k)/d)) =  3.7037
a2: h/(1-b*(1-tau)+((h*k)/d)) = 0.185185

Parameter values:
c0 = 10; i0 =  5; G = 5;b = 0.8; k = 0.05; d = 0.05; h = 0.1; tau =
 0.3;
alpha = 0.1; beta = 1; mdot = 0.01; yn = 1

Initial conditions:
y0 = 0.05; pie0 =  0.15
------------------------------------------------------------
```

IVP: Initial value problem

solving the differential equations numerically

```
[t,ydot]=ode45(@dynamicasad,[t0 tf],[y0 pie0]);

if all(ydot(1,:) > 0) %If values for real output are nonnegative,
 continue.
else
    ButtonName = questdlg(['Chosen parameter values imply negative'...
        'real output. Continue?'], ...
                          'Caution!','Yes', 'No','No');
                      fprintf(['\n!!!Real output cannot'...
                          'be negative at any t:\nDisplaying'...
                          'simulation as illustration. \n\n'])
                      if strcmp(ButtonName,'Não')==1 % Cancels program
                          fprintf(['\nSimulation cancelled: Real'...
                              'output cannot be negative at any t\n
\n']);
                              return
                  end
end
```

Steady-state equilibrium.

```
options=optimset('Display','off');
% solving for the steady state of y and pie
[pe]=fsolve(@(y) dynamicasad(t,y),[y0;pie0],options);
pieq=mdot;
disp(' ');
disp('Steady-State:');
fprintf(['\nSteady-state real ouput  '...
    'Steady state expected inflation\n'...
    '%f                 %f\n'],pe(1,1),pe(2,1));
disp('Steady state actual inflation');
disp(pieq);
```

```
Steady-State:

Steady-state real ouput  Steady state expected inflation
1.000000                 0.010000
Steady state actual inflation
    0.0100
```

Stability of equilibria

Jacobian matrix of @dynamicasad

```
j = zeros(2,2);
j(1,1) = -alpha.*(a1-a2.*beta);
j(1,2) = -a1;
```

```
j(2,1) = alpha*beta;
j(2,2) = 0;
disp('Eigenvalues');
disp(eig(j));

if 1/d>beta % this is equivalent to real(eig(J))<0
    if isreal(eig(j))==0
    disp('Stable spiral.');
    else
        disp('Stable node.')
    end
elseif 1/d<beta % this is equivalent to real(eig(J))>0
    if isreal(eig(j))==0
    disp('Unstable spiral.')
    elseif isreal(eig(j))==1
    disp('Unstable node.')
    end
else    % this is equivalent to real(eig(J))==0
    disp('Stable center with periodic solutions');
                % No saddle points because det(j)>0!
end
```

```
Eigenvalues
  -0.1759 + 0.5826i
  -0.1759 - 0.5826i

Stable spiral.
```

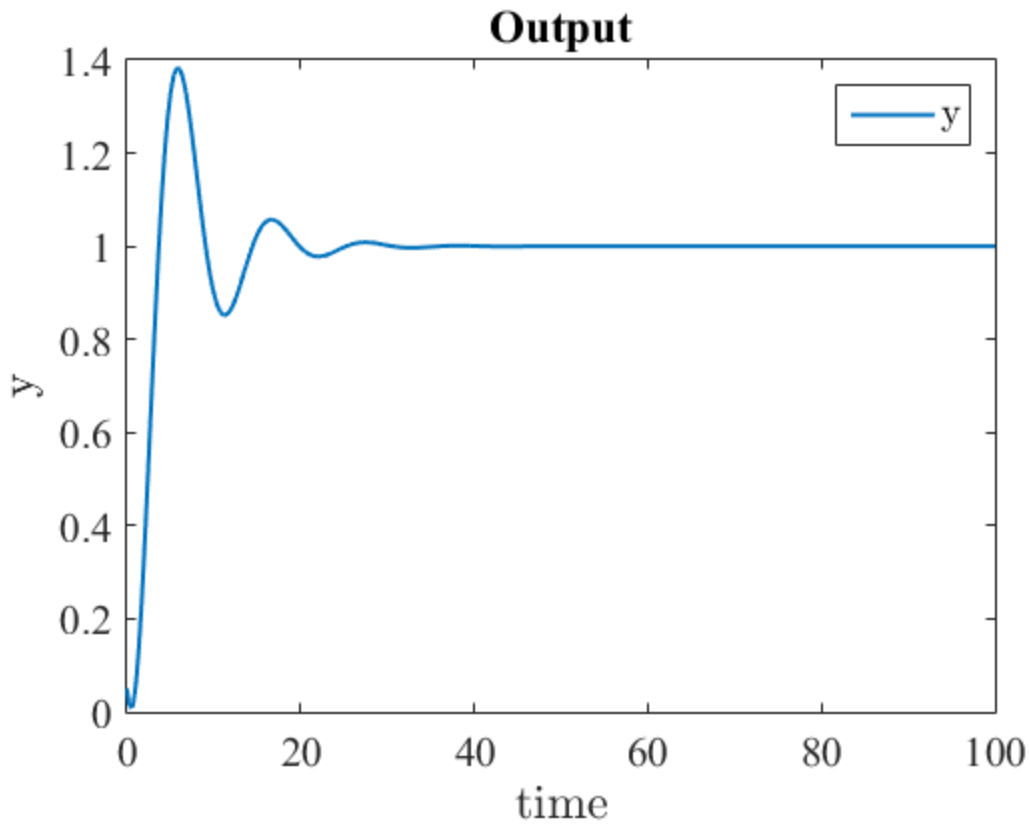
Numerical solution to the IVP

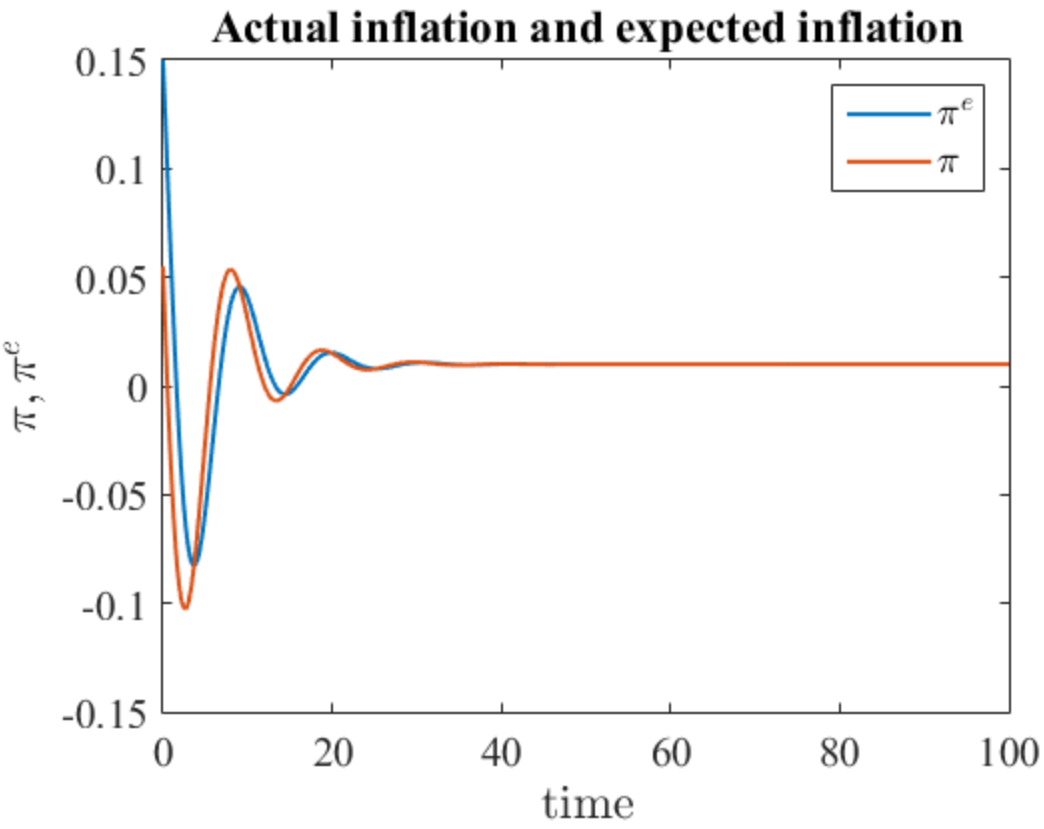
```
% solving the differential equations numerically
[t,ydot]=ode45(@dynamicasad,[t0 tf],[y0 pie0]);

% Plot solution for output
figure(1);
plot(t,ydot(:,1),'LineWidth',1.5);
set(gca,'FontName','Times','FontSize',16);
title('Output');
xl=xlabel('time','Interpreter','latex');
yl=ylabel('y','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('y');
set(leg,'Interpreter','latex');

% Plot solutions for actual and expected inflation
pi=alpha.*((ydot(:,1)-yn)./yn)+ydot(:,2); % Actual inflation
figure(2);
plot(t,[ydot(:,2) pi],'LineWidth',1.5); % actual and expected
 inflation
set(gca,'FontName','Times','FontSize',16);
title('Actual inflation and expected inflation');
xl=xlabel('time','Interpreter','latex');
```

```
yl=ylabel('$\pi, \pi^e$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('$\pi^e$',' $\pi$');
set(leg,'Interpreter','latex');
```





Phase diagram with vector field

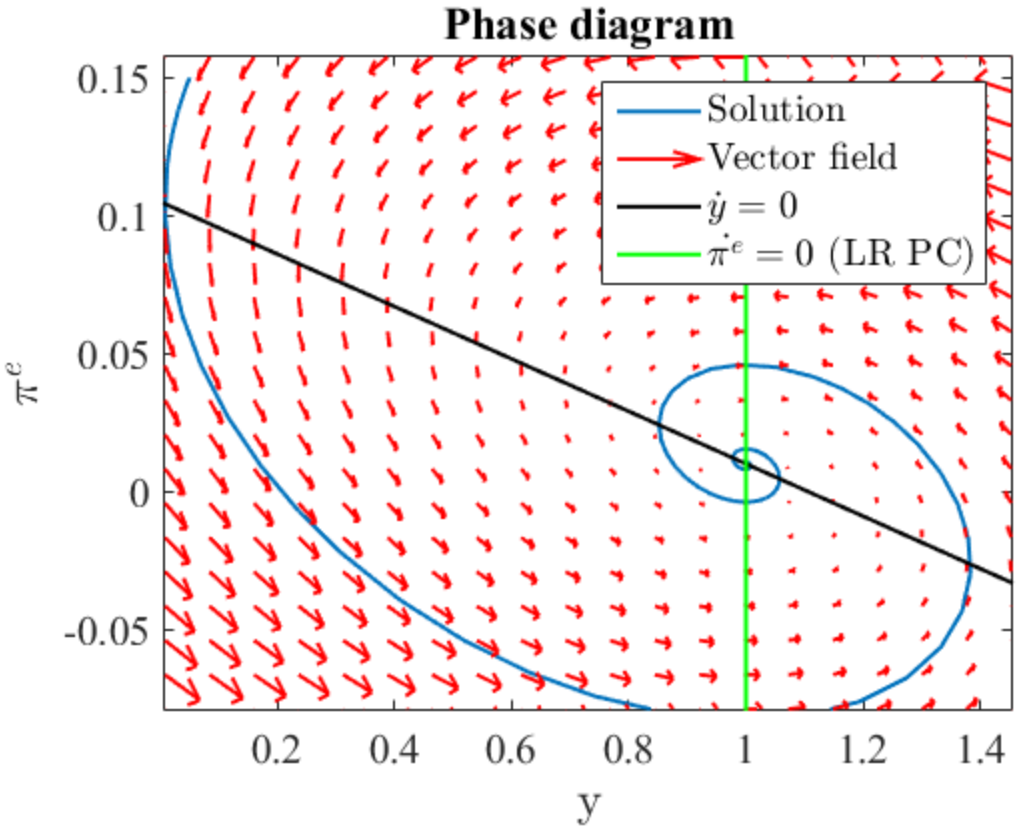
```
figure(3)
plot(ydot(:,1),ydot(:,2),'LineWidth',1.5);   % plot solutions
hold on;
% the following commands are for purposes of graphical representation
if min(ydot(:,1))<0
    yval=linspace(min(ydot(:,1))/1.05,max(ydot(:,1))*1.05,20);
    % using any normalization may be better than 1.05 though...
else
    yval=linspace(min(ydot(:,1))/1.05,max(ydot(:,1))*1.05,20);
end

if min(ydot(:,2))<0     % "  "
    pieval=linspace(min(ydot(:,2))/1.05,max(ydot(:,2))*1.05,20);
else
    pieval=linspace(min(ydot(:,2))/1.05,max(ydot(:,2))*1.05,20);
end
if y0 ~= yn || pie0 ~= mdot     % no phase diagram if starting at ss.
campovectorial(@dynamicasad,yval,pieval);    % Vector fields: See end.
else
end

% Demand pressure curve: dy/dt=0 locus
y=a1.*(mdot-pieval)./(alpha.*(a1-a2*beta))+yn;
```

```
plot(y, pieval,'k','LineWidth',1.5);

% Long-run Phillips curve: dpie/dt=0 locus
line([yn yn],[min(pieval) max(pieval)], [1
 1],'Color','g','LineWidth',1.5);
set(gca,'FontName','Times','FontSize',16);
title('Phase diagram');
xl = xlabel('y','Interpreter','latex');
yl = ylabel('$\pi^e$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('Solution','Vector field','$\dot{y}=0$',...
    '$\dot{\pi^e}=0$ (LR PC)');
set(leg,'Interpreter','latex');
axis([min(yval) max(yval) min(pieval) max(pieval)])
hold off;
```



Add variables to workspace

```
assignin('base','tbeforeshock',t);
assignin('base','Ybeforeshock',ydot(:,1));
assignin('base','Piebeforeshock',ydot(:,2));
```

Modeling shocks

Apply shocks to the model?

```
ButtonName3 = questdlg('Apply shocks to the AS-AD model?', ...
                       'Shocks to the AS-AD
 model','Yes', 'No','No');
if strcmp(ButtonName3,'No')==1     % End program
   return
end
y0=ydot(:,1);   % New initial condition for y (1)
pie0=ydot(:,2); % New initial condition for pie (1)
pieinit=pie0(length(pie0)); % New initial condition for pie (1)
yn0=yn; % Define old Long-run Phillips curve: dpie/dt=0 locus

% Monetary policy shocks through money supply growth
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt = {'Define new money supply growth rate, dm/dt'};
name = 'Monetary policy shocks (default value means no shock)';
numlines = 1;
defaultanswer = {num2str(mdot)};     % Default values
answer4 = inputdlg(prompt,name,numlines,defaultanswer,options);
if isempty(answer4)
    mdot=str2double(answer2{3});
else
    mdot=str2double(answer4{1});
end

% Fiscal policy shocks through taxes and public spending
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt = {'Define new public spending (permanent change),  G=',...
    'Define new tax rate (permanent change) , \tau='};
name = 'Fiscal policy shocks (default values means no shocks)';
numlines = 1;
defaultanswer = {num2str(G),num2str(tau)};     %Default values
answer5 = inputdlg(prompt,name,numlines,defaultanswer,options);
mdotp=(y0-a0)./a1-(a2/a1).*pie0;    %This comes from the static AD
 curve;
% mdotp=m-p is the residual from the AD static curve that will be used
% to compute the new initial value of y after a fiscal policy shock.
if isempty(answer5)
    G=str2double(answer1{3});
    tau=str2double(answer1{8});
    yinit=y0(length(y0));   % New initial conditions for y (2)
else
    % redefine variables
   G=str2double(answer5{1});
   tau=str2double(answer5{2});
   a0=(c0+i0+G)./(1-b.*(1-tau)+((h.*k)./d));       %
```

```
    a1=(h./d)/(1-b.*(1-tau)+((h.*k)./d));              %
    a2=h/(1-b.*(1-tau)+((h.*k)./d));
    % Set new initial condition for y after fiscal policy shock
    yinit=a0+a1.*mdotp(length(mdotp))+...
        a2.*pieinit;  % New initial conditions for y (2)

end

% Technology shocks through natural output
options.Resize='on';
options.WindowStyle='normal';
options.Interpreter='tex';
prompt = {'Define new natural output level, y_n='};
name = 'Technology shocks (default value means no shock)';
numlines = 1;
defaultanswer = {num2str(yn)};     % Default values
answer6 = inputdlg(prompt,name,numlines,defaultanswer,options);
if isempty(answer6)
    yn=str2double(answer2{4});
else
    yn=str2double(answer6{1});
end
if isempty(answer4) && isempty(answer5) && isempty(answer6)
    return %no shocks -> end program
end

% Solving the differential equations numerically
[t,ydot2]=ode45(@dynamicasad,[t0 tf],[yinit pieinit]);

% Output
figure(4);
plot(t,ydot2(:,1),'LineWidth',1.5); % Output
set(gca,'FontName','Times','FontSize',16);
title('Output');
xl = xlabel('time','Interpreter','latex');
yl = ylabel('y','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('y');
set(leg,'Interpreter','latex');
if yinit <= yn+0.0001 && yinit >= yn-0.00001 && pieinit <= mdot
+0.0001...
        && pieinit >= mdot-0.0001   % initial value depends on
 numerical
    %solution of ode45
    axis([min(t) max(t) 0 max(ydot(:,1))+1]);
else
end

% Actual and expected inflation
pi2=alpha.*((ydot2(:,1)-yn)./yn)+ydot2(:,2); % Actual inflation
figure(5);
plot(t,[ydot2(:,2) pi2],'LineWidth',1.5);   %actual and expected
 inflation
```

```
set(gca,'FontName','Times','FontSize',16);
title('Actual inflation and expected inflation');
xl = xlabel('time','Interpreter','latex');
yl = ylabel('$\pi, \pi^e$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('$\pi^e$',' $\pi$');
set(leg,'Interpreter','latex');
if yinit <= yn+0.0001 && yinit >= yn-0.0001 && pieinit <= mdot
+0.0001...
        && pieinit >= mdot-0.0001   % initial value depends on
 numerical
    %solution of ode45
    axis([min(t) max(t) min(ydot(:,2)) max(ydot(:,2))+0.1]);
else
end

% Phase diagram
figure(6)

% the following commands are for purposes of graphical representation
if min(ydot2(:,1))<0
    yval2=linspace(min(ydot2(:,1))*1.05,max(ydot2(:,1))*1.05,20);
else
    yval2=linspace(min(ydot2(:,1))/1.05,max(ydot2(:,1))*1.05,20);
end

if min(ydot2(:,2))<0     % "  "
    pieval2=linspace(min(ydot2(:,2))*1.05,max(ydot2(:,2))*1.05,20);
else
    pieval2=linspace(min(ydot2(:,2))/1.05,max(ydot2(:,2))*1.05,20);
end
if yinit >= yn+0.0001 || yinit <= yn-0.0001 || pieinit >= mdot
+0.001...
        || pieinit <= mdot-0.0001   % initial value depends on
 numerical
    %solution of ode45
plot(ydot2(:,1),ydot2(:,2),'LineWidth',1.5);
hold on;
campovectorial(@dynamicasad,yval2,pieval2);    % Vector fields - See
 end.
else
end

% Demand pressure curve: dy/dt=0 locus
y1=a1.*(mdot-pieval2)./(alpha.*(a1-a2*beta))+yn; %dy/dt=0 locus after
 shock
plot(y, pieval,'k','LineWidth',1.5);    % old dy/dt locus
plot(y1, pieval2,'m','LineWidth',1.5);  % new dy/dt


% Long-run Phillips curve: dpie/dt=0 locus
line([yn yn],[min(pieval2) max(pieval2)], [1 1],'Color','g',...
    'LineWidth',1.5);   %new L-R PC
```

```
line([yn0 yn0],[min(pieval2) max(pieval2)], [1 1],'Color','c',...
    'LineWidth',1.5);   %old L-R PC
set(gca,'FontName','Times','FontSize',16);
title('Phase diagram');
xl =xlabel('y','Interpreter','latex');
yl = ylabel('$\pi^e$','Interpreter','latex');
set(xl,'FontSize',18);
set(yl,'FontSize',18);
leg=legend('Solution','Vector field','$\dot{y}=0$ (1)',...
    '$\dot{y}=0$ (2)','$\dot{\pi^e}=0$ (LR PC) (1)',...
    '$\dot{\pi^e}=0$ (LR PC) (2)');
set(leg,'Interpreter','latex');
axis([min(yval2) max(yval2) min(pieval2) max(pieval2)])
hold off;

% Display parameter values after shocks
disp('----------------------------');
disp('Policy and Technology shocks'),
disp('----------------------------');
fprintf('\na0 = %g\na1 =  %g\a2 = %g\n',...
    a0,a1,a2);
fprintf(['\nNew parameter values:\nc0 = %g; i_0 =  %g;'...
    'G = %g; b = %g; k = %g;'...
    'd = %g; h = %g; tau = %g;\n'],...
    c0,i0,G,b,k,d,h,tau);
fprintf('alpha = %g; beta = %g; mdot = %g; yn = %g',...
    alpha,beta,mdot,yn);
fprintf(['\nNew initial conditions:\ny0 = %g; pie0 =  %g
\n----------'...
    '-------------------------------------------------\n'],...
    yinit,pieinit);

%New steady-state values
options=optimset('Display','off');
[pe1]=fsolve(@(y) dynamicasad(t,y),[y0(length(y0));...
    pie0(length(pie0))],options);    %solve for the ss values of y and
 pie
pieq=mdot;
disp(' ');
disp('New Steady-State:');
fprintf(['\nSteady-state real ouput    Steady state expected'...
    'inflation\n      %f                 %f\n'],pe1(1,1),pe1(2,1));
disp('Steady state actual inflation');
disp(pieq);

%%Add variables to workspace
assignin('base','taftershock',t);     %time vector
assignin('base','Yaftershock',ydot2(:,1));  %output vector after shock
assignin('base','Pieaftershock',ydot2(:,2));    % exp. inflation a/s.
```

Functions

```
%Differential equations for the AS-AD model
```

```
function [ ydot ] = dynamicasad( ~,y )
global alpha beta yn mdot  a1 a2

ydot=zeros(2,1);
ydot(1)=a1.*mdot-alpha.*(a1-a2.*beta).*...
    (y(1)-yn)-a1.*y(2);       %Aggregate demand dynamics
ydot(2)=beta.*alpha.*(y(1)-yn);    %Aggregate supply dynamics

%Vector field
function campovectorial(func,y1val,r1val)

n1=length(y1val);
n2=length(r1val);
%yp1 e yp2 are the vectors associated with yval and rval,
 respectivelly.
yp1=zeros(n2,n1);       % y1val e y2val have to be equally spaced
yp2=zeros(n2,n1);
for i=1:n1
  for j=1:n2
    ypv = feval(func,0,[y1val(i);r1val(j)]);  %yp1 e yp2 are the
    %vectors associated with y1val and r1val.
    yp1(j,i) = ypv(1);
    yp2(j,i) = ypv(2);
  end
end
quiver(y1val,r1val,yp1,yp2,'r','LineWidth',1.4);   %Draw arrows on the
%velocity vectors
axis tight;
```

```
----------------------------
Policy and Technology shocks
----------------------------

a0 = 37.037
a1 =  3.7037
a2 = 0.185185

New parameter values:
c0 = 10; i_0 =  5;G = 5; b = 0.8; k = 0.05;d = 0.05; h = 0.1; tau =
 0.3;
alpha = 0.1; beta = 1; mdot = 0.01; yn = 1.1
New initial conditions:
y0 = 0.999986; pie0 =  0.0100036
------------------------------------------------------------

New Steady-State:

Steady-state real ouput    Steady state expectedinflation
      1.100000                0.010000
Steady state actual inflation
    0.0100
```

Published with MATLAB® R2015a