Computational Economics: Problem Set 1

Iulian Gramatki, 5442168
Yusong Huang, 5442197
Kamil Sudiyarov, 5442388
Mátyás Farkas , 4835695

May 4, 2015

Problem 1

1)

The market equilibrium is when the demand equals the supply.

$$\textbf{D:}\ p = a - b * q \tag{1}$$

$$\textbf{S:}\ p = c + d * q \tag{2}$$

In equilibrium there exists one price such that demand and supply are equal:

$$\textbf{D=S:}\ p = a - b * q = c + d * q \tag{3}$$

Rearranging this gives the expression stated in the question:

$$b * q + d * q - (a - c) = 0$$

2)

From the previous expression follows the equilibrium allocation:

$$q^* = \frac{a - c}{b + d}$$

Substituting this back to either the demand equation or the supply one gets:

$$p^* = \frac{b * c + a * d}{b + d}$$

Therefore the equilibrium price and equilibrium allocation in terms of model parameters are:

$$(p^*, q^*) = \left(\frac{b * c + a * d}{b + d}, \frac{a - c}{b + d}\right)$$

3)

First consider the ordering, that the first equation is the demand and the second is the supply. Then the equation system can be rewritten in the format $Ax{=}y$ using the following steps:

$$\left.\begin{array}{l} p = a - b * q \rightarrowtail p + b * q = a \\ p = c + d * q \rightarrowtail p - d * q = c \end{array}\right\} \iff \begin{pmatrix} 1 & b \\ 1 & -d \end{pmatrix} * \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} a \\ c \end{pmatrix}$$

This way we arrive at the desired format:

$$A = \begin{pmatrix} 1 & b \\ 1 & -d \end{pmatrix}, x = \begin{pmatrix} p \\ q \end{pmatrix}, y = \begin{pmatrix} a \\ c \end{pmatrix} \tag{4}$$

Solving the equation system using the LU decomposition means the following steps:

1. Calculate the lower triangular L and the upper triangular U by Gauss elimination from the identity and matrix A, respectively.

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Bigg| \begin{pmatrix} 1 & b \\ 1 & d \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \Bigg| \begin{pmatrix} 1 & b \\ 0 & -d - b \end{pmatrix}$$

2. Solve the equations $Lz = y$ using forward substitution:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} a \\ c \end{pmatrix} \rightarrow z_1 = a \rightarrow z_2 = c - a$$

3. Solve the equation $Ux = z$ using backward substitution:

$$\begin{pmatrix} 1 & b \\ 0 & -d - b \end{pmatrix} * \begin{pmatrix} p \\ q \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} a \\ c - a \end{pmatrix}$$

$$q^* = \frac{c - a}{-d - b} = \frac{a - c}{b + d} \rightarrow p^* = z_1 - b * q = a - b * \frac{a - c}{b + d} = \frac{a * d + b * c}{b + d}$$

4. Comparing the solutions we can verify that we have the correct analytical solution.

$$(p^*, q^*) = \left(\frac{b * c + a * d}{b + d}, \frac{a - c}{b + d} \right)$$

$\mathcal{Q.E.D}$

4)

Using the parameters $a = 3, b = 0.5, c = d = 1$ gives:

$$(p^*, q^*) = \left(\frac{7}{3}, \frac{4}{3} \right)$$

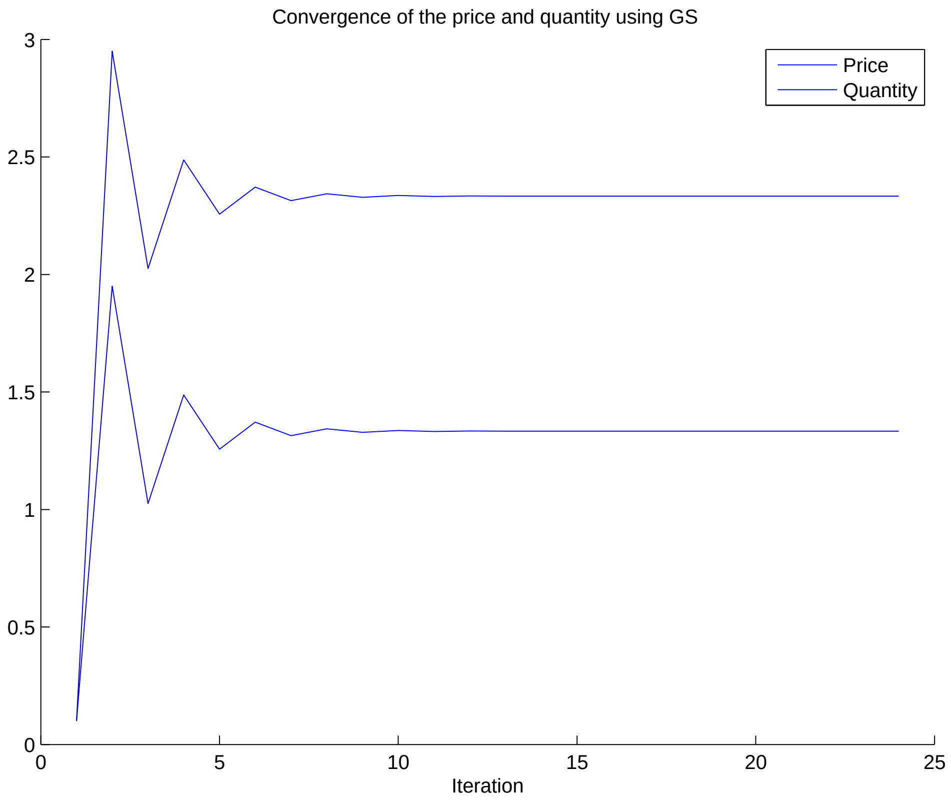


Figure 1: Graphical illustration of convergence for first ordering

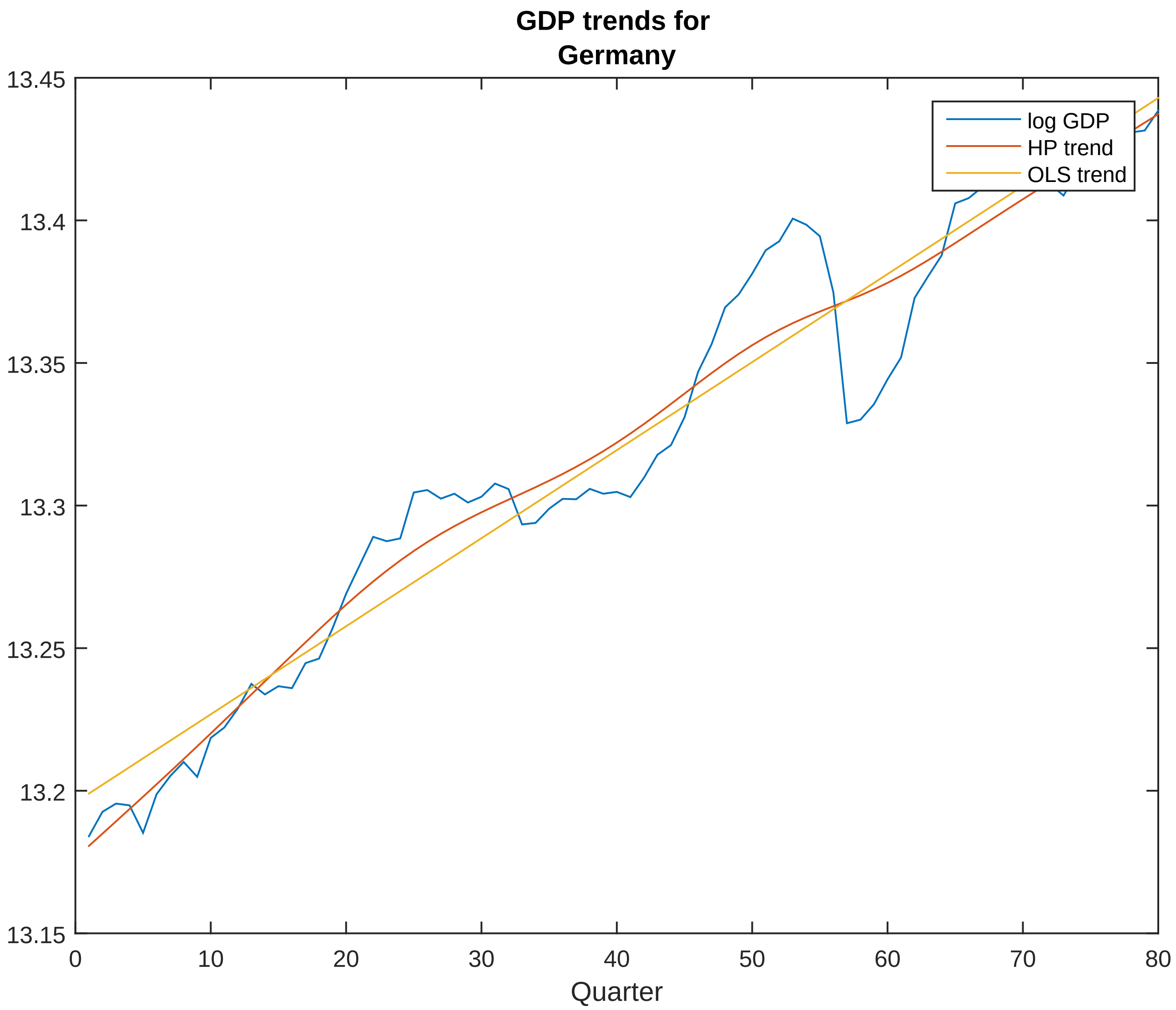
```matlab
%% PROBLEM 1.
clear all
close all
clc
%% 1.5 Gauss Seidel Iteration
%Initialize the problem
disp('Initializing problem... ');
a = 3;
b = 0.5;
c = 1; d = 1;
A = [1 b; 1 -d ];
Z = [a; c];                % Z is the "b" vector

% Starting point for the solution

x_init = [0.1  0.1]';
maxit = 100000;
tol=1e-6;

%Iteration with dx: dx = Q^-1(b-Ax_(i-1))
disp('Gauss Seidel iteration. ');
Q = tril(A);               % create lower triangular matrix of A
Qinv = inv(Q);
X = zeros(2,maxit+1);
X(:, 1) = x_init;
x =  x_init;

for i = 1:maxit
    dx= Qinv * (Z-A*x);
    x = x + dx;
    X(:, i+1) = x;
        if norm(dx) < tol ,  break, end
        if i == maxit, disp('No convergence'), end
end
disp('Plotting..');
figure(1)
hold on
convX=X(:, 1:i+1);
plot(convX(1,:));
plot(convX(2,:))
title(['Convergence of the price and quantity using GS ' ]);
legend('Price','Quantity');
xlabel('Iteration');
disp(['The number of iterations used before convergence: ' , num2str(i)]);
hold off
saveas(1,['PS1P1_convergence.pdf']);
disp('Please press any key to continue.');
pause('on');
pause;
%% No convergence case
disp('No convergence case: Initializing problem...  ');
A = [1 -d ; 1 b  ];
Z = [c; a];

%Starting point for the solution
```

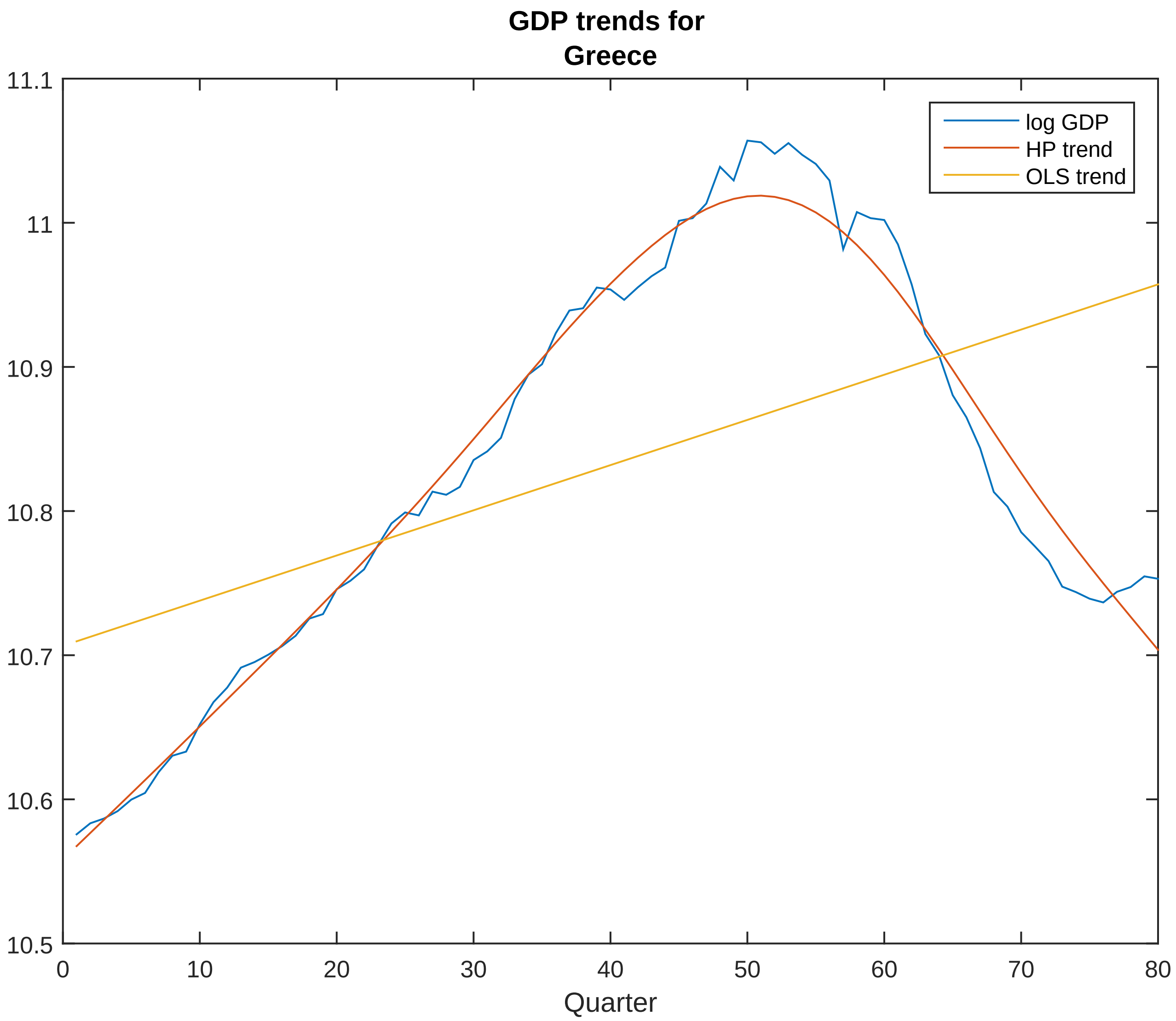
```
x = [0.1  0.1]';

%Iteration with dx: dx = Q^-1(b-Ax_(i-1))

maxit = 100000;
tol=1e-6;
Q = tril(A);
% eig(Q)
Qinv = inv(Q);
X = zeros(2 , maxit+1); %Interation values of x
X(:, 1) = x;
disp('No convergence case: GS iteration...  ');
for i = 1:maxit
    dx= Qinv * (Z-A*x);
    x = x + dx;
    X(:, i+1) = x;
        if norm(dx) < tol ,  break, end
        if i == maxit, disp('No convergence'), end
end

a = input('Would you like to illustrate the no convergence in graph? Please
type y or n and press enter to continue.','s') ;
if strcmpi(a,'y')
% To illustrate that there is no convergence, plot the convergence graph
figure(2)
convX=X(:, 1:i+1);
plot(convX(1,:));
hold on
plot(convX(2,:))
title(['Convergence of the price and quantity using GS ' ]);
legend('Price','Quantity');
xlabel('Iteration');
hold off
end

%% 1.6  Revisiting non convergence using Successive Over-Relaxation
disp('Revisiting non convergence using Successive Over-Relaxation.  ');
maxit = 100000;
tol=1e-6;
Q = tril(A);
Qinv = inv(Q);
X_1 = zeros(2 , maxit+1);
X_1(:, 1) = x;
lam_grid = 0.1:0.1:0.9;

for j = 1: size(lam_grid,2)
    x =  x_init;
    lam = lam_grid(j);
    disp(['Trying lambda: ', num2str(lam)])
              for i = 1:maxit
                dx= Qinv * (Z-A*x);
                x = x + lam*dx;
                X_1(:, i+1) = x;
                    if norm(lam*dx) < tol ,  break, end
                    if i == maxit, disp(['For lamda =', num2str(lam), '
successive over-relaxation does not converge.']), end
```

```
            end
               num_it_for_lam(j) = i;
end

[opt_lam_min_inter ,opt_lam_index] = min(num_it_for_lam);
disp(['The lambda for which the smallest number of iteration to converge was needed:  ' , num2str(lam_grid(opt_lam_index))]);
```

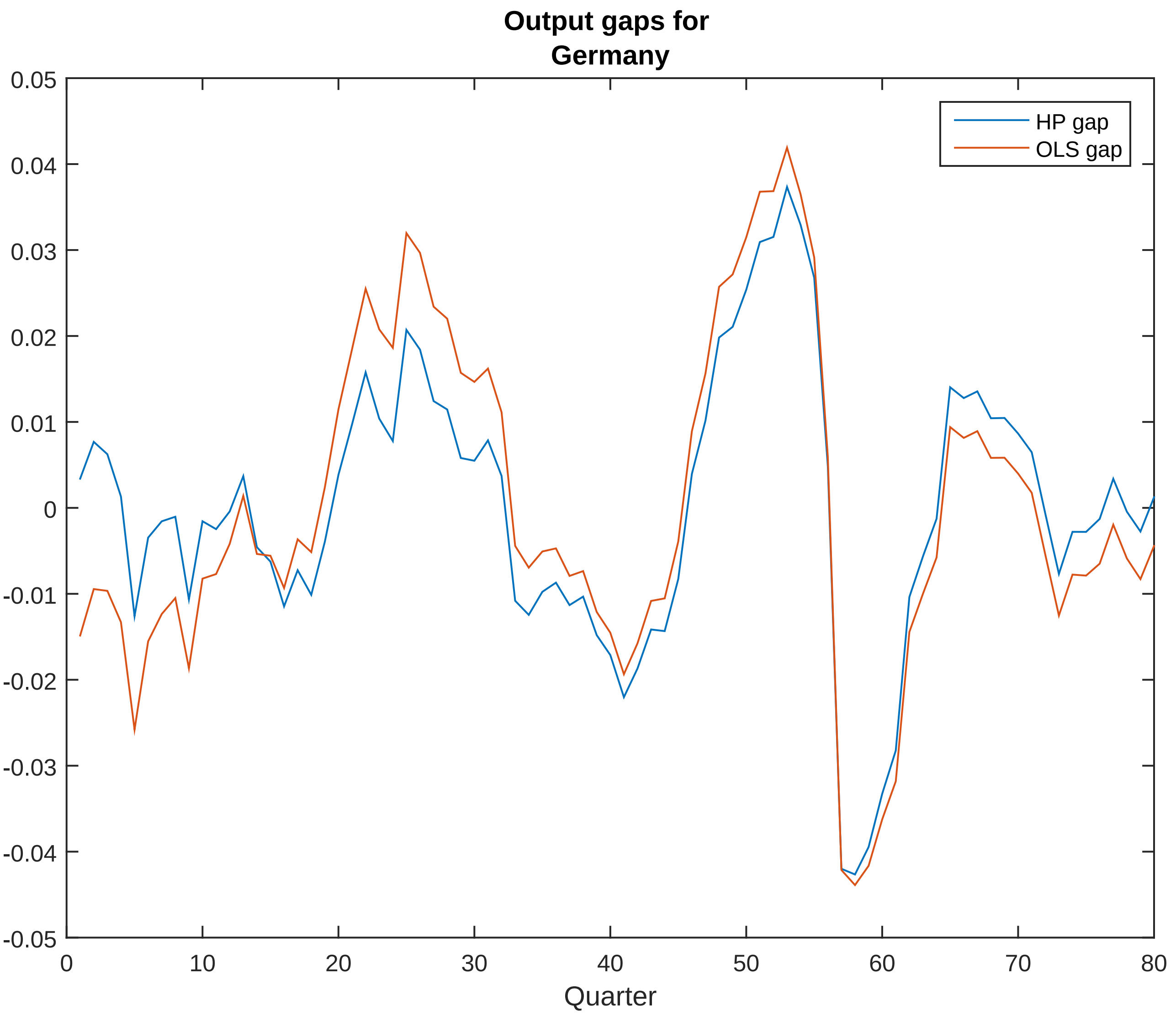
Problem 2.

```
clear all;
clc;
close all;

%% LOAD SERIES

GDP = xlsread('E:\GSEFM\Computational Economics\MatlabCode\data\OECD-
Germany_Greece_GDP.xls', 'E7:CF8');
GDP = GDP';
lGDP = log(GDP);
trendGDP = hpfilter(lGDP,1600);

%% OLS

X = zeros(80,2);
X(:,1) = 1;
X(:,2) = 1:1:80;
OLStrendGDP = zeros(80,2);

for i=1:2
    y = lGDP(:,i);
    b = inv(X'*X)*X'*y;
    OLStrendGDP(:,i) = b(1)+b(2)*X(:,2);
end

exptrendGDP=exp(trendGDP);
expOLStrendGDP=exp(OLStrendGDP);
HPgap = (GDP-exptrendGDP)./exptrendGDP;
OLSgap = (GDP-expOLStrendGDP)./expOLStrendGDP;

V=cell(2);
V{1}='Germany';
V{2}='Greece';

for i=1:2
    figure(i);
    plot(X(:,2), lGDP(:,i), X(:,2), trendGDP(:,i), X(:,2), OLStrendGDP(:,i));
    title(['GDP trends for ' V(i)]);
    legend('log GDP','HP trend','OLS trend');
    xlabel('Quarter');
    saveas(i,['figure ' num2str(i) '.pdf']);
    figure(i+2);
    plot(X(:,2),HPgap(:,i), X(:,2), OLSgap(:,i));
    title(['Output gaps for ' V(i)]);
    legend('HP gap','OLS gap');
    xlabel('Quarter');
    saveas(i+2,['figure ' num2str(i+2) '.pdf']);
end
```

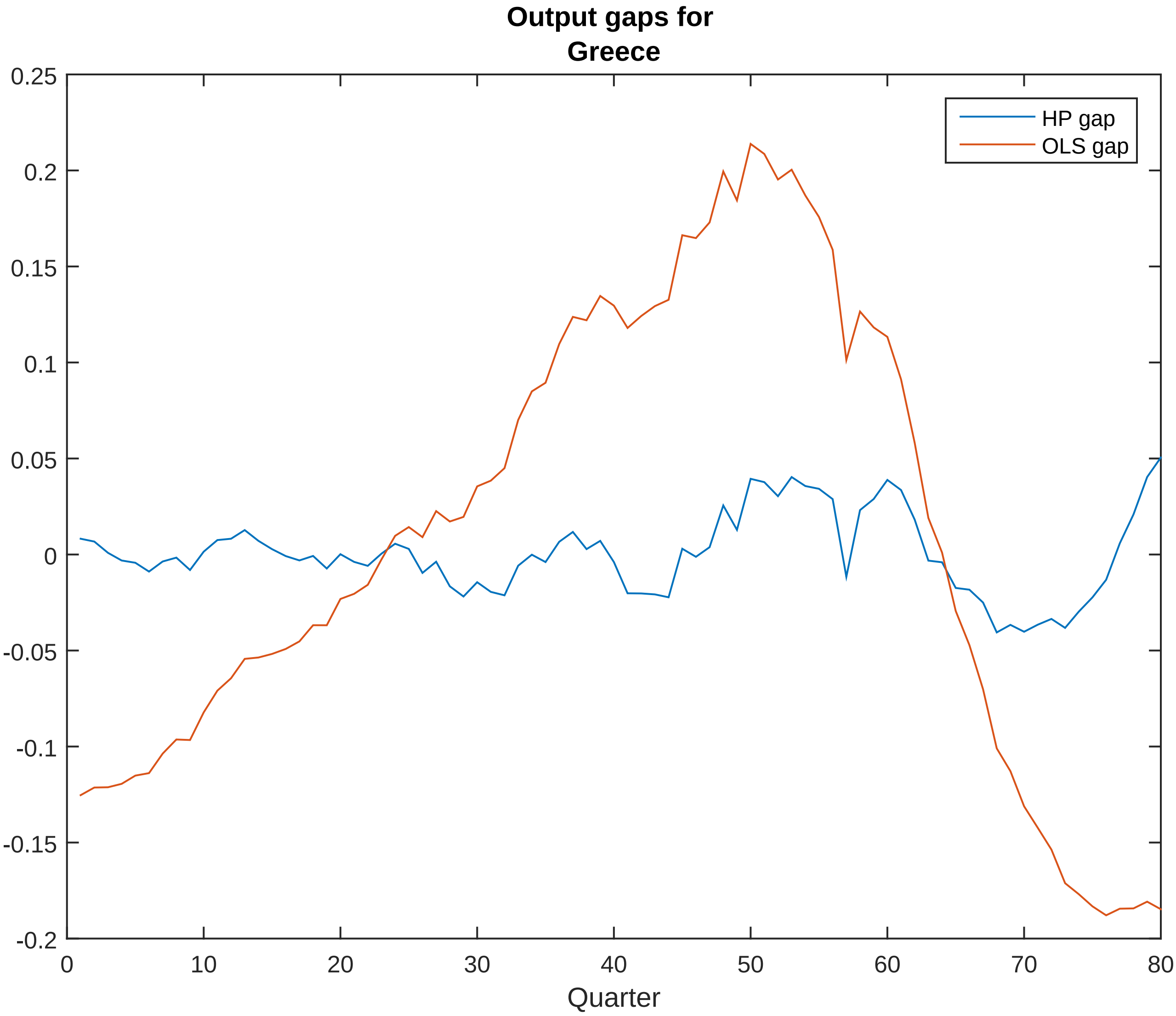

GDP trends for
Germany
13.45
13.4
13.35
13.3
13.25
13.2
13.15
0
10
20
30
40
50
60
70
80
Quarter
log GDP
HP trend
OLS trend



GDP trends for
Greece
11.1
11
10.9
10.8
10.7
10.6
10.5
0
10
20
30
40
50
60
70
80
Quarter
log GDP
HP trend
OLS trend



Output gaps for
Germany
HP gap
OLS gap
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
10
20
30
40
50
60
70
80
Quarter

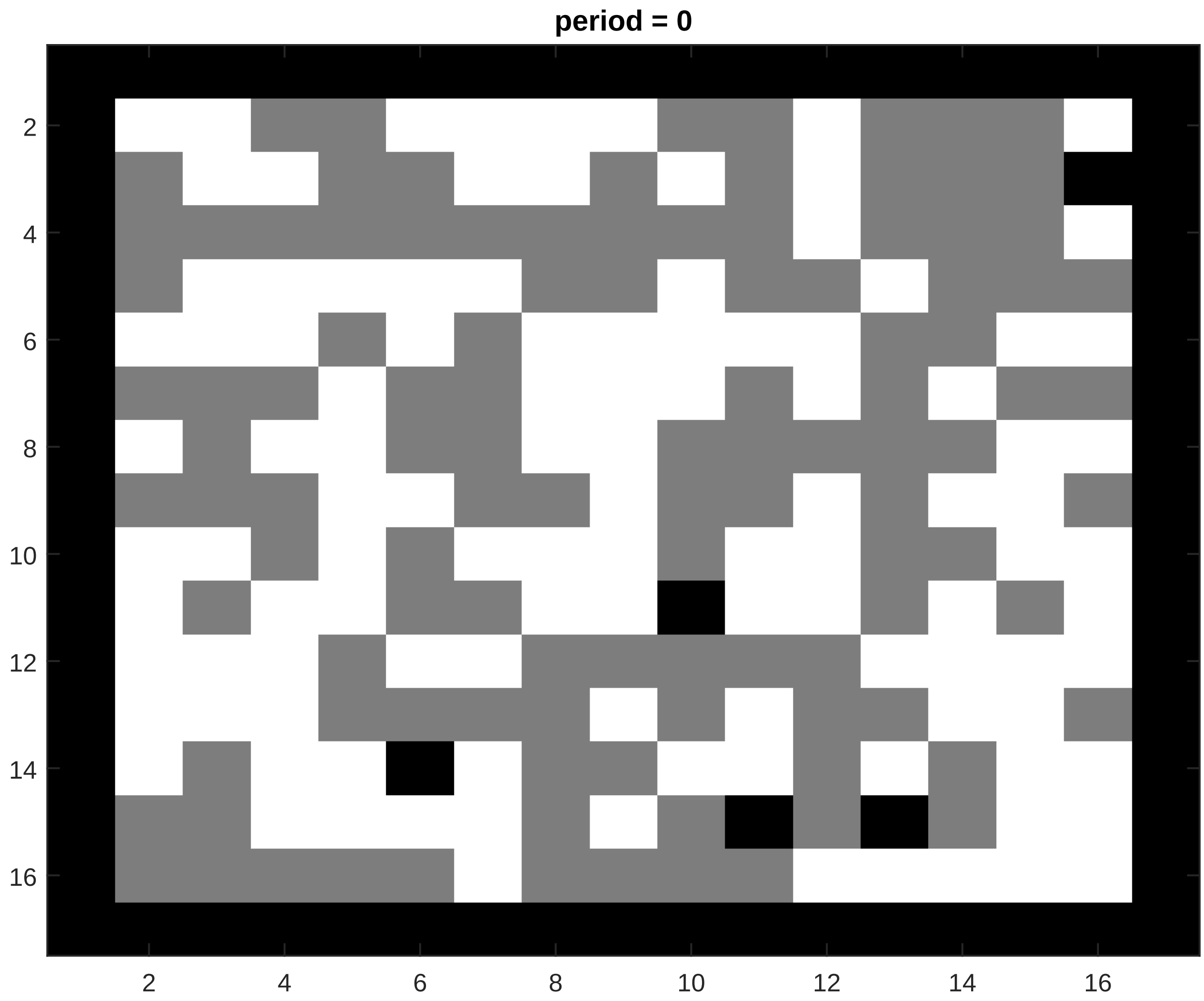


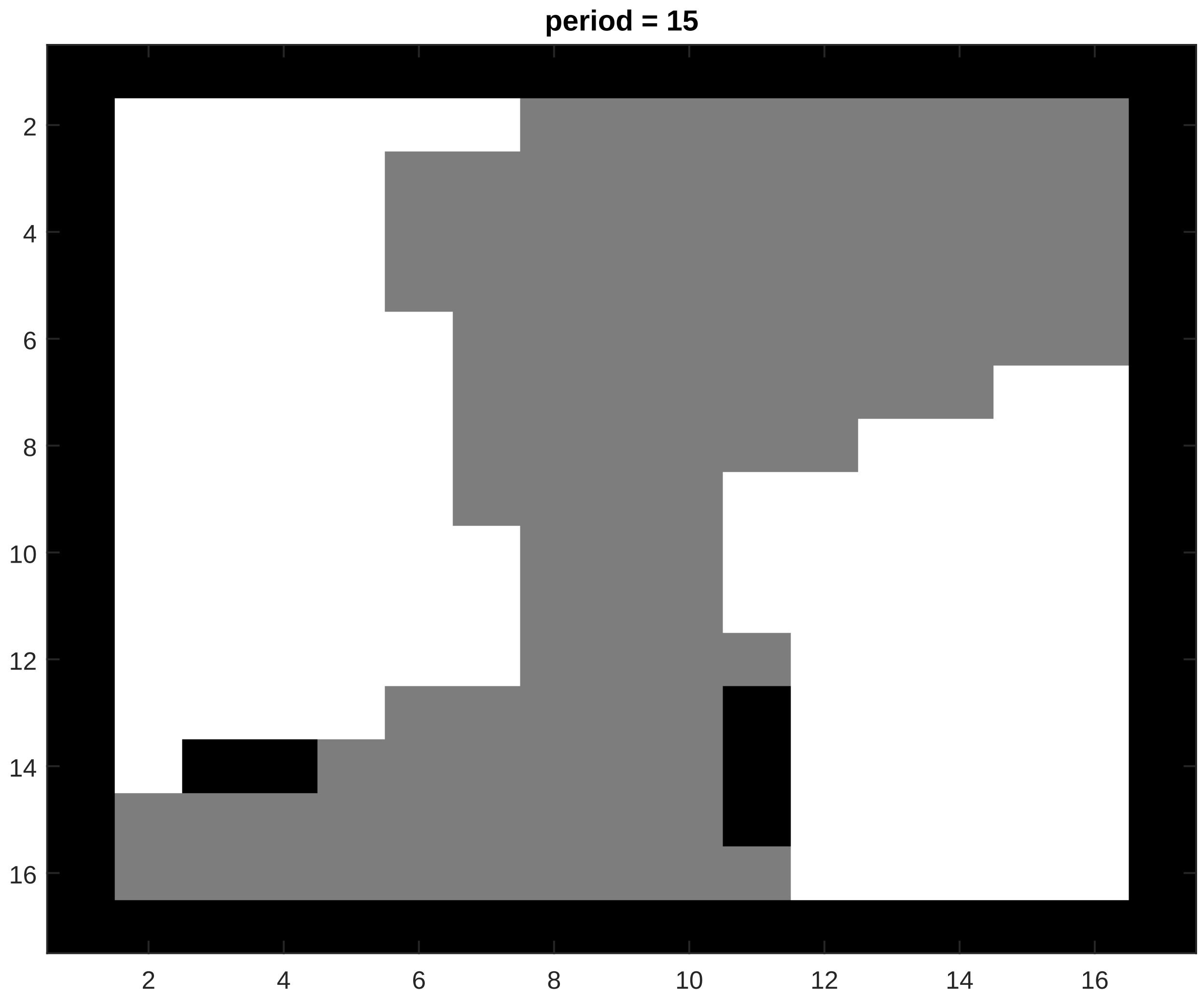
Output gaps for
Greece
HP gap
OLS gap
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
10
20
30
40
50
60
70
80
Quarter

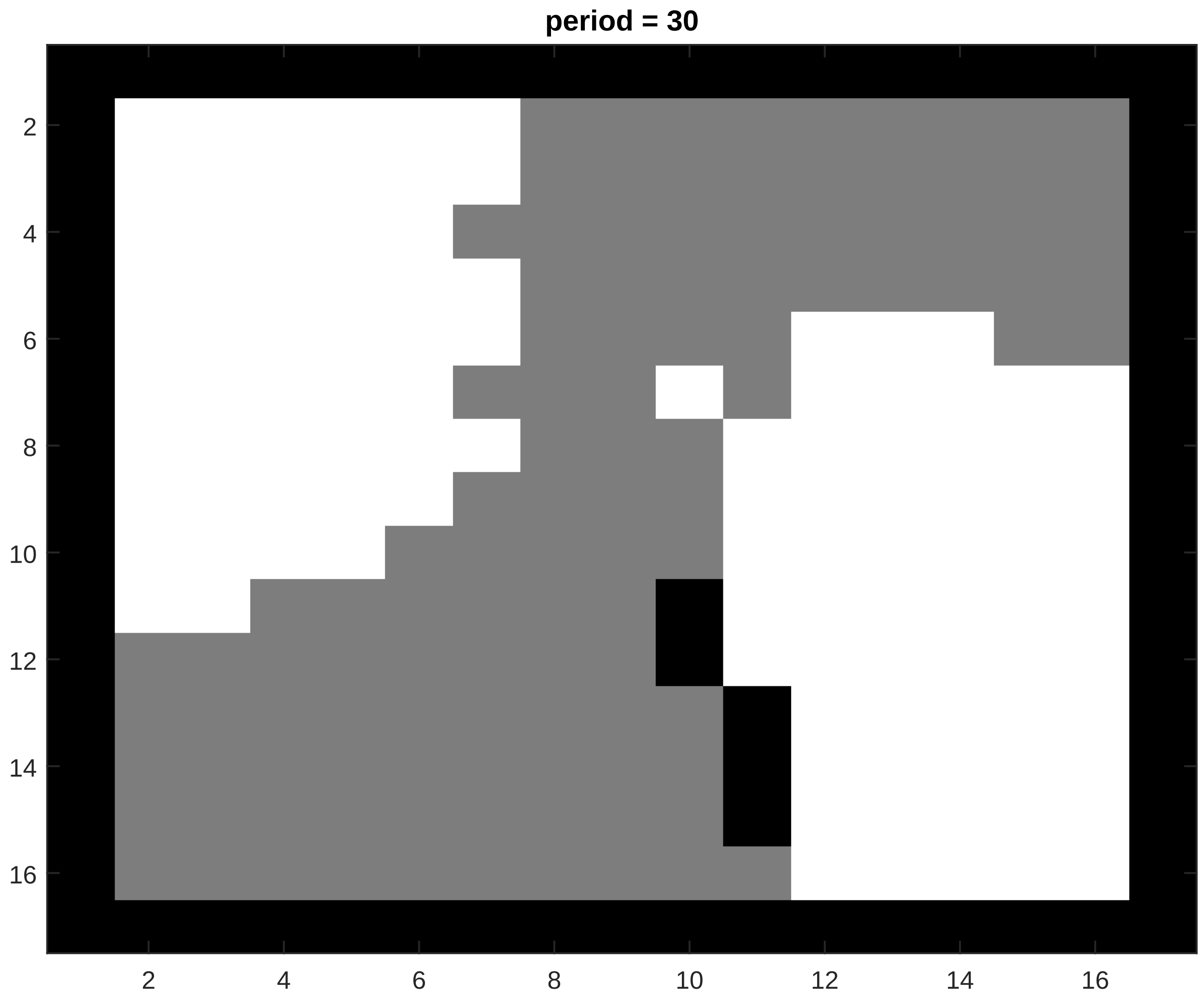
Problem 3.

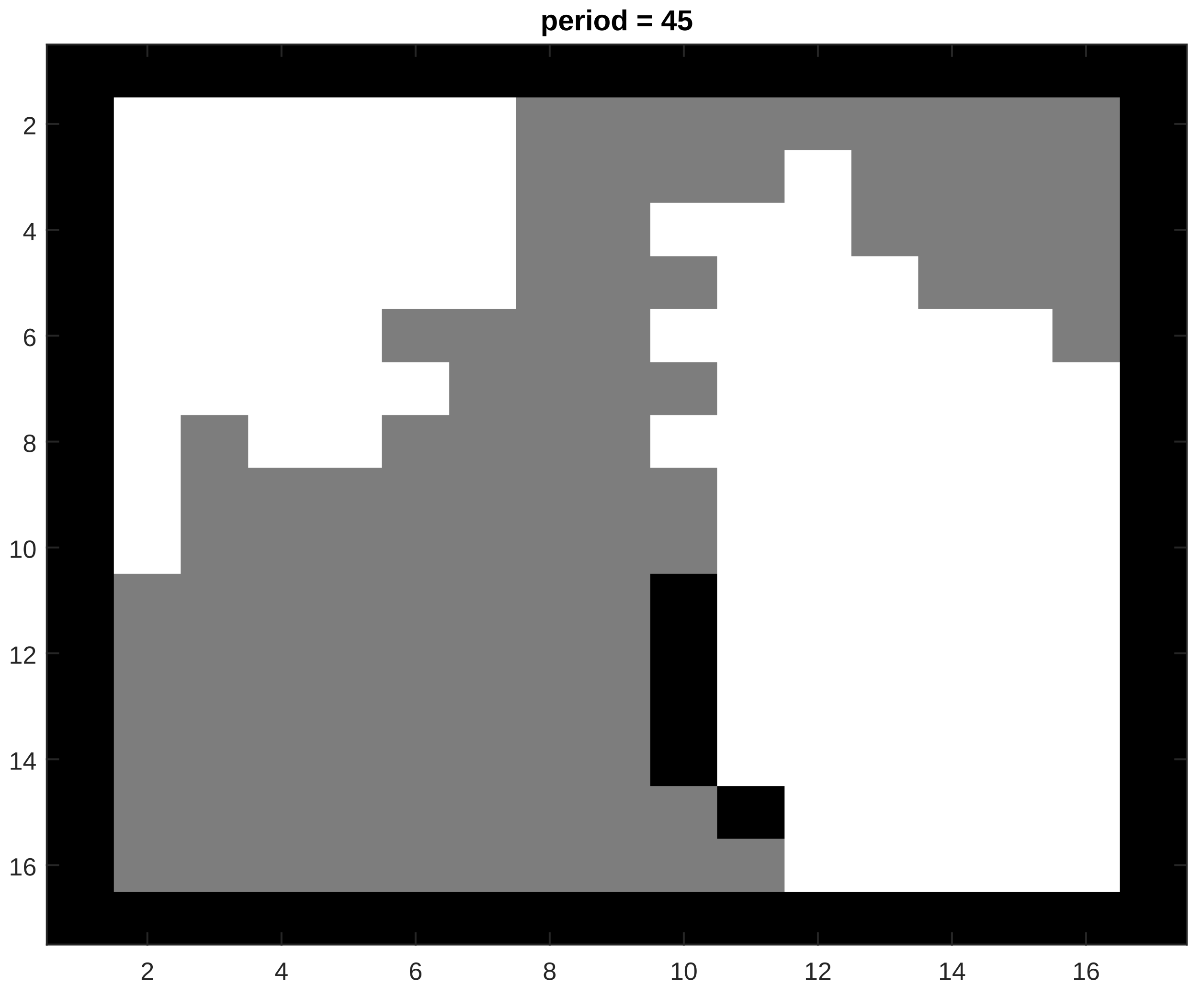
```
clear all;
clc;
close all;
rng('default');

%% initialize the array

a=zeros(17,17);
used=zeros(15,15);  % auxiliary matrix for initial generation, counts which
squared have already been placed;
a(1,:)=2;   % '2' stands for unoccupied houses, '1' for black and '0' for
white
a(17,:)=2;
a(:,1)=2;
a(:,17)=2;
r=randi(225,500);   % generate random staring position of unoccupied and
black houses
count=0;

for i=1:500
    x=floor((r(i)+14)/15);
    y=mod(r(i),15)+1;  %transform random number into 2D coordinates
    if used(x,y)==0     %check is square if unused
        count=count+1;
        used(x,y)=1;
        if count<=5
            a(x+1,y+1)=2;     %make the square unoccupied
            unused(count)=r(i);      %remember unoccupied household for later
use
        else
            a(x+1,y+1)=1;     %make the square black
        end
    end
    if count>=115
        break;
    end
end;

colormap(flipud(gray));
imagesc(a);
title('period = 0');
pause(0.3);
saveas(1,'period0.pdf');

t=45;   %number of periods
for j=1:t       %loop over periods
    for i=1:225     %loop over squares
        x=floor((i+14)/15);
        y=mod(i,15)+1;      %transform into 2D coordinates
        if a(x+1,y+1)<2 % unoccupied squares cannot move
            move=0;     % the decision variable - how many neighbors have a
different color
            if a(x,y)==1-a(x+1,y+1) %check the color of all neighbors; our
square of interest has coordinates (x+1,y+1)
                move=move+1;
            end
```

```
            if a(x+1,y)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x+2,y)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x,y+1)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x+2,y+1)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x,y+2)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x+1,y+2)==1-a(x+1,y+1)
                move=move+1;
            end
            if a(x+2,y+2)==1-a(x+1,y+1)
                move=move+1;
            end
            if move>=3      %start the moving process
                new=unused(1);      %pick an unoccupied house
                newx=floor((new+14)/15);
                newy=mod(new,15)+1;     %transform into 2D coordinates
                a(newx+1,newy+1)=a(x+1,y+1);      %enter new house
                a(x+1,y+1)=2;                      %leave old house
                unused=unused(2:5);   %shift the queue of unoccupied houses
                unused(5)=i;        %the square from which we moved is now
unoccupied; put it last in the queue
                %{
                colormap(flipud(gray));  (display image after every move)
                imagesc(a);
                pause(0.003);
                %}
            end
        end
    end
    colormap(flipud(gray));     %display image after a full period
    imagesc(a);
    title(['period = ' num2str(j)]);
    pause(0.2);
    if mod(j,15)==0
        saveas(1,['period' num2str(j) '.pdf']);
    end
end
```


period = 0
2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16


period = 15
2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16


period = 30
2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16


period = 45
2
4
6
8
10
12
14
16
2
4
6
8
10
12
14
16