Answers to Problem Set 1
Group name: Ferienspass

Sebastian Kühnl: 5642348
Alexander Dück (as: reebyte): 5504077
Patrick Blank (as: paddyblank): 6729110
Christian Wierschem: 6729288

1 Question 1

1.1

A market equilibrium occurs when markets clear. This implies no excess demand (D) or supply (S) of Goods. Thus, $q_D = q_S$. This only occurs when $p_D = p_S$ (the market clearing price prevails).

$$p_D = p_S$$

using

$$p_D = a - b * q_D \text{ and } p_S = c + d * q_S$$

we get

$$a - b * q_D = c + d * q_S$$

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

$$0 = c - a + d * q_S + b * q_D$$

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

Since $q_D = q_S$ holds, this can be simplified even further

$$0 = (b + d) * q - (a - c) \tag{1}$$

■

1.2

Analytical computation of the equilibrium allocation. Alternative approach of previous question used. First, set quantities equal, $q_D = q_S$ and calculate the resulting equilibrium price p^*. By inserting the equilibrium price into both quantity functions, we get the equilibrium quantity and can show that $q_D = q_S$ in fact holds.

$$q_D = q_S$$

$$\frac{a-p}{b} = \frac{c-p}{d}$$

$$d(a-p) = b(p-c)$$

$$da + bc = p(d+b)$$

$$\frac{da+bc}{d+b} = p^*$$

Now, insert into the quantity functions:

$$q_D = \frac{a-p^*}{b} \qquad q_S = \frac{c-p^*}{d}$$

$$q_D = \frac{a-\frac{da+bc}{d+b}}{b} \qquad q_S = \frac{c-\frac{da+bc}{d+b}}{d}$$

$$q_D = \frac{a-c}{d+b} = q \qquad q_S = \frac{a-c}{d+b} = q$$

which can also be computed by rearranging (1):

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

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

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

1.3

The LU decomposition. The application of this procedure can be found in the MATLAB file PS1Q1.m.

1. Rearrange the equations given in the problem set so that, when solving for x, we solve for $x = [p, q]'$.

   $$a = p + bq$$

   $$c = p - dq$$

   Which gives the system

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

2. Decompose the matrix A into the two factors L and U:

   $$A = L * U = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & b \\ 0 & -b-d \end{pmatrix}$$

   Which then gives the following system of equations:

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

3. Solve this system of equations.

   (a) First solve $Ly = b$ by forward induction.

$$\begin{pmatrix}1 & 0\\ 1 & 1\end{pmatrix}\begin{pmatrix}y_1\\ y_2\end{pmatrix} = \begin{pmatrix}a\\ c\end{pmatrix}$$

$$y_1 = a$$

$$y_1 + y_2 = c$$

   which gives

$$y_1 = a$$

$$y_2 = c - a$$

   (b) Then solve $Ux = y$ by backward induction.

$$\begin{pmatrix}1 & b\\ 0 & -(b+d)\end{pmatrix}\begin{pmatrix}p\\ q\end{pmatrix} = \begin{pmatrix}y_1\\ y_2\end{pmatrix} = \begin{pmatrix}a\\ c-a\end{pmatrix}$$

$$-(b+d)q = y_2 = c - a$$

$$p + bq = a$$

   which gives

$$q = \tfrac{a-c}{b+d}$$

$$p = \tfrac{ad+bc}{b+d}$$

1.4

```
%% Subquestion 4 - LU-Decomposition
clear;
clc;
close all;

a=3;
b=0.5;
c=1;
d=c;

A=[1,b;1 -d];

y=[a; c];

[L,U]=lu(A);

t=L\y;
x=U\t;

disp(['LU Result: The market clearing price ', num2str(x(1,1)), ' ...
    clears the market at the quantity ', num2str(x(2,1)), '!']);
```

LU Result: The market clearing price 2.3333 clears the market at the quantity 1.3333!

1.5

```
%% Subquestion 5 - Gauss-Seidel fixed-point iteration

clear;


a=3;
b=0.5;
c=1;
d=c;

%initial guess of quantity
q=0.1;
%Set up difference criterion to a value higher than in the while loop
q_difference=100;

%Set up empty vectors for storage of historical values
q_difference_hist=nan(100,1);
q_Dp_hist=nan(100,1);
q_Dq_hist=nan(100,1);
q_Sq_hist=nan(100,1);
q_Sp_hist=nan(100,1);
q_Time=nan(100,1);

%Iteration index
i=1;
```

```
%Begin iteration
while q_difference>0.01

    q_Dp=a-b*q;        %Demand-price from initial quantity
    q_Dp_hist(i,1)=q_Dp;
    q_Dq_hist(i,1)=q;
    q_Sq=(q_Dp-c)/d;     %Supply-quantity from Demand-price
    q_Sq_hist(i,1)=q_Sq;
    q_Sp=c+d*q_Sq;        %Supply-price for difference
    q_Sp_hist(i,1)=q_Sp;

    if i>1
    q_difference=abs(q_Sp_hist(i,1)-q_Dp_hist(i-1,1));
    q_difference_hist(i,1)=q_difference;
    end
    q=q_Sq;            %Quantity for next guess set
    q_Time(i,1)=i;
    i=i+1;

end
disp(['Gauss-Seidel Iteration Result (using quantity as initial ...
    guess): The market clearing price ', num2str(q_Sp), ' clears ...
    the market at the quantity ', num2str(q_Sq), ' after ...
    ',num2str(i-1),' iterations!']);
```

Gauss-Seidel Iteration Result (using quantity as initial guess): The market clearing price 2.3357 clears the market at the quantity 1.3357 after 9 iterations!

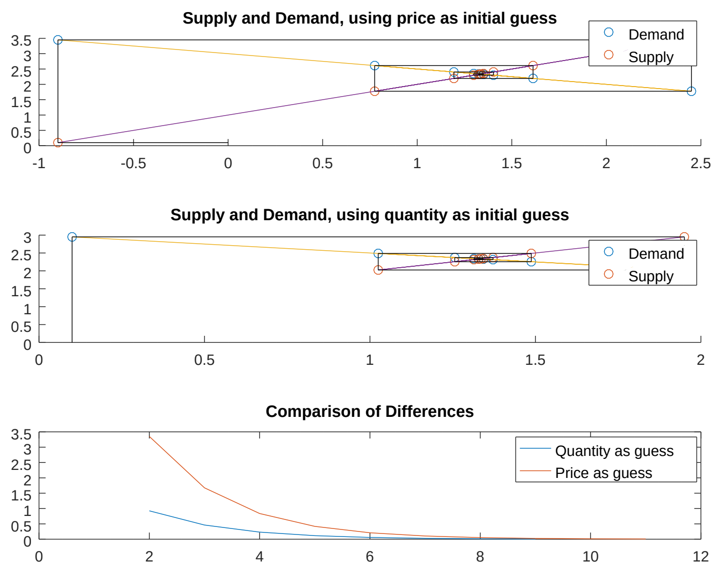
```
%Alternatively: Using the price as a first guess

%Initial Guess
p=0.1;

%Set up difference criterion to a value higher than in the while loop
difference=100;

%Set up empty vectors for storage of historical values
difference_hist=nan(100,1);
Dp_hist=nan(100,1);
Dq_hist=nan(100,1);
Sq_hist=nan(100,1);
Sp_hist=nan(100,1);
Time=nan(100,1);

%Iteration index
i=1;

while difference>0.01

    Sq=(p-c)/d;     %Supply-quantity from price
    Sq_hist(i,1)=Sq;
    Sp=p;       %Supply-price for difference
    Sp_hist(i,1)=Sp;
```

```
    Dp=a-b*Sq;         %Demand-price from initial quantity
    Dp_hist(i,1)=Dp;
    Dq_hist(i,1)=Sq;

    if i>1
    difference=abs(Sq_hist(i-1,1)-Dq_hist(i,1));
    difference_hist(i,1)=difference;
    end
    p=Dp;             %Quantity for next guess set
    Time(i,1)=i;
    i=i+1;

end

disp(['Gauss-Seidel Iteration Result (using price as initial ...
    guess): The market clearing price ', num2str(Sp), ' clears the ...
    market at the quantity ', num2str(Sq), ' after ',num2str(i-1),' ...
    iterations!']);
```

Gauss-Seidel Iteration Result (using price as initial guess): The market clearing price 2.3312 clears the market at the quantity 1.3312 after 11 iterations!

```
figure
subplot(3,1,1);

scatter(Dq_hist,Dp_hist)
hold on
scatter(Sq_hist,Sp_hist)
plot(Dq_hist,Dp_hist,Sq_hist,Sp_hist)
line([Sq_hist(1,1) 0], [Sp_hist(1,1) Sp_hist(1,1)])

%TO SHOW HOW THE ALGORITHM WORKS!
for j=1:(i-1)
line([Dq_hist(j,1) Sq_hist(j+1,1)], [Dp_hist(j,1) Dp_hist(j,1)])
end

for j=1:(i-1)
line([Sq_hist(j,1) Sq_hist(j,1)], [Dp_hist(j,1) Sp_hist(j,1)])
end

title('Supply and Demand, using price as initial guess')
legend('Demand','Supply')
hold off

subplot(3,1,2);
scatter(q_Dq_hist,q_Dp_hist)
hold on
scatter(q_Sq_hist,q_Sp_hist)
plot(q_Dq_hist,q_Dp_hist,q_Sq_hist,q_Sp_hist)
line([q_Dq_hist(1,1) q_Dq_hist(1,1)], [q_Dp_hist(1,1) 0])
%TO SHOW HOW THE ALGORITHM WORKS!
for j=1:(i-1)
line([q_Dq_hist(j,1) q_Sq_hist(j,1)], [q_Dp_hist(j,1) q_Dp_hist(j,1)])
```

```
end

for j=1:(i-1)
line([q_Sq_hist(j,1) q_Sq_hist(j,1)], [q_Dp_hist(j,1) ...
    q_Sp_hist(j+1,1)])
end
title('Supply and Demand, using quantity as initial guess')
legend('Demand','Supply')
hold off

%To show convergence
%figure
%plot(q_Time,q_difference_hist)
%title('Difference, using quantity as initial guess')

subplot(3,1,3);
plot(q_Time,q_difference_hist,Time,difference_hist)
title('Comparison of Differences')
legend('Quantity as guess','Price as guess')




%Non convergent case

a=3;
b=0.5;
c=b;
d=c;

%initial guess of quantity
q=0.1;
%Set up difference criterion to a value higher than in the while loop
nq_difference=100;

%Set up empty vectors for storage of historical values
nq_difference_hist=nan(100,1);
nq_Dp_hist=nan(100,1);
nq_Dq_hist=nan(100,1);
nq_Sq_hist=nan(100,1);
nq_Sp_hist=nan(100,1);
nq_Time=nan(100,1);

%Iteration index


%Begin iteration
for i=1:100
    nq_Dp=a-b*q;       %Demand-price from initial quantity
    nq_Dp_hist(i,1)=nq_Dp;
    nq_Dq_hist(i,1)=q;
    nq_Sq=(nq_Dp-c)/d;    %Supply-quantity from Demand-price
    nq_Sq_hist(i,1)=nq_Sq;
    nq_Sp=c+d*nq_Sq;       %Supply-price for difference
    nq_Sp_hist(i,1)=nq_Sp;
```

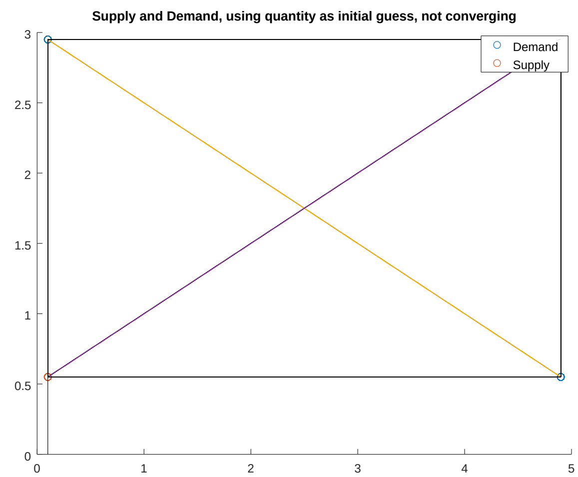



```
    if i>1
    nq_difference=abs(nq_Sp_hist(i,1)-nq_Dp_hist(i-1,1));
    nq_difference_hist(i,1)=nq_difference;
    end
    q=nq_Sq;             %Quantity for next guess set
    nq_Time(i,1)=i;

end
```

```
figure
scatter(nq_Dq_hist,nq_Dp_hist)
hold on
scatter(nq_Sq_hist,nq_Sp_hist)
plot(nq_Dq_hist,nq_Dp_hist,nq_Sq_hist,nq_Sp_hist)
line([nq_Dq_hist(1,1) nq_Dq_hist(1,1)], [nq_Dp_hist(1,1) 0])
%TO SHOW HOW THE ALGORITHM WORKS!
for j=1:(i-1)
line([nq_Dq_hist(j,1) nq_Sq_hist(j,1)], [nq_Dp_hist(j,1) ...
    nq_Dp_hist(j,1)])
end
for j=1:(i-1)
line([nq_Sq_hist(j,1) nq_Sq_hist(j,1)], [nq_Dp_hist(j,1) ...
    nq_Sp_hist(j+1,1)])
end
title('Supply and Demand, using quantity as initial guess, not ...
    converging')
legend('Demand','Supply')
hold off
```

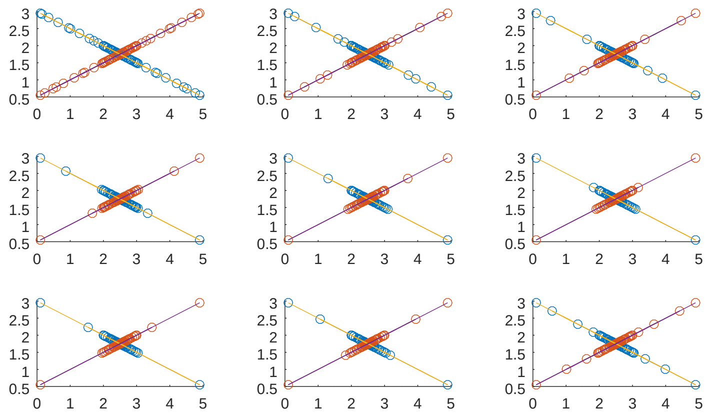
1.6

```
%% Subquestion 6 - Using a dampening factor

lambda=linspace(0.1,0.9,9);
iteration_count=nan(9,1);

a=3;
b=0.5;
c=b;
d=c;




%Set up empty vectors for storage of historical values
d_difference_hist=nan(100,1);
d_Dp_hist=nan(100,1);
d_Dq_hist=nan(100,1);
d_Sq_hist=nan(100,1);
d_Sp_hist=nan(100,1);
d_Time=nan(100,1);


figure
for j=1:9
%initial guess of quantity
q=0.1;
%Set up difference criterion to a value higher than in the while loop
d_difference=100;
i=1;
while d_difference>0.01
```




```
d_Dp=a-b*q;        %Demand-price from initial quantity
d_Dp_hist(i,1)=d_Dp;
d_Dq_hist(i,1)=q;
d_Sq=(d_Dp-c)/d;     %Supply-quantity from Demand-price
d_Sq_hist(i,1)=d_Sq;
d_Sp=c+d*d_Sq;        %Supply-price for difference
d_Sp_hist(i,1)=d_Sp;

if i>1
d_difference=abs(d_Sp_hist(i,1)-d_Dp_hist(i-1,1));
d_difference_hist(i,1)=d_difference;
q=lambda(1,j)*d_Sq_hist(i,1)+(1-lambda(1,j))*d_Sq_hist(i-1,1);
else
q=d_Sq;
end
%Quantity for next guess set
d_Time(i,1)=i;
```

```
    i=i+1;

end
iteration_count(j,1)=i;

subplot(4,3,j)
scatter(d_Dq_hist,d_Dp_hist)
hold on
scatter(d_Sq_hist,d_Sp_hist)
plot(d_Dq_hist,d_Dp_hist,d_Sq_hist,d_Sp_hist)
hold off
end
[M,I] = min(iteration_count);
Size_I=size(I);
c = ...
    categorical({'0.1','0.2','0.3','0.4','0.5','0.6','0.7','0.8','0.9'});

%bar(c,iteration_count)
subplot(4,3,[10 11 12]);
for i=1:Size_I(1,2) %allows for multiple minima
b = bar(c,iteration_count);
title({'Number of iterations needed to find the solution';'The ...
    lowest value is colorized differently'})
set(get(gca,'title'),'Position',[5.5 60 1])
b.FaceColor = 'flat';
b.CData(I(1,i),:) = [.5 0 .5];
end
```



2 Question 2

2.1

```
%% Part 1 - loading and computing the logarithms
clear all;
clc;
```

```
A=xlsread('/Users/sebastiankuhnl/Desktop/GSEFM/Year 1/Semester ...
    1.2/Mathematical ...
    Methods/Problem_Sets/PS1/data/OECD-Germany_Greece_GDP.xls');
Germany=A(1,:);
Greece=A(2,:);

LogGermany=log(Germany);
LogGreece=log(Greece);
```

2.2

```
%% Part 2 - Applying HP filter

%This version requires machine learning and statistics toolbox
smoothing = 1600; %unecessary since it is the default value of the ...
    function but done for sake of completeness
TrendGermany = hpfilter(LogGermany,smoothing);
TrendGreece = hpfilter(LogGreece,smoothing);
```

2.3

```
%% Part 3 - Applying OLS

%Creating time variable
Time = zeros(size(Greece));
Size = size(Greece);
for i=1:Size(1,2)
    Time(1,i)=i;
end

%OLS

Var_GRE=var(LogGreece);
Var_GER=var(LogGermany);
Cov_GRE=cov(Time,LogGreece);
Cov_GER=cov(Time,LogGermany);

b_1_GER=Var_GER/Cov_GER(1,2);
b_1_GRE=Var_GRE/Cov_GRE(1,2);

Mean_Time=mean(Time);
Mean_GER=mean(LogGermany);
Mean_GRE=mean(LogGreece);

b_0_GER=Mean_GER-Mean_Time*b_1_GER;
b_0_GRE=Mean_GRE-Mean_Time*b_1_GRE;

Yhat_GER=b_0_GER+Time*b_1_GER;
Yhat_GRE=b_0_GRE+Time*b_1_GRE;
```

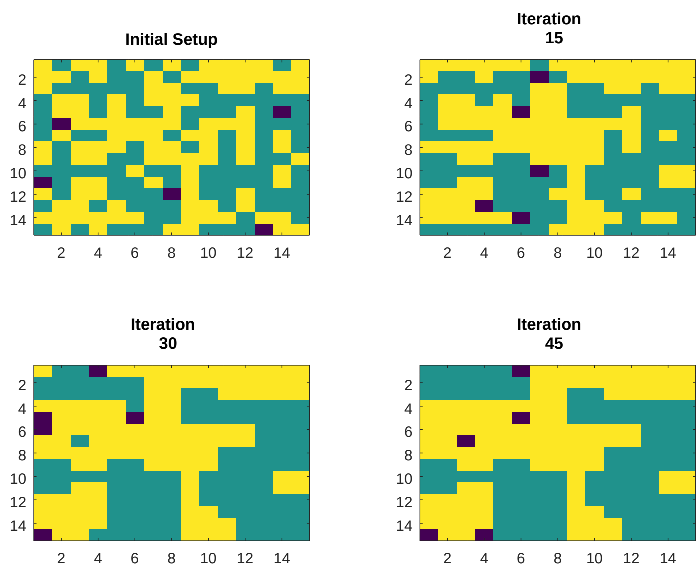
2.4

```
%% Part 4 - Output Gap

Y_GER_trend_HP=exp(TrendGermany);
Y_GER_trend_OLS=exp(Yhat_GER);

Y_GRE_trend_HP=exp(TrendGreece);
Y_GRE_trend_OLS=exp(Yhat_GRE);

Y_Gap_GER_HP=Germany-Y_GER_trend_HP;
Y_Gap_GER_OLS=Germany-Y_GER_trend_OLS;

Y_Gap_GRE_HP=Greece-Y_GRE_trend_HP;
Y_Gap_GRE_OLS=Greece-Y_GRE_trend_OLS;
```

2.5

```
%% Part 5 - Plot

figure
plot(Time,LogGermany,Time,Yhat_GER,Time,TrendGermany);

figure
plot(Time,LogGreece,Time,Yhat_GRE,Time,TrendGreece);

figure
plot(Time,LogGermany,Time,Yhat_GER,Time,TrendGermany,Time,LogGreece,Time,Yhat_GRE,Time,TrendGreece);
```

3 Question 3




```
%% Set up the grid

clear;
clc;
```

```
Board=zeros(15,15);
Iteration_Count=0;
%110 White (1 indicates white)
i=1;
while i<111

Col = randi(size(Board, 2));
Row = randi(size(Board, 1));
if Board(Col,Row)==0
    Board(Col,Row)=1;
    i=i+1;
else
end
end

%110 Black (2 indicates black)

j=1;
while j<111

Col = randi(size(Board, 2));
Row = randi(size(Board, 1));
if Board(Col,Row)==0
    Board(Col,Row)=2;
    j=j+1;
else
end
end

%Plot situation after zero iterations
figure
subplot(2,2,1)
Plot_0=imagesc(Board);
title('Initial Setup')

for ij=1:45
%% Spot unoccupied houses

Non_Occupied=zeros(5,2);
Non_Index=1;

for i=1:15
    for j=1:15
        if Board(i,j)==0

            Non_Occupied(Non_Index,1)=i;
            Non_Occupied(Non_Index,2)=j;
            if Non_Index<5
            Non_Index=Non_Index+1;

            end
        else
        end
    end
end

%% Evaluate position
```

```
Movers_Location=zeros(15,15);
Movers_Count=0;

for i=1:15
    for j=1:15
       Self=Board(i,j);
       if Self ~= 0 %Check if the considered house is actually occupied

       %If possible, get neighbor values, alternatively, set equal ...
           to zero

           if i==1
           Upper_N=0;
           else
           Upper_N=Board(i-1,j);
           end

           if i==15
           Lower_N=0;
           else
           Lower_N=Board(i+1,j);
           end

           if j==1
           Left_N=0;
           else
           Left_N=Board(i,j-1);
           end

           if j==15
           Right_N=0;
           else
           Right_N=Board(i,j+1);
           end

           %Compare to neighbors

           Neigh=zeros(4,1);
           Neigh(1,1)=double((Self==Upper_N));
           Neigh(2,1)=double((Self==Lower_N));
           Neigh(3,1)=double((Self==Left_N));
           Neigh(4,1)=double((Self==Right_N));

           %at least 35% must be equal to not move
           Neigh_Eval=sum(Neigh)/4;

          if Neigh_Eval<0.35
              Movers_Location(i,j)=1; %everyone with a one wants to ...
                  move
              Movers_Count=Movers_Count+1;
          end

       end

    end
end
```

```
%% Only five households are allowed to move each period (since only ...
    five houses are unoccupied)

Allowed_Movers=zeros(5,1);
Allowed_Movers_Pos=zeros(5,2);
Want_to_Move=zeros(Movers_Count,2);
k=1;

for i=1:15
    for j=1:15
       if Movers_Location(i,j)==1
           %Store to movers

           Want_to_Move(k,1)=i;
           Want_to_Move(k,2)=j;

           k=k+1;
       end
    end
end

%Pick five movers

m=1;

if Movers_Count>=5
    Movers_Counter=6;
else
    Movers_Counter=Movers_Count+1;
end
while m<Movers_Counter
    New_Mover=randi(Movers_Count);
    if ismember(New_Mover,Allowed_Movers)==0
        Allowed_Movers(m,1)=New_Mover;
        m=m+1;
    end
end

for i=1:Movers_Counter-1
    Allowed_Movers_Pos(i,1)=Want_to_Move(Allowed_Movers(i,1),1);
    Allowed_Movers_Pos(i,2)=Want_to_Move(Allowed_Movers(i,1),2);
end

%% Let the five households move to a random unoccupied house

Moves_To=zeros(5,2);
Unoccupied_NowOccupied=zeros(5,1);
i=1;

while i<Movers_Counter

    Row=randi(Movers_Counter-1);
    if ismember(Row,Unoccupied_NowOccupied)==0
    Unoccupied_NowOccupied(i,1)=Row;

    Moves_To(i,1)=Non_Occupied(Row,1);
```

```
    Moves_To(i,2)=Non_Occupied(Row,2);
    i=i+1;
    end

end

%Set new location of selected movers to zero, set unoccupied houses ...
    to new
%values

for i=1:Movers_Counter-1
    Board(Non_Occupied(i,1),Non_Occupied(i,2))=Board(Allowed_Movers_Pos(i,1),Allowed_Movers_Pos(i,2)
    Board(Allowed_Movers_Pos(i,1),Allowed_Movers_Pos(i,2))=0;
end

Iteration_Count=Iteration_Count+1;
Title= ['Iteration' num2cell(Iteration_Count)];

if ij==15
    subplot(2,2,2)
    Plot_1=imagesc(Board);
    title(Title)
elseif ij== 30
    subplot(2,2,3)
    Plot_2=imagesc(Board);
    title(Title)
elseif ij== 45
    subplot(2,2,4)
    Plot_3=imagesc(Board);
    title(Title)
end

end
```