Answers to Problem Set 1
Group name: Ferienspass

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

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

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

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!']);
```

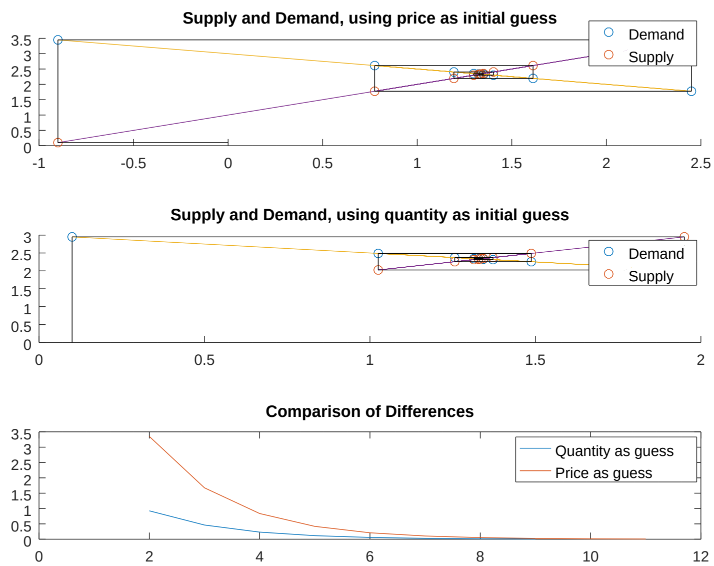
Gauss-Seidel Iteration Result (using quantity as initial guess): The market clearing price 2.3357 clears the market at the quantity 1.3357 after 9 iterations!

```
%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
```

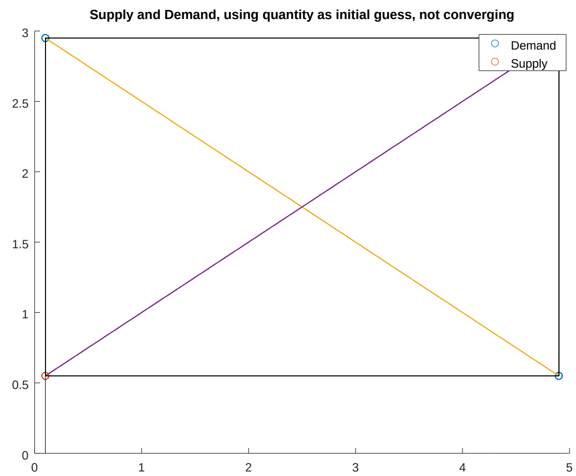



```
q=0.1;
%Set up difference criterion to a value higher than in the while loop
nq_difference=100;

%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
```