September 25, 2018

Empirircal Methods HA 2

Konstantin Guryev

Pennsylvania State University

2018

Problem 1

$$\begin{pmatrix} D_A \\ D_B \\ D_0 \end{pmatrix} = \begin{pmatrix} 0.4223 \\ 0.4223 \\ 0.1554 \end{pmatrix};$$

Problem 2

The starting values are: $p_A = p_B = 2$; $v_A = v_B = 2$. Nash pricing equilibrium: $p_A = p_B = 1.598942$. Number of iterations needed for convergence is equal to 5 for the chosen starting values. Elapsed time (for 5 compilations) varied from 0.009430 to 0.064949 seconds. I tried other starting values for p, but the numerical solution for NE vector of prices is the same. Tolerance level is set to be equal to $1e - 10$.

Problem 3

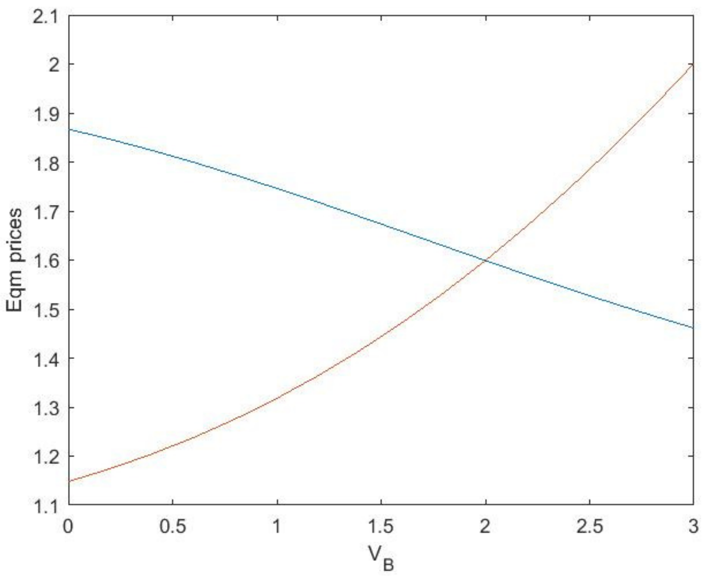
The Gauss-Seidel method converges in 40 iterations, but the elapsed time (for 5 compilations) varied from 0.010673 to 0.019461. The equilibrium price values are the same as for Broyden's method. We can conclude that The G-S method is on average faster than Broyden's method. The latter approximates the whole Jacobian whereas G-S does not - that is the reason why G-S is faster.

Problem 4

The proposed updating rule method converges in 18 iterations an the elapsed time (for 5 compilations) varied from 0,007514 to 0,008043. So we can conclude that this method is faster than the other two methods. The equilibrium price values are the same as in the previous calculations.

Problem 5

I solved the task using Broyden's method. The results are on the graph:



Matlab Code

```
function [fval, Jac]=FOC(p,v)
%Here we have fval = -Di(p)/D'i(p)-pi, i.e. the FOC of the profit function,
% where i=A,B.

fval=[(1+exp(v(1)-p(1))+exp(v(2)-p(2)))/(1+exp(v(2)-p(2)))-p(1);(1+exp(v(1)-p(1))+exp(v(2)-p(2)))/(1+exp(v(1)-p(1)))-p(2)];
Jac=[(-exp(v(1)-p(1)))/(1+exp(v(2)-p(2)))-1 (exp(sum(v)-sum(p)))/(1+exp(v(2)-p(2)))^2;...
        (exp(sum(v)-sum(p)))/(1+exp(v(1)-p(1)))^2 (-exp(v(2)-p(2)))/(1+exp(v(1)-p(1))) -1];
end
```

Matlab Code

```
function p1=br(v)
p=[2;2];
tol=1e-10;
maxit=10000;
[fval, iJac_0]=cournot_hw2(p,v);
iJac=inv(iJac_0);
for i=1:maxit
    if norm(fval) < tol
    fprintf('i: %d P_A = %f, P_B = %f, norm(f(x)) = %.6f\n', i, p(1)
        , p(2), norm(fval));
    break
    end
   d = -(iJac*fval);
   p = p + d;
   f_prev = fval;
   fval=cournot_hw2(p,v);
   u=iJac*(fval-f_prev);
   iJac=iJac+((d-u)*(d'*iJac))/(d'*u);
end
p1=p;
end
```

Matlab Code

```
clear;


%% Problem 1.
D = @(p) exp(2-p.*1)/(sum(exp(2-p.*1))); % demand system
Demand = D([1;1;2]);

%% Problem 2.
p=[2;2];
v=[2;2];
tol=1e-10;
maxit=10000;
[fval, iJac_0]=FOC(p,v);
iJac=inv(iJac_0);
tic
for i1=1:maxit
    fnorm = norm(fval);
    if norm(fval) < tol
    fprintf('i1 %d: P_A = %f, P_B = %f, norm(f(x)) = %.6f\n', i1, p
        (1), p(2), norm(fval));
    break
    end
   d=-(iJac*fval);
   p=p+d;
   f_prev=fval;
   fval=FOC(p,v);
   u=iJac*(fval-f_prev);
   iJac=iJac+((d-u)*(d'*iJac))/(d'*u);
end
toc

%% Problem 3.

```

```
p3=[2;2];
p3_prev=[3,3];
f_prev=FOC(p3_prev,v);
tic
for i2=1:maxit
    fval=FOC(p3,v);
    if norm(fval)<tol
    fprintf('i2: %d, P_A: %f, P_B: %f, norm(f(p)): %.6f\n',i2,p3(1),
        p3(2),norm(fval));
    break
    else
        p3_A=p3(1)-(p3(1)-p3_prev(1))*fval(1)/(fval(1)-f_prev(1));
        dGS=FOC([p3_A;p3(2)],v);
        p3_B=p3(2)-(p3(2)-p3_prev(2))*dGS(2)/(fval(2)-f_prev(2));
        p3_prev=p3;
        f_prev=fval;
        p3=[p3_A;p3_B];
    end
end
toc

%% Problem 4.
p4=[2;2];
p4_prev=[3;3];
D4=@(p)[(exp(2-p(1)))/(1+exp(2-p(1))+exp(2-p(2)));(exp(2-p(2)))/(1+
    exp(2-p(1))+exp(2-p(2)))];
tic
for i3=1:maxit
    fnorm=norm(FOC(p4,v));
    p4norm=norm(p4-p4_prev);
    if p4norm<tol
    fprintf('i3 %d: P_A = %f, P_B = %f, norm(f(x)) = %.6f\n', i3, p4
        (1), p4(2), fnorm);
    break
    end
```

```
        p4_prev=p4;
        p4_iter=(ones(2,1)-D4(p4)).^(-1);
        p4=p4_iter;
end
toc

%% Problem 5.
v_B=0:0.2:3;
v_A=repmat(2,1,length(v_B));
v=[v_A;v_B]; %2 by 16 Matrix
price=ones(2,length(v_B));
for i4=1:length(v_B)
        price(:,i4)=br(v(:,i4));
end
plot(v(2,:),price(1,:),v(2,:),price(2,:));
xlabel('V_B');
ylabel('Eqm prices');
```