Homework 2

Kensuke Suzuki

September 20, 2018

The elapsed times described below may not coincides with the ones in the log file—They may change every time I run the command, but the relative size between algorithms does not affected.

Problem 1

I define a function `bertrand.m` which returns vector of demand for each good, for given vector of price $\mathbf{p}$ and $\mathbf{v}$ (which are potentially $n \times 1$ vectors).

```
function fval = bertrand(p,v)

% for given vector of p and v, solve demand
fval = exp(v - p) ./ ( 1 + sum( exp(v - p) ) );

end
```

For $\mathbf{p} = [1, 1]^\top$ and $\mathbf{v} = [2, 2]^\top$, we obtain $[D_A, D_B]^\top = [0.422319, 0.422319]^\top$ and $D_0 = 1.55362$.

Problem 2

Each firm solves profit maximization problem:

$$\max_{p_i} p_i D_i$$

for $i = A, B$. The first order conditions yields:

$$D_i + \left[\frac{\partial D_i}{\partial p_i} p_i\right] = D_i - p_i D_i(1 - D_i) = D_i \left[1 - p_i(1 - D_i)\right] = 0$$

Provided $D_i \neq 0$, the FOC boils down to $[1 - p_i(1 - D_i)] = 0$. The Bertrand-Nash equilibrium is the set of prices which satisfy the system of equation:

$$1 - p_A(1 - D_A) = 0$$
$$1 - p_B(1 - D_B) = 0$$

In matrix notation, we have

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 1 - D_A & 0 \\ 0 & 1 - D_B \end{bmatrix} \begin{bmatrix} p_A \\ p_B \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{1}$$

We define the LHS of (1) as a function `bertrandfoc.m`

```
function fval = bertrandfoc(p,v)

% for given vector of p and v, solve demand
D = exp(v - p) ./ ( 1 + sum( exp(v - p) ) );

% First order condition boils down to
fval = ones(size(p,1),1) - diag(ones(size(p,1),1)-D)*p;

end
```

We then solve the system of nonlinear equation using Broyden's Method. The algorithm is completely analogue to what we did in the class. For the starting value of $\mathbf{p}$, we use $[1, 1]^\top$ and we use the identity matrix as an initial inverse of Jacovian. For convergence criterion, we use 1e-6. The iteration converges and we get the set of equilibrium prices $\mathbf{p} = [1.598942, 1.598942]^\top$

Problem 3

First we define the function `bertrandfocg.m` which return the FOC for gth good price.

```
function fval = bertrandfocg(p,v,g)

% for given vector of p and v, solve demand
D = exp(v - p) ./ ( 1 + sum( exp(v - p) ) );

% First order condition boils down to
foc = ones(size(p,1),1) - diag(ones(size(p,1),1)-D)*p;

fval = foc(g,1);

end
```

We then solve the system by using a Gauss-Seidel method. First we set the initial value for $\mathbf{p} = [1, 1]^\top$ and set $\mathbf{p}_{\text{old}} = [2, 2]$. Then, for given p_B we solve the FOC for good A price using the secant method. This sub-iteration solves p_A for given initial guess on p_B. Then next sub-iteration solves p_B using the FOC for good B price using the p_A obtained in the previous sub-iteration.

$$\text{(1) } p_A^{k+1} \Leftarrow 1 - p_A^k(1 - D_A(p_A^k, p_B^k)) = 0$$
$$\text{(2) } p_B^{k+1} \Leftarrow 1 - p_B^k(1 - D_B(p_A^{k+1}, p_B^k)) = 0$$

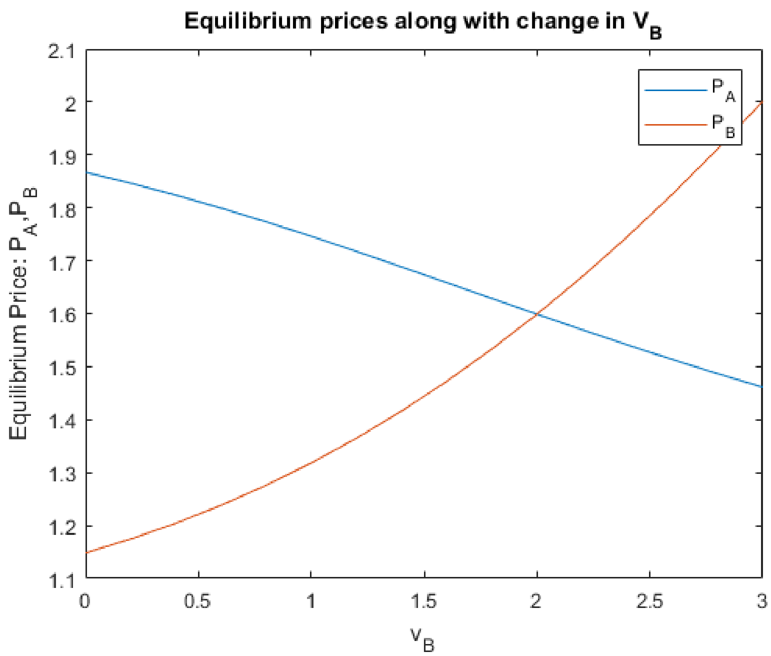
We iterate the set of two subiteration (indexed by k above) until the norm of the vector obtained using the function `bertrandfoc.m` (which returns the vector of the LHS of FOCs) is below the tolerance level. This algorithm also yields the same equilibrium price vector $\mathbf{p} = [1.598942, 1.598942]^\top$. The elapsed time for problem 2 (Broyden Method) is 0.005407 and the one for problem 3 (Gauss-Seidel Method) is 0.014861. Therefore, Gauss-Seidel Method is slower. **WHY:** intuitively it is because the Gauss-Seidel method does not update the p_A and p_B at once, rather, it solves two equations separately. More precisely, the Gauss-Seidel method solves for the first equation (FOC for firm A) for given price of p_B (step 1) and then solves for the second equation (FOC for firm B) p_B for given price of p_A (step 2). Each step entails iteration. Of course, p_A obtained in the first step is not necessarily the best response for given p_B obtained in the subsequent step. Therefore, we need to iterate this set of two steps until it converges. So there exists "double loop," which make the algorithm slower than Broyden.

Problem 4

In this problem, we use the update rule specified in the problem set and solve the system. It again yields the same result: $\mathbf{p} = [1.598942, 1.598942]^\top$. Time elapsed was 0.007433.

Problem 5

We define the vector of v_B values (0:.2:2) and for each v_B and $v_A = 2$, we solve the system of equation for $\mathbf{p}$. We store the equilibrium vector of prices in `result` matrix (where the first raw is v_B and the second and third raws contain the vector of equilibrium prices corresponding to each v_B value). 2-way plotted graph is demonstrated below.



Matlab Code

```
% ECON512 Homework 2
% Kensuke Suzuki
clear all
delete HW2log.txt
diary('HW2log.txt')
diary on

disp('ECON512 HOMEWORK2: Ken Suzuki')

%% Define bertrand and bertrandfoc function
% bertrand: return demand for each good
% bertrandfoc: return system of FOC (LHS)

%% Problem 1

p = [1;1];
v = [2;2];

Ans1 = bertrand(p,v)

D0 = 1 / (1+ sum(exp(v-p)) );

P1 = sprintf('Problem1: for vA=vB=2 and pA=pB=1, DA= %f, DA= %f, and D0 = %f.', Ans1(1,1),Ans1(2,1),D0);
disp(P1);

```

```
%% Problem 2

clear all

v = [2;2];

p = [1;1];

fVal_foc = @(p) bertrandfoc(p,v);
i_fVal_foc = fVal_foc(p);

iJac = eye(size(p,1));

maxit = 100;
tol = 1e-6;

tic
for iter = 1:maxit
    fnorm = norm(i_fVal_foc);
    fprintf('iter %d: p(1)=%f, p(2)=%f, norm(f(x))=%.8f \n', iter, p(1),p(2),norm(i_fVal_foc));
    if norm(i_fVal_foc)<tol
        break
    end
    d = -(iJac*i_fVal_foc);
    p = p + d;
    fOld_foc = i_fVal_foc;
    i_fVal_foc = fVal_foc(p);
    u = iJac*(i_fVal_foc - fOld_foc);
    iJac = iJac + ( (d-u)* (d'*iJac) )/(d'*u);
end
elapsedTime_p2 = toc;

P2 = sprintf('Problem2: for vA=vB=2, equilibrium prices are: PA= %f, PB = %f; time elapsed is %f.', p(1,1),p(2,1), elapsedTime_p2);
disp(P2);



%% Problem 3

clear all

v = [2;2];
p = [1;1];

fVal_foc = @(p) bertrandfoc(p,v);
fVal_focg = @(p,g) bertrandfocg(p,v,g);
```

```
maxit = 100;
tol = 1e-6;

tic
for iter = 1:maxit

    fval = fVal_foc(p);
    if norm(fval) < tol
        break
    end
    fprintf('iter %d: p(1)=%f, p(2)=%f, norm(f(x))=%.8f \n', iter, p(1)
       ,p(2),norm(fval));

    % set pOld
    pOld = [2;2];
    pA_Old = pOld(1,1);

    % compute the LHS of FOC for good A price
    fOld_1 = fVal_focg(pOld,1);

    % for given pB, solve the first equation for pA
    % We use Secant Method
    for iter_1 =1:maxit
        fval_1 = fVal_focg(p,1);
        if abs(fval_1) < tol
            break
        else
            pA_New = p(1,1) - ( (p(1,1) - pA_Old) / (fval_1 - fOld_1) )
               * fval_1;
            pA_Old = p(1,1);
            p(1,1) = pA_New;
            fOld_1 = fval_1;
        end
    end

    % Use the solution for pA obtained above, solve for pB
    pB_Old = pOld(2,1);
    fOld_2 = fVal_focg(pOld,2);
    for iter_2 = 1:maxit
        fval_2 = fVal_focg(p,2);
        if abs(fval_2) < tol
            break
        else
            pB_New = p(2,1) - ( (p(2,1) - pB_Old) / (fval_2 - fOld_2) )
               * fval_2;
            pB_Old = p(2,1);
            p(2,1) = pB_New;
```

```
            fOld_2 = fval_2;
        end
    end

end
elapsedTime_p3 = toc;

P3 = sprintf('Problem3: for vA=vB=2, equilibrium prices are: PA= %f, PB = %f; time elapsed is %f.', p(1,1),p(2,1), elapsedTime_p3);
disp(P3);

%% Problem 4

clear all

v = [2;2];

p = [1;1];

fVal_bertrand = @(p) bertrand(p,v);
fVal_foc = @(p) bertrandfoc(p,v);
i_fVal_foc = fVal_foc(p);

maxit = 100;
tol = 1e-6;

tic
for iter = 1:maxit
    fnorm = norm(i_fVal_foc);
    fprintf('iter %d: p(1)=%f, p(2)=%f, norm(f(x))=%.8f \n', iter, p(1),p(2),norm(i_fVal_foc));
    if norm(i_fVal_foc)<tol
        break
    end
    p_next = 1./ ([1;1] - fVal_bertrand(p) );
    p = p_next;
    i_fVal_foc = fVal_foc(p);
end
elapsedTime_p4 = toc;

P4 = sprintf('Problem4: for vA=vB=2, equilibrium prices are: PA= %f, PB = %f; time elapsed is %f.', p(1,1),p(2,1), elapsedTime_p4);
disp(P4);

%% Problem 5

clear all

```

```
vB_5 = [0:.2:3];
v_5 = [2*ones(1,size(vB_5,2));vB_5 ];
result = [vB_5; ones(1,size(vB_5,2)); ones(1,size(vB_5,2)) ];

for vindex = 1:size(vB_5,2)

    p = [1;1];
    v = v_5(:,vindex);

    fVal_foc = @(p) bertrandfoc(p,v);
    i_fVal_foc = fVal_foc(p);
    iJac = eye(size(p,1));

    maxit = 100;
    tol = 1e-6;

    for iter = 1:maxit
        fnorm = norm(i_fVal_foc);
        %fprintf('iter %d: p(1)=%f, p(2)=%f, norm(f(x))=%.8f \n', iter,
            p(1),p(2),norm(i_fVal_foc));
        if norm(i_fVal_foc)<tol
            break
        end
        d = -(iJac*i_fVal_foc);
        p = p + d;
        fOld_foc = i_fVal_foc;
        i_fVal_foc = fVal_foc(p);
        u = iJac*(i_fVal_foc - fOld_foc);
        iJac = iJac + ( (d-u)* (d'*iJac) )/(d'*u);
    end
    result(2,vindex) = p(1);
    result(3,vindex) = p(2);
end

plot(vB_5,result(2,:),vB_5,result(3,:))
title('Equilibrium prices along with change in V_B')
xlabel('v_B')
ylabel('Equilibrium Price: P_A,P_B')
legend('P_A','P_B')

diary off
```