Online Appendix: Procurement when both price and quality matter

Code

This contains contains the matlab code used to construct simulations.

```
% This code is used to create the simulations for the optimal auctions
% paper

% Note this code is writen so the the types are called a (=lH), b (=hH), c
% (=hL) and d (=lL)

% You will need to make some new m-files (functions) before you can use
% this code. The code for these functions is in the commented stuff at the
% very end of this m-file.

% one of the things that will make this code look like it is doing
% something wierd is if you hit the exact border between two cases. this is
% the first thing to look for when diagnosing the cause of a spike in the
% outputed revenue. once identified it is easy fixed by dividing the
% offending number by (usually) two.

% some other problems exist with the way this code is written. As it is
% written, in some settings lambdas may not exist. This is a computational
% issue and not a probem with the proofs in the paper. This does not affect
% any of the simulations reported in the paper.


clear all
global x_bar lambda N x_b x qaz qbz fbx_d t2_H t1_H t1_L t2_L alpha_c alpha_d
alpha_b alpha_a delta2 delta2 a b
rand('state',201)
simsize = 201;

% Recall costs are theta1 + theta2*q
% The valuation of the quality for the buyer is V(q)=3*q^(1/2)
a = 3;
b = 1/2;
% V = a*q^(b);


% need to pick the probs of individual types
alpha_a = 15/100;
alpha_b = 35/100;
alpha_c = 35/100;
alpha_d = 15/100;

% note whether they satisfy the hazard conditions: d/a<(a+d)/b
% d/c<(c+d)/b
if alpha_d./alpha_c > (alpha_c+alpha_d)/alpha_b
    disp('Hazard Rate 1 violated')
end
if alpha_d./alpha_a > (alpha_a+alpha_d)/alpha_b
    disp('Hazard Rate 2 violated')
end

%need to set the number of bidders
N = 2;
```

```
% Pick the types
t1_L = 1;
t2_L = 1;
delta1 = [0.00001; (1/200).*[1:200]'.*1.125];
delta2 = 1.*ones(simsize,1);
t1_H = 1 + delta1;
t2_H = 1 + delta2;

% define first best qualities

fbq_a = (t2_H/(a*b)).^(1/(b-1));
fbq_b = fbq_a;
fbq_c = (t2_L/(a*b)).^(1/(b-1));
fbq_d = fbq_c;

% define first best probs of winning

fbx_a = ((1-alpha_d-alpha_c).^N-(1-alpha_d-alpha_c-alpha_a).^N)/(N*alpha_a);
fbx_b = ((1-alpha_d-alpha_c-alpha_a).^N-(1-alpha_d-alpha_c-alpha_a-
alpha_b).^N)/(N*alpha_b);
fbx_c = ((1-alpha_d).^N-(1-alpha_d-alpha_c).^N)/(N*alpha_c);
fbx_d = ((1).^N-(1-alpha_d).^N)/(N*alpha_d);

% define the first best welfares

fbW_a =  a*fbq_a.^(b) - t1_L - t2_H.*fbq_a;
fbW_b =  a*fbq_b.^(b) - t1_H - t2_H.*fbq_b;
fbW_c =  a*fbq_c.^(b) - t1_H - t2_L.*fbq_c;
fbW_d =  a*fbq_d.^(b) - t1_L - t2_L.*fbq_d;

% define qb^2 & qa^2
x = t2_H + delta2*(alpha_c + alpha_d)/alpha_b;
q_b2 = (x/(a*b)).^(1/(b-1));
x = t2_H + delta2*(alpha_c + alpha_d)/alpha_a;
q_a2 = (x/(a*b)).^(1/(b-1));

% and define the qb^2 & qa^2 welfares

W_a_q_b2 =  a*q_b2.^(b) - t1_L - t2_H.*q_b2;
W_b_q_b2 =  a*q_b2.^(b) - t1_H - t2_H.*q_b2;
W_c_q_b2 =  a*q_b2.^(b) - t1_H - t2_L.*q_b2;
W_d_q_b2 =  a*q_b2.^(b) - t1_L - t2_L.*q_b2;

W_a_q_a2 =  a*q_a2.^(b) - t1_L - t2_H.*q_a2;
W_b_q_a2 =  a*q_a2.^(b) - t1_H - t2_H.*q_a2;
W_c_q_a2 =  a*q_a2.^(b) - t1_H - t2_L.*q_a2;
W_d_q_a2 =  a*q_a2.^(b) - t1_L - t2_L.*q_a2;

% and define the fbq_c welfares

W_a_fbq_b =  a*fbq_b.^(b) - t1_L - t2_H.*fbq_b;
W_b_fbq_b =  a*fbq_b.^(b) - t1_H - t2_H.*fbq_b;
W_c_fbq_b =  a*fbq_b.^(b) - t1_H - t2_L.*fbq_b;
W_d_fbq_b =  a*fbq_b.^(b) - t1_L - t2_L.*fbq_b;
```

```
W_a_fbq_c =  a*fbq_c.^(b) - t1_L - t2_H.*fbq_c;
W_b_fbq_c =  a*fbq_c.^(b) - t1_H - t2_H.*fbq_c;
W_c_fbq_c =  a*fbq_c.^(b) - t1_H - t2_L.*fbq_c;
W_d_fbq_c =  a*fbq_c.^(b) - t1_L - t2_L.*fbq_c;

% Now, there are lots of solutions that apply to different bits of the parameter
% space. here it intialise the triggers for each solution so that we can keep
track
%  of what applies. this helps keep redundant coding to a minimum
x = zeros(simsize,1);
key11a = x;
key11b = x;
key11c = x;
key11d = x;
key11e = x;
key12a = x;
key12b = x;
key12c = x;
key21a = x;
key21b = x;
key21c = x;
key21d = x;
key21e = x;
key21de = x;
key12c21de = x;
key11c11d11e = x;
key12c11c11d11e = x;
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
% Optimal Mechanism

% Prelim for Part II

% Buyer Optimal Efficient Mechanism: for part 2

x = (alpha_a.*fbx_a).*(fbW_a + delta1.*(alpha_c + alpha_d)./alpha_a -
fbq_a.*delta2*(alpha_c + alpha_d)./alpha_a);
x = x + (alpha_b.*fbx_b).*(fbW_b - delta1.*(alpha_c + alpha_d + alpha_a)./alpha_b
);
x = x + (alpha_c.*fbx_c).*(fbW_c - delta1.*alpha_d./alpha_c );
rev_BOEM = x + (alpha_d.*fbx_d).*fbW_d;

% Define the class that they fall into

% condition1 = 1 if fbW_c > fbW_a
condition1 = sign(sign(fbW_c - fbW_a)+1);

% condition2 = 1 if W_a_fbq_b - W_c_fbq_b < 0
condition2 = sign(sign(W_c_fbq_b - W_a_fbq_b)+1);

key_BOEM = condition1.*condition2;
```

```
rev_BOEM = rev_BOEM.*key_BOEM;

%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
% Prelim for Part II Senario 2
% I do senario 2 first as it refers to solutions in senario 1 and pt1 senario 2
extensively -
% this is the most efficient way to code the solution

% condition3 = 1 if fbx_a*(W_a(qa^2)-W_c(qa^2)) > fbx_b*(W_a(fbq_b)-W_c(fbq_b))
condition3 = 1 - sign(sign(fbx_b.*(W_a_fbq_b - W_c_fbq_b) - fbx_a.*(W_a_q_a2 -
W_c_q_a2))+1);

key = condition1.*condition2.*condition3;

tag = find(key);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);

    L1 = 0;
    lambda = L1;
    qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_c_qaz =  a*qaz.^(b) - t1_H(x,:) - t2_L.*qaz;
    W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
    W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
    LHS = fbx_a.*(W_a_qaz-W_c_qaz);
    RHS = fbx_b.*(W_a_qbz-W_c_qbz);
    click1 = (abs(LHS - RHS));

    L2 = alpha_c + alpha_d;
    lambda = L2;
    qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_c_qaz =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
    W_a_qbz =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
    W_c_qbz =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
    LHS = fbx_a.*(W_a_qaz-W_c_qaz);
    RHS = fbx_b.*(W_a_qbz-W_c_qbz);
    click2 = (abs(LHS - RHS));

    count = 1;
```

```
    click =
100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
    while click > 0.000000000001;

        if count > 100
            disp(['click is:   ' num2str(click)])
            disp(['At 64732'])
            keyboard
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_c_qaz =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
        W_a_qbz =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
        W_c_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_qaz =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        W_a_qbz =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
        W_c_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_qaz =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        W_a_qbz =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
        W_c_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
```

```
            click = click3;
            qaz = qaz3;
            qbz = qbz3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
            qaz = qaz4;
            qbz = qbz4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
            qaz = qaz5;
            qbz = qbz5;
        end
    end
    lambdas(x,1) = lambda;
    qazs(x,1) = qaz;
    qbzs(x,1) = qbz;
end

W_a_qaz =  a*qazs.^(b) - t1_L - t2_H.*qazs;
W_b_qbz =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;

% condition4 = 1 if W_c_fbq_c - delta1*(alpha_d/alpha_c) > W_a_qaz+(delta1-
qazs*delta2)*(alpha_c + alpha_d)/alpha_a
condition4 = sign(sign(W_c_fbq_c - delta1.*(alpha_d/alpha_c) - W_a_qaz - (delta1 -
qazs.*delta2).*(lambdas)./alpha_a)+1);

% condition5 = 1 if W_a_qaz + (delta1 - qazs*delta2)*(alpha_c + alpha_d)/alpha_a >
fbW_b - delta1*(alpha_c + alpha_d + alpha_a)/alpha_b);
condition5 = sign(sign(W_a_qaz + (delta1 - qazs.*delta2).*(lambdas)./alpha_a -
W_b_qbz + delta1.*(lambdas + alpha_a)./alpha_b+ qbzs.*delta2.*(-lambdas + alpha_c
+ alpha_d)./alpha_b)+1);

% condition6 = 1 if W_c_fbq_c - delta1*(alpha_d/alpha_c) > fbW_b - delta1*(alpha_c
+ alpha_d + alpha_a)/alpha_b);
condition6 = sign(sign(W_c_fbq_c - delta1.*(alpha_d/alpha_c) - W_b_qbz +
delta1.*(lambdas + alpha_a)./alpha_b+ qbzs.*delta2.*(-lambdas + alpha_c +
alpha_d)./alpha_b)+1);

%*************************************************************************
*************************************************************************
% Part II Senario 2 Solution 1.2.a
% condition for this applying is key(element) = 1;

key = condition1.*condition2.*condition3.*condition4.*condition5;
key12a = key + key12a;
% the rest is continued when Solution 1.2.a arises below

%*************************************************************************
```

```
 ********************************************************************************
 % Part II senario 2
 % Case2: VWa > VWc > VWb  or  VWa > VWb > VWc
 % condition for this applying is key(element) = 1;

 key = condition1.*condition2.*condition3.*(1-condition4);

 % there are a variety of soultions that could apply in this setting. 1.2.b,
 % 1.2.c, 2.1.d or 2.1.e. We need to work out which solutions apply
 % The approach that i take it to compute the lambda2's for each of the
 % three situations that we can find ourselves in and then choose the one
 % closest to the starting lambda2. note the this original lambda 2 is used
 % to choose the range that i search over to find these lambdas

 signkey = sign(sign(delta1 - delta2.*qazs)+1); % if this = 1 then we are
 decreasing lambda2 from lambdas set above

 %_____________________________________________________________
 %_____________________________________________________________
 % Work out where VWa = VWc

 tag = find(key);
 lambdas_a = zeros(simsize,1);
 lambdas_b = lambdas_a;
 lambdas_c = lambdas_a;
 qazs = lambdas_a;
 qbzs = lambdas_a;
 W_a_zs= lambdas_a;
 tags = size(tag,1);
 for i = 1:tags

     x = tag(i);

     L11 = 0 + (alpha_c + alpha_d).*(1-signkey(x,:));
     L22 = (lambdas(x,:));

     lambda = L11;
     qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
     LHS1 = W_a_qaz + lambda*delta1(x,:)/alpha_a - lambda*delta2(x,:)*qaz/alpha_a;
     RHS = W_c_fbq_c(x,:) - delta1(x,:).*(alpha_d/alpha_c);
     click1 = (abs(LHS1 - RHS));

     lambda = L22;
     qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
     LHS2 = W_a_qaz + lambda*delta1(x,:)/alpha_a - lambda*delta2(x,:)*qaz2/alpha_a;
     click2 = (abs(LHS2 - RHS));
     L1 = L11;
     L2 = L22;
     count = 1;
     click = 10000000000;
     if sign(LHS1 - RHS) == sign(LHS2 - RHS)
         click = 0;
```

```
        lambda = L1;
    end
    while click > 0.000000000001;

        if count > 200
            disp(['click is:   ' num2str(click)])
            disp(['At 44399'])
            keyboard
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
lambda*delta2(x,:)*qaz3/alpha_a;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
lambda*delta2(x,:)*qaz4/alpha_a;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
lambda*delta2(x,:)*qaz5/alpha_a;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas_a(x,1) = lambda;
end
```

```
 for i = 1:tags

     x = tag(i);

     L1 = 0 + (alpha_c + alpha_d).*(1-signkey(x,:));
     L2 = (lambdas(x,:));

     lambda = L1;
     qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
     W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
     LHS1 = W_a_qaz + lambda*delta1(x,:)/alpha_a - lambda*delta2(x,:)*qaz/alpha_a;
     RHS1 = W_b_qbz - (alpha_a + lambda)*delta1(x,:)/alpha_b - (alpha_c + alpha_d -
 lambda)*delta2(x,:)*qbz/alpha_b;
     click1 = (abs(LHS1 - RHS1));

     lambda = L2;
     qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
     W_b_qbz =  a*qbz2.^(b) - t1_H(x,:) - t2_H(x,:).*qbz2;
     LHS2 = W_a_qaz + lambda*delta1(x,:)/alpha_a - lambda*delta2(x,:)*qaz2/alpha_a;
     RHS2 = W_b_qbz - (alpha_a + lambda)*delta1(x,:)/alpha_b - (alpha_c + alpha_d -
 lambda)*delta2(x,:)*qbz2/alpha_b;
     click2 = (abs(LHS2 - RHS2));

     count = 1;
     click = 10000000000;
     if sign(LHS1 - RHS1) == sign(LHS2 - RHS2)
         click = 0;
         lambda = L1;
     end
     while click > 0.000000000001;

         if count > 100
             disp(['click is:   ' num2str(click)])
             disp(['At line 727'])
             keyboard
         end
         count = count +1;

         L3 = (2/3)*L1 + L2/3;
         lambda = L3;
         qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
         W_a_qaz =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
         W_b_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_H(x,:).*qbz3;
         LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
 lambda*delta2(x,:)*qaz3/alpha_a;
```

```
        RHS = W_b_qbz - (alpha_a + lambda)*delta1(x,:)/alpha_b - (alpha_c +
 alpha_d - lambda)*delta2(x,:)*qbz3/alpha_b;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));

        qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_b_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_H(x,:).*qbz4;
        LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
 lambda*delta2(x,:)*qaz4/alpha_a;
        RHS = W_b_qbz - (alpha_a + lambda)*delta1(x,:)/alpha_b - (alpha_c +
 alpha_d - lambda)*delta2(x,:)*qbz4/alpha_b;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_b_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_H(x,:).*qbz5;
        LHS = W_a_qaz + lambda*delta1(x,:)/alpha_a -
 lambda*delta2(x,:)*qaz5/alpha_a;
        RHS = W_b_qbz - (alpha_a + lambda)*delta1(x,:)/alpha_b - (alpha_c +
 alpha_d - lambda)*delta2(x,:)*qbz5/alpha_b;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas_b(x,1) = lambda;
 end

 for i = 1:tags

    x = tag(i);
```

```
    x_a = (1 - N*alpha_d*fbx_d - N*alpha_b*fbx_b)./(N*alpha_a + N*alpha_c);

    L1 = 0 + (alpha_c + alpha_d).*(1-signkey(x,:));
    L2 = (lambdas(x,:));

    lambda = L1;
    qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_c_qaz =  a*qaz.^(b) - t1_H(x,:) - t2_L.*qaz;
    W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
    W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
    LHS1 = x_a.*(W_a_qaz-W_c_qaz);
    RHS1 = fbx_b.*(W_a_qbz-W_c_qbz);
    click1 = (abs(LHS1 - RHS1));

    lambda = L2;
    qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_c_qaz =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
    W_a_qbz =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
    W_c_qbz =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
    LHS2 = x_a.*(W_a_qaz-W_c_qaz);
    RHS2 = fbx_b.*(W_a_qbz-W_c_qbz);
    click2 = (abs(LHS2 - RHS2));

    count = 1;
    click = 10000000000;
    if sign(LHS1 - RHS1) == sign(LHS2 - RHS2)
        click = 0;
        lambda = L1;
    end

    while click > 0.000000000001;

        if count > 100
            disp(['click is:   ' num2str(click)])
            disp(['At line 4545'])
            keyboard
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_c_qaz =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
        W_a_qbz =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
        W_c_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
```

```
        LHS = x_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_qaz =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        W_a_qbz =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
        W_c_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
        LHS = x_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_qaz =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        W_a_qbz =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
        W_c_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
        LHS = x_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas_c(x,1) = lambda;
end

% now need to work out the abs distance lambda and set the keys

select_sln = [abs(lambdas - lambdas_a).*key abs(lambdas - lambdas_b).*key
abs(lambdas - lambdas_c).*key];
```

```
select_min = [min(select_sln,[],2)]*[1 1 1];
keys = select_sln == select_min;
keys = [keys(:,1).*key keys(:,2).*key keys(:,3).*key];

key12b = keys(:,1) + key12b;
key21de = keys(:,2) + key21de;
key12c21de = keys(:,3) + key12c21de;

%*************************************************************************
*************************************************************************
% Part 2 Senario 2 VWc > VWb > VWa
% this is Solution 2.1.c
key = condition1.*condition2.*condition3.*condition6.*(1-condition5);
key21c = key + key21c;
%the rest of this case is dealt with below in part 2 senario 1
%*************************************************************************
*************************************************************************
% Part 2 Senario 2 VWb > VWa > VWc
% this is Solution 2.1.d or 2.1.e
key = condition1.*condition2.*condition3.*(1-condition4).*(1-condition5);
key21de = key + key21de;
%the rest of this case is dealt with below in part 2 senario 1
%*************************************************************************
*************************************************************************
% Part 2 Senario 2 VWb > VWc > VWa
% this is Solution 2.1.d or 2.1.e or 2.1.c
key = condition1.*condition2.*condition3.*(1-condition4).*(1-condition5);
key21c = key + key21c;
% note that the coding for 2.1.c will sort the 2.1.d and 2.1.e cases
% through to that solution.
%the rest of this case is dealt with below in part 2 senario 1
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
%*************************************************************************
*************************************************************************
% Should do part 1 next since we refer to it in above
% start with pt2s2 since it will have to refer to pt2 s1

% Prelim for Part I

% Define the class that they fall into

% condition1 = 1 if fbW_c > fbW_a
condition1 = sign(sign(fbW_c - fbW_a)+1);

% keyboard
```

```
% condition2 = 1 if W_a_fbq_b - W_c_fbq_b > 0
condition2 = sign(sign(W_a_fbq_b - W_c_fbq_b)+1);

key_BOEM = condition1.*condition2;
x = (alpha_a.*fbx_a).*(fbW_a);
x = x + (alpha_b.*fbx_b).*(fbW_b - delta1.*(alpha_a)./alpha_b -
fbq_b.*delta2.*(alpha_c + alpha_d)./alpha_b );
x = x + (alpha_c.*fbx_c).*(fbW_c - delta1.*alpha_d./alpha_c );
x = (x + (alpha_d.*fbx_d).*fbW_d);
rev_BOEM = rev_BOEM + x.*key_BOEM;

% condition3 = 1 if fbx_b*(W_a(qb^2)-W_c(qb^2)) > fbx_a*(W_a(fbq_a)-W_c(fbq_a))
condition3 = sign(sign(fbx_b.*(W_a_q_b2 - W_c_q_b2) - fbx_a.*(fbW_a - W_c_fbq_b)
)+1);

% now need to work out lambda2* and the q's that go with it

key = condition1.*condition2.*condition3;

% note that we use the lambdas out of the loop below for computing sln
% 1.2.a so have to compute it over the whole range hence tag = key + key12a;

tag = find(key + key12a);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_z= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);

    L1 = 0;
    lambda = L1;
    qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_c_qaz =  a*qaz.^(b) - t1_H(x,:) - t2_L.*qaz;
    W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
    W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
    LHS = fbx_a.*(W_a_qaz-W_c_qaz);
    RHS = fbx_b.*(W_a_qbz-W_c_qbz);
    click1 = (abs(LHS - RHS));

    L2 = alpha_c + alpha_d;
    lambda = L2;
    qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_c_qaz =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
    W_a_qbz =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
```

```
    W_c_qbz =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
    LHS = fbx_a.*(W_a_qaz-W_c_qaz);
    RHS = fbx_b.*(W_a_qbz-W_c_qbz);
    click2 = (abs(LHS - RHS));

    click =
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
    while click > 0.000000000001;


        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_c_qaz =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
        W_a_qbz =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
        W_c_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_qaz =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        W_a_qbz =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
        W_c_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_qaz =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        W_a_qbz =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
        W_c_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
        LHS = fbx_a.*(W_a_qaz-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
```

```
            qaz = qaz3;
            qbz = qbz3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
            qaz = qaz4;
            qbz = qbz4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
            qaz = qaz5;
            qbz = qbz5;
        end
    end
    lambdas(x,1) = lambda;
    qazs(x,1) = qaz;
    qbzs(x,1) = qbz;
end

W_a_qaz =  a*qazs.^(b) - t1_L - t2_H.*qazs;
W_b_qbz =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;
W_c_qaz =  a*qazs.^(b) - t1_H - t2_L.*qazs;

% condition4 = 1 if W_b_qbz -((alpha_a+lambdas)./alpha_b).*delta1 - (alpha_c +
alpha_d-lambdas).*delta2.*qbzs./alpha_b < W_c_fbq_c - (alpha_d/alpha_c).*delta1
condition4 = sign(sign(W_c_fbq_c - (alpha_d/alpha_c).*delta1 - (W_b_qbz -
(alpha_a./alpha_b).*delta1 - (alpha_c + alpha_d).*delta2.*qbzs./alpha_b))+1);

% condition5 = 1 if W_a_qaz + (lambdas./alpha_a).*(W_a_qaz - W_c_qaz) < W_c_fbq_c
- (alpha_d/alpha_c).*delta1
condition5 = sign(sign(W_c_fbq_c - (alpha_d/alpha_c).*delta1 - (W_a_qaz +
(lambdas./alpha_a).*(W_a_qaz - W_c_qaz)))+1);

%
**************************************************************************
****************************************
% Part 1 Senario 2 Solution 1.2.a

key = condition1.*condition2.*condition3.*condition5;
key12a = key + key12a;

VWa12a = W_a_qaz + (lambdas./alpha_a).*(W_a_qaz - W_c_qaz);
VWb12a = W_b_qbz - delta1.*(alpha_a + lambdas)./alpha_b - qbzs.*delta2.*(alpha_c +
alpha_d - lambdas)./alpha_b ;
VWc12a = fbW_c - delta1.*alpha_d./alpha_c ;

x = (alpha_a.*fbx_a).*(VWa12a) + (alpha_b.*fbx_b).*(VWb12a) +
(alpha_c.*fbx_c).*(VWc12a);
rev_1_2_a = (x + (alpha_d.*fbx_d).*fbW_d);
OptimalRev = rev_1_2_a.*sign(key12a);
```

```
disp('rev_I_2_a is done')

%**********************************************************************************
***********************************************************************************
% Part I Senario 2 Case: VWa > VWc > VWb
% in this section I work out which solution applies and then I will go
% through and apply them after doing the allocation for the case VWb>VWc
%______________________________________________________________________
% Allocation to solution
key = condition1.*condition2.*condition3.*condition4.*(1-condition5);
lambdas2_star = lambdas;

tag = find(key);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
% This loop solves for lambda2_1 s.t. VWa(lambda2_1)=VWc
for i = 1:tags

    x = tag(i);

    L1 = -0.002;
    lambda = L1;
    qaz1 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    W_a_z =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
    W_c_z =  a*qaz1.^(b) - t1_H(x,:) - t2_L.*qaz1;
    LHS1 = W_a_z + (lambda./alpha_a).*(W_a_z - W_c_z);
    RHS = fbW_c(x,:) -((alpha_d )./alpha_c).*delta1(x,:) ;
    click1 = (abs(LHS1 - RHS));

    L2 = lambdas2_star(x,:);
    lambda = L2;
    qaz2 =  ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    W_a_z =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_c_z =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
    LHS2 = W_a_z + (lambda./alpha_a).*(W_a_z - W_c_z);
    click2 = (abs(LHS2 - RHS));

    stop = 0;
    count = 1;
    click =
10000000000000000000000000000000000000000000000000000000000000000000000000000;
    while click > 0.00000000001;

        if count > 200
            stop = 1;
            L1 = -1000000000000;
            L2 = L1;
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
```

```
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_z =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_c_z =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
        LHS = W_a_z + (lambda./alpha_a).*(W_a_z - W_c_z);
        click3 = (1-stop).*(abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_z =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_z =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        LHS = W_a_z + (lambda./alpha_a).*(W_a_z - W_c_z);
        click4 = (1-stop).*(abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_z =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_z =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        LHS = W_a_z + (lambda./alpha_a).*(W_a_z - W_c_z);
        click5 = (1-stop).*(abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas(x,1) = lambda;
end
lambdas2_1 = lambdas;

tag = find(key);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
% This loop solves for lambda2_3 s.t xa_max.*(W_a_qaz-W_c_qaz)=fbx_b.*(W_a_qbz-
W_c_qbz)
for i = 1:tags
    xa_max = (((1-(alpha_b + alpha_c)^N)./N) - (alpha_d*fbx_d))./alpha_a;
```

```
      xa_bar = (1 - N*alpha_d*fbx_d - N*alpha_b*fbx_b)./(N*alpha_a + N*alpha_c); %
 new code
      xa_max = xa_bar; % new code
 ++++++++++++++++++++++++++++++++++++++++++++++++++++
     x = tag(i);

     L1 = -0.5;
     lambda = L1;
     qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
     W_c_qaz =  a*qaz.^(b) - t1_H(x,:) - t2_L.*qaz;
     W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
     W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
     LHS1 = xa_max.*(W_a_qaz-W_c_qaz);
     RHS1 = fbx_b.*(W_a_qbz-W_c_qbz);
     click1 = (abs(LHS1 - RHS1));

     L2 = lambdas2_star(x,:);
     lambda = L2;
     qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
     W_c_qaz =  a*qaz.^(b) - t1_H(x,:) - t2_L.*qaz;
     W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
     W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
     LHS2 = xa_max.*(W_a_qaz-W_c_qaz);
     RHS2 = fbx_b.*(W_a_qbz-W_c_qbz);
     click2 = (abs(LHS2 - RHS2));

     count = 1;
     click =
 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
     while click > 0.000000000000001;

         if count > 100
             disp(['click is:   ' num2str(click)])
             disp(['At line 2255'])
             keyboard
         end
         count = count +1;

         L3 = (2/3)*L1 + L2/3;
         lambda = L3;
         qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
         W_a_qaz3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
         W_c_qaz =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
         W_a_qbz =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
         W_c_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
         LHS = xa_max.*(W_a_qaz3-W_c_qaz);
```

```
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_qaz =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        W_a_qbz =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
        W_c_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
        LHS = xa_max.*(W_a_qaz4-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_qaz =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        W_a_qbz =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
        W_c_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
        LHS = xa_max.*(W_a_qaz5-W_c_qaz);
        RHS = fbx_b.*(W_a_qbz-W_c_qbz);
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas(x,1) = lambda;
end

% Now see which lambda is the maximum one
lambdas2_2 =lambdas;
compare = [lambdas2_1 lambdas2_2 zeros(simsize,1)];
compare_max = max(compare,[],2)*[1 1 1];
keys = [compare == compare_max].*[key key key];
```

```
key12c = keys(:,2) + key12c;

xa_max = (((1-(alpha_b + alpha_c)^N)./N) - (alpha_d*fbx_d))./alpha_a;
xc_min = (((1-(alpha_b)^N)./N) - (alpha_d*fbx_d) - xa_max.*alpha_a)./alpha_c;
condition6 = sign(sign(xc_min.*delta1 - xa_max.*fbq_a.*delta2 - (fbx_b.*(W_a_q_b2
- W_c_q_b2)))+1);
key11b = keys(:,3).*condition6 + key11b;
key11c = keys(:,3).*(1-condition6) + key11c;

key2_1 = keys(:,1);
qazs = ((t2_H + lambdas2_1.*delta2./alpha_a)./(a.*b)).^(1/(b-1));
qbzs = ((t2_H + (alpha_c + alpha_d -
lambdas2_1).*delta2./alpha_b)./(a.*b)).^(1/(b-1));
W_a_qazs =  a*qazs.^(b) - t1_L - t2_H.*qazs;
W_c_qazs =  a*qazs.^(b) - t1_H - t2_L.*qazs;
W_a_qbzs =  a*qbzs.^(b) - t1_L - t2_H.*qbzs;
W_c_qbzs =  a*qbzs.^(b) - t1_H - t2_L.*qbzs;
x = (W_a_qazs-W_c_qazs);
z = find(x == 0);
x(z,:) = 0.00000000001;
x_a = fbx_b.*(W_a_qbzs-W_c_qbzs)./x;
x_c = (1-N.*(alpha_d.*fbx_d + alpha_b.*fbx_b + alpha_a.*x_a))./(N.*alpha_c);
signx = sign(sign(x_c - x_a)+1); %ie =1 if x_c > x_a
key2_1b = key2_1.*signx;
key2_1c = key2_1.*(1-signx);

key12b = sign(key2_1b + key12b);
key12c = sign(key2_1c + key12c);
% the sign function in the above 2 lines are there since often this assignmnet
% operates at a boundary and we end up picking up the same point twice (see
% note at start of code)
% That ends the allocation
% now the logical thing to do is to work out the allocation for the case
% VWa>VWb>VWc
%_____________________________________________________________________________

% Part I Senario 2 Case: VWa > VWb > VWc
% in this section I work out which solution applies and then I will go
% through and apply them after doing the allocation for the case VWb>VWc

key = condition1.*condition2.*condition3.*(1-condition4);
% what I am doing is looking at the sign of x_a - x_c when lambda2 = 0
x = (W_a_fbq_b-W_c_fbq_b);
z = find(x == 0);
x(z,:) = 0.00000000001;
x_a = (fbx_b.*(W_a_q_b2-W_c_q_b2)./x).*key;
x_c = (1-N.*(alpha_d.*fbx_d + alpha_b.*fbx_b + alpha_a.*x_a))./(N.*alpha_c);
key12c11c11d11e = (x_a >= x_c).*key + key12c11c11d11e;
key11c11d11e = (x_a < x_c).*key + key11c11d11e;
%*****************************************************************************
*****************************************************************************
% Part I Senario 2 Solution 1.2.b
key12b = key12b;
```

```
 % from the above we have lambda2_1 which gives the lambdas for this
 % most of this solution - however there are a few instances where we need
 % to compute some other points in the parameter spapce so we run through
 % the computation again
 tag = find(key12b);
 lambdas = zeros(simsize,1);
 qazs = lambdas;
 qbzs = lambdas;
 W_a_zs= lambdas;
 tags = size(tag,1);
 for i = 1:tags

     x = tag(i);

     L1 = 0;
     lambda = L1;
     qaz1 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     W_a_z1 =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
     LHS1 = W_a_z1 - (lambda./alpha_a).*( delta2(x,:).*qaz1) +
 (lambda./alpha_a).*delta1(x,:);
     RHS = fbW_c(x,:) -((alpha_d )./alpha_c).*delta1(x,:) ;
     click1 = (abs(LHS1 - RHS));

     L2 = alpha_d + alpha_c ;
     lambda = L2;
     qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     W_a_z2 =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
     LHS2 = W_a_z2 - (lambda./alpha_a).*( delta2(x,:).*qaz2) +
 (lambda./alpha_a).*delta1(x,:);
     click2 = (abs(LHS2 - RHS));
     stop = 0;
     count = 1;
     click =
 1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
     while click > 0.00000000001;

         if count > 200
             disp('problem at 1215')
             stop = 1;
         end
         count = count +1;

         L3 = (2/3)*L1 + L2/3;
         lambda = L3;
         qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         W_a_z =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
         LHS = W_a_z - (lambda./alpha_a).*( delta2(x,:).*qaz3) +
 (lambda./alpha_a).*delta1(x,:);
         click3 = (1-stop).*(abs(LHS - RHS));

         L4 = (1/2)*L1 + L2/2;
         lambda = L4;
         qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         W_a_z =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
```

```
        LHS = W_a_z - (lambda./alpha_a).*( delta2(x,:).*qaz4) +
 (lambda./alpha_a).*delta1(x,:);
        click4 = (1-stop).*(abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        W_a_z =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        LHS = W_a_z - (lambda./alpha_a).*( delta2(x,:).*qaz5) +
 (lambda./alpha_a).*delta1(x,:);
        click5 = (1-stop).*(abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
    end
    lambdas(x,1) = lambda;

 end

 qazs = ((t2_H + lambdas.*delta2./alpha_a)./(a.*b)).^(1/(b-1));
 qbzs = ((t2_H + (alpha_c + alpha_d - lambdas).*delta2./alpha_b)./(a.*b)).^(1/(b-
 1));
 W_a_qazs =  a*qazs.^(b) - t1_L - t2_H.*qazs;
 W_c_qazs =  a*qazs.^(b) - t1_H - t2_L.*qazs;
 W_a_qbzs =  a*qbzs.^(b) - t1_L - t2_H.*qbzs;
 W_b_qbzs =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;
 W_c_qbzs =  a*qbzs.^(b) - t1_H - t2_L.*qbzs;
 x = (W_a_qazs-W_c_qazs);
 z = find(x == 0);
 x(z,:) = 0.00000000001;
 x_a = fbx_b.*(W_a_qbzs-W_c_qbzs)./x;
 x_c = (1-N.*(alpha_d.*fbx_d + alpha_b.*fbx_b + alpha_a.*x_a))./(N.*alpha_c);

 VWa12b = W_a_qazs - (lambdas./alpha_a).*(delta2.*qazs) +
 (lambdas./alpha_a).*delta1;
 VWb12b = W_b_qbzs - delta1.*(alpha_a + lambdas)./alpha_b - qbzs.*delta2.*(alpha_c
 + alpha_d - lambdas)./alpha_b;
 VWc12b = fbW_c -((alpha_d )./alpha_c).*delta1 ;

 x = (alpha_a.*x_a).*(VWa12b) + (alpha_b.*fbx_b).*(VWb12b) +
```

```
 (alpha_c.*x_c).*(VWc12b);
 rev_1_2_b = (x + (alpha_d.*fbx_d).*fbW_d);
 OptimalRev = [OptimalRev rev_1_2_b.*sign(key12b)];
 disp('rev_1_2_b is done')

 %*************************************************************************
 *************************************************************************
 % Part I Senario 2 Solution 1.2.c

 key = key12c21de + key12c + key12c11c11d11e;

 x_bar = (1 - N*alpha_d*fbx_d - N*alpha_b*fbx_b)./(N*alpha_a + N*alpha_c);
 tag = find(key);
 lambdas = zeros(simsize,1);
 qazs = lambdas;
 qbzs = lambdas;
 W_a_zs= lambdas;
 tags = size(tag,1);
 for i = 1:tags

     x = tag(i);

     L1 = 0;
     lambda = L1;
     qaz1 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz1 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz1 =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
     W_c_qaz1 =  a*qaz1.^(b) - t1_H(x,:) - t2_L.*qaz1;
     W_a_qbz1 =  a*qbz1.^(b) - t1_L - t2_H(x,:).*qbz1;
     W_c_qbz1 =  a*qbz1.^(b) - t1_H(x,:) - t2_L.*qbz1;
     LHS1 = x_bar.*(W_a_qaz1-W_c_qaz1);
     RHS1 = fbx_b.*(W_a_qbz1-W_c_qbz1);
     click1 = (abs(LHS1 - RHS1));

     L2 = alpha_c + alpha_d;
     lambda = L2;
     qaz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
     qbz2 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
     W_a_qaz2 =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
     W_c_qaz2 =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
     W_a_qbz2 =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
     W_c_qbz2 =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
     LHS2 = x_bar.*(W_a_qaz2-W_c_qaz2);
     RHS2 = fbx_b.*(W_a_qbz2-W_c_qbz2);
     click2 = (abs(LHS2 - RHS2));

     count = 1;
     click =
 10000000000000000000000000000000000000000000000000000000000000000000000000000000;
     if sign(LHS1 - RHS1) == sign(LHS2 - RHS2)
         click = 0;
         lambda = L2;
```

```
    end

    while click > 0.0000000000001;

        if count > 200
            disp(['click is:   ' num2str(click)])
            disp(['At line 1079'])
            keyboard
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz3 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_c_qaz3 =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
        W_a_qbz3 =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
        W_c_qbz3 =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
        LHS = x_bar.*(W_a_qaz3-W_c_qaz3);
        RHS = fbx_b.*(W_a_qbz3-W_c_qbz3);
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_c_qaz4 =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
        W_a_qbz4 =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
        W_c_qbz4 =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
        LHS = x_bar.*(W_a_qaz4-W_c_qaz4);
        RHS = fbx_b.*(W_a_qbz4-W_c_qbz4);
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_c_qaz5 =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
        W_a_qbz5 =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
        W_c_qbz5 =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
        LHS = x_bar.*(W_a_qaz5-W_c_qaz5);
        RHS = fbx_b.*(W_a_qbz5-W_c_qbz5);
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
```

```
            click = click3;
            qaz = qaz3;
            qbz = qbz3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
            qaz = qaz4;
            qbz = qbz4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
            qaz = qaz5;
            qbz = qbz5;
        end
    end
    lambdas(x,1) = lambda;
    qazs(x,1) = qaz;
    qbzs(x,1) = qbz;
end

qbzs = ((t2_H + (lambdas).*delta2./alpha_b)./(a.*b)).^(1/(b-1));
W_a_qaz =  a*qazs.^(b) - t1_L - t2_H.*qazs;
W_b_qbz =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;

% lambdas3 = lambdas;
% x = (fbW_c - W_a_qaz) + ((alpha_c + alpha_d - lambdas3)./alpha_a).*delta2.*qazs;
% lambdas5 = (x - (alpha_c -
lambdas3).*delta1./alpha_a)./(delta1.*((1/alpha_a)+(1/alpha_c)));
% lambda2 = (alpha_c + lambdas5 - lambdas3).*key;
% key_lambda2 = sign(sign(lambda2)+1); % =1 if lambda2 > 0
% key11c = key11c + (1-key_lambda2).*key;
%
% VWc12c = fbW_c - delta1.*lambdas5./alpha_c;
% VWa12c = VWc12c;
% VWb12c = W_b_qbz - qbzs.*delta2.*(lambdas3)./alpha_b - delta1.*(-lambdas3
+alpha_c + alpha_a + alpha_d)./alpha_b;
% key_VWc_VWb = sign(sign(VWc12c-VWb12c)+1); % =1 if VWc > VWb
% key21de = key21de + (1-key_VWc_VWb).*key;
% key12c = key.*key_VWc_VWb.*key_lambda2;

lambdas3 = lambdas;
x = (fbW_c - W_a_qaz) + ((alpha_c + alpha_d - lambdas3)./alpha_a).*delta2.*qazs;
lambdas5 = (x - (alpha_c -
lambdas3).*delta1./alpha_a)./(delta1.*((1./alpha_a)+(1./alpha_c)));
lambda2 = (alpha_c + lambdas5 - lambdas3).*key;
key_lambda2 = sign(sign(lambda2)+1); % =1 if lambda2 > 0

VWc12c = fbW_c - delta1.*lambdas5./alpha_c;
VWa12c = VWc12c;
VWb12c = W_b_qbz - qbzs.*delta2.*(lambdas3)./alpha_b - delta1.*(-lambdas3 +alpha_c
```

```
 + alpha_a + alpha_d)./alpha_b;
 key_VWb_VWc = sign(sign(VWb12c-VWc12c)+1).*key12c21de; % =1 if VWb > VWc
 key21de = key21de + key_VWb_VWc;
 key11c = key11c + (1-key_lambda2).*key.*(1-key_VWb_VWc).*(1-key12c11c11d11e);
 key11c11d11e = key11c11d11e + (1-key_lambda2).*key.*(1-
 key_VWb_VWc).*key12c11c11d11e;
 % line above added due to proof not being quite right
 key12c = key.*key_lambda2.*(1-key_VWb_VWc);



 x = (alpha_a.*x_bar).*(VWa12c) + (alpha_b.*fbx_b).*(VWb12c) +
 (alpha_c.*x_bar).*(VWc12c);
 rev_1_2_c = (x + (alpha_d.*fbx_d).*fbW_d);
 OptimalRev = [OptimalRev rev_1_2_c.*sign(key12c)];
 disp('rev_1_2_c is done')

 %*************************************************************************
 *************************************************************************
 %*************************************************************************
 *************************************************************************
 %*************************************************************************
 *************************************************************************
 %*************************************************************************
 *************************************************************************
 %*************************************************************************
 *************************************************************************
 %*************************************************************************
 *************************************************************************
 % Now do part 1 senario 1

 % Prelim for Part I

 % Define the class that they fall into

 % condition1 = 1 if fbW_c > fbW_a
 condition1 = sign(sign(fbW_c - fbW_a)+1);

 % condition2 = 1 if W_a_fbq_b - W_c_fbq_b > 0
 condition2 = sign(sign(W_a_fbq_b - W_c_fbq_b)+1);

 % condition3 = 1 if fbx_b*(W_a(qb^2)-W_c(qb^2)) < fbx_a*(W_a(fbq_a)-W_c(fbq_a))
 condition3 = sign(sign(fbx_a.*(W_a_fbq_b - W_c_fbq_b) - fbx_b.*(W_a_q_b2 -
 W_c_q_b2))+1);

 % condition4 = 1 if W_a_fbq_a < W_c_fbq_c - (alpha_d/alpha_c).*delta1
 condition4 = sign(sign(W_c_fbq_c - (alpha_d/alpha_c).*delta1 - fbW_a)+1);

 % condition5 = 1 if W_b_q_b2 - alpha_a/alpha_b.*delta1-
 ((alpha_c+alpha_d)./alpha_b).*q_b2.*delta2 < W_c_fbq_c - (alpha_d/alpha_c).*delta1
 condition5 = sign(sign(W_c_fbq_c - (alpha_d/alpha_c).*delta1- (W_b_q_b2 -
 alpha_a/alpha_b.*delta1-((alpha_c+alpha_d)./alpha_b).*q_b2.*delta2))+1);

 %*************************************************************************
```

```
**************************************************************************
% Start with Part I senario 1 Solution 11a

key = condition1.*condition2.*condition3.*condition4;
key11a = key11a + key;

% now work out the revenue
xa = (alpha_a.*fbx_a).*(fbW_a);
xb =  (alpha_b.*fbx_b).*(W_b_q_b2 - delta1.*(alpha_a)./alpha_b -
q_b2.*delta2.*(alpha_c + alpha_d)./alpha_b );
xc =  (alpha_c.*fbx_c).*(fbW_c - delta1.*alpha_d./alpha_c );
x = xa + xb + xc;
rev_1_1_a = (x + (alpha_d.*fbx_d).*fbW_d);
OptimalRev = [OptimalRev rev_1_1_a.*sign(key11a)];
disp('rev_1_1_a is done')

%*************************************************************************
*************************************************************************
% Start with Part I senario 1 Solutions 1.1.b and 1.1.c

% Preliminaries: we do the allocation into the solution first
xa_max = (((1-(alpha_b + alpha_c)^N)./N) - (alpha_d*fbx_d))./alpha_a;
xc_min = (((1-(alpha_b)^N)./N) - (alpha_d*fbx_d) - xa_max.*alpha_a)./alpha_c;
condition6 = sign(sign(xc_min.*delta1 - xa_max.*fbq_a.*delta2 - (fbx_b.*(W_a_q_b2
- W_c_q_b2)))+1);

% allocation for 1.1.b:
keya = condition1.*condition2.*condition3.*(1-condition4).*condition5.*condition6;
key11b = key11b + keya;

% allocation for 1.1.c:
keyb = condition1.*condition2.*condition3.*(1-condition4).*condition5.*(1-
condition6);
key11c = key11c + keyb;
%
_____________________________________________________________________________
_
% Now do the solution for 1.1.b

VWb11b = W_b_q_b2 - delta1.*(alpha_a)./alpha_b - q_b2.*delta2.*(alpha_c +
alpha_d)./alpha_b ;
VWc11b = fbW_c - delta1.*alpha_d./alpha_c ;
x = (alpha_a.*xa_max).*(fbW_a) + (alpha_b.*fbx_b).*(VWb11b) +
(alpha_c.*xc_min).*(VWc11b);
rev_1_1_b = (x + (alpha_d.*fbx_d).*fbW_d);

% OptimalRev = [OptimalRev rev_1_1_b.*sign(key11b + key11c )];
OptimalRev = [OptimalRev rev_1_1_b.*sign(key11b )]; %R++++++++++++++++++
disp('rev_1_1_b is done')

% solution 1.1.c. is after the next case

%*************************************************************************
*************************************************************************
```

```
% Start with Part I senario 1 Case VWa > VWb > VWc

key = condition1.*condition2.*condition3.*(1-condition4).*(1-condition5);
key11c11d11e = key + key11c11d11e;

% Explaination of algroithm:
% what i need to do first is to work out which solution applies. this is done in a
basically two step process.
% Step 1: compare lambdas -
%     a. compute lambda4_1 which sets VWb=VWc
%     b. compute lambda4_2 which sets VWb=VWa
%     c. work out which one is lower.
% Step 2: work out implications for x's
%     a. if lambda4_2 is lower then done sln 1.1.e
%     b. if not, need to look and see if x_b=fbx_b if so then in sln 1.1.c, else
sln 1.1.d
% The last thing to do is to do a bit of work to get sln 1.1.e
tag = find(key11c11d11e);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);

    L1 = 0;
    lambda = L1;
    qbz1 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_b_qbz1 =  a*qbz1.^(b) - t1_H(x,:) - t2_H(x,:).*qbz1;
    LHS1 = W_b_qbz1 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz1.*delta2(x,:)./alpha_b;
    RHS1 = fbW_c(x,:) - delta1(x,:).*(alpha_d -lambda)./alpha_c;
    click1 = (abs(LHS1 - RHS1));

    L2 = alpha_d;
    lambda = L2;
    qbz2 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_b_qbz2 =  a*qbz2.^(b) - t1_H(x,:) - t2_H(x,:).*qbz2;
    LHS2 = W_b_qbz2 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz2.*delta2(x,:)./alpha_b;
    RHS2 = fbW_c(x,:) - delta1(x,:).*(alpha_d -lambda)./alpha_c;
    click2 = (abs(LHS2 - RHS2));

    count = 1;
    click =
1000000000000000000000000000000000000000000000000000000000000000000000000000000;
    if sign(LHS1 - RHS1) == sign(LHS2 - RHS2)
        click = 0;
        lambda = L2;
    end
```

```
    while click > 0.0000000000001;

        if count > 100
            disp(['click is:   ' num2str(click)])
            disp(['At line 1073'])
            keyboard
        end
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qbz3 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_b_qbz3 =  a*qbz3.^(b) - t1_H(x,:) - t2_H(x,:).*qbz3;
        LHS = W_b_qbz3 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz3.*delta2(x,:)./alpha_b;
        RHS = fbW_c(x,:) - delta1(x,:).*(alpha_d -lambda)./alpha_c;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qbz4 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_b_qbz4 =  a*qbz4.^(b) - t1_H(x,:) - t2_H(x,:).*qbz4;
        LHS = W_b_qbz4 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz4.*delta2(x,:)./alpha_b;
        RHS = fbW_c(x,:) - delta1(x,:).*(alpha_d -lambda)./alpha_c;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qbz5 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_b_qbz5 =  a*qbz5.^(b) - t1_H(x,:) - t2_H(x,:).*qbz5;
        LHS = W_b_qbz5 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz5.*delta2(x,:)./alpha_b;
        RHS = fbW_c(x,:) - delta1(x,:).*(alpha_d -lambda)./alpha_c;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
```

```
        end
    end
    lambdas(x,1) = lambda;
end
lambdas4_1 = lambdas;

tag = find(key11c11d11e);
lambdas = zeros(simsize,1);
qazs = lambdas;
qbzs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);

    L1 = 0;
    lambda = L1;
    qaz1 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz1 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz1 =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
    W_b_qbz1 =  a*qbz1.^(b) - t1_H(x,:) - t2_H(x,:).*qbz1;
    LHS1 = W_b_qbz1 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz1.*delta2(x,:)./alpha_b;
    RHS1 = W_a_qaz1 - qaz1.*delta2(x,:).*lambda./alpha_a;
    click1 = (abs(LHS1 - RHS1));

    L2 = alpha_d;
    lambda = L2;
    qaz2 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz2 = ((t2_H(x,:) + (alpha_d + alpha_c -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_qaz2 =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_b_qbz2 =  a*qbz2.^(b) - t1_H(x,:) - t2_H(x,:).*qbz2;
    LHS2 = W_b_qbz2 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
alpha_d - lambda).*qbz2.*delta2(x,:)./alpha_b;
    RHS2 = W_a_qaz2 - qaz2.*delta2(x,:).*lambda./alpha_a;
    click2 = (abs(LHS2 - RHS2));

    count = 1;
    click =
10000000000000000000000000000000000000000000000000000000000000000000000000000;
    if sign(LHS1 - RHS1) == sign(LHS2 - RHS2)
        click = 0;
        lambda = L2;
    end
    while click > 0.0000000000001;

        if count > 100
            disp(['click is:   ' num2str(click)])
            disp(['At line 1073'])
            keyboard
        end
```

```
        count = count +1;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz3 = ((t2_H(x,:) + (alpha_d + alpha_c -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_b_qbz3 =  a*qbz3.^(b) - t1_H(x,:) - t2_H(x,:).*qbz3;
        LHS = W_b_qbz3 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
 alpha_d - lambda).*qbz3.*delta2(x,:)./alpha_b;
        RHS = W_a_qaz3 - qaz3.*delta2(x,:).*lambda./alpha_a;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz4 = ((t2_H(x,:) + (alpha_d + alpha_c -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_b_qbz4 =  a*qbz4.^(b) - t1_H(x,:) - t2_H(x,:).*qbz4;
        LHS = W_b_qbz4 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
 alpha_d - lambda).*qbz4.*delta2(x,:)./alpha_b;
        RHS = W_a_qaz4 - qaz4.*delta2(x,:).*lambda./alpha_a;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz5 = ((t2_H(x,:) + (alpha_d + alpha_c -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_qaz5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_b_qbz5 =  a*qbz5.^(b) - t1_H(x,:) - t2_H(x,:).*qbz5;
        LHS = W_b_qbz5 - (alpha_a + lambda).*delta1(x,:)./alpha_b - (alpha_c +
 alpha_d - lambda).*qbz5.*delta2(x,:)./alpha_b;
        RHS = W_a_qaz5 - qaz5.*delta2(x,:).*lambda./alpha_a;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
        end
```

```
     end
     lambdas(x,1) = lambda;
 end

 lambdas4_2 = lambdas;
 compare = [lambdas4_1 lambdas4_2];
 compare_min = min(compare,[],2)*[1 1];
 keys = [compare == compare_min].*[key11c11d11e key11c11d11e];

 key11e = keys(:,2) + key11e;
 key = keys(:,1);
 qazs = ((t2_H + (lambdas4_1).*delta2./alpha_a)./(a.*b)).^(1/(b-1));
 qbzs = ((t2_H + (alpha_d + alpha_c -
 lambdas4_1).*delta2./alpha_b)./(a.*b)).^(1/(b-1));
 xa_max = (((1-(alpha_b + alpha_c)^N)./N) - (alpha_d.*fbx_d))./alpha_a;
 xc_min = (((1-(alpha_b)^N)./N) - (alpha_d.*fbx_d) - xa_max.*alpha_a)./alpha_c;
 W_a_qbzs =  a*qbzs.^(b) - t1_L - t2_H.*qbzs;
 W_c_qbzs =  a*qbzs.^(b) - t1_H - t2_L.*qbzs;
 LHS = fbx_b.*(W_a_qbzs - W_c_qbzs);
 RHS = xc_min.*delta1 - xa_max.*qazs.*delta2;

 key11c = key11c + key.*sign(sign(LHS - RHS)+1); %=1 if LHS > RHS
 key11d = key11d + key.*(1-sign(sign(LHS - RHS)+1)); %=1 if LHS < RHS
 %_____________________________________________________________________
 % this gives the allocation but still need to work out what x_b and x_c
 % actually are: This doesn't matter since VWb=VWc
 % Now do solution for 1.1.d
 % have to use the feasibility constraint and the constaint on IC_c,a
 W_b_qbzs =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;
 W_a_qazs =  a*qazs.^(b) - t1_L - t2_H.*qazs;
 VWb11d = W_b_qbzs - (alpha_a + lambdas4_1).*delta1./alpha_b - (alpha_c + alpha_d -
 lambdas4_1).*qbzs.*delta2./alpha_b;
 VWa11d = W_a_qazs - qazs.*delta2.*lambdas4_1./alpha_a;
 VWc11d = fbW_c - delta1.*(alpha_d -lambdas4_1)./alpha_c;

 x_bar = (1 - N.*alpha_d.*fbx_d - N.*alpha_a.*xa_max)./(N.*alpha_c + N.*alpha_b);

 x = (alpha_a.*xa_max).*(VWa11d) + ((alpha_c+alpha_b).*x_bar).*(VWb11d);
 rev_1_1_d = (x + (alpha_d.*fbx_d).*fbW_d);
 OptimalRev = [OptimalRev rev_1_1_d.*sign(key11d)];
 disp('rev_1_1_d is done')

 %
 ____________________________________________________________________________
 _
 % Now do the solution for 1.1.c

 key11c = key11c;

 xaMAX = xa_max;
 tag = find(key11c);
 lambdas = zeros(simsize,1);
 qazs = lambdas;
 qbzs = lambdas;
```

```
 xazs = lambdas;
 xczs = lambdas;
 W_a_zs= lambdas;
 tags = size(tag,1);
 for i = 1:tags
     x = tag(i);

     xa_max  = xaMAX ;
     xa_bar = (1 - N*alpha_d*fbx_d - N*alpha_b*fbx_b)./(N*alpha_a + N*alpha_c);
     xa_min = fbx_a;
     xa = (xa_bar + xa_max)/2;
     xc = (((1-(alpha_b)^N)./N) - (alpha_d*fbx_d) - xa.*alpha_a)./alpha_c;

     countx = 1;

     click_x = 1000000000000000000000000000000000000000000000000;
     while click_x > 0.0000001;

         countx = countx + 1;

         if countx > 400
             disp(['click_x is:  ' num2str(click_x)])
             disp('At line 1136')
             keyboard
         end

         stop = 0;
         L1 = 0;
         lambda = L1;
         qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
         W_a_qbz =  a*qbz.^(b) - t1_L - t2_H(x,:).*qbz;
         W_c_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_L.*qbz;
         LHS1 = fbx_b.*(W_a_qbz - W_c_qbz);
         RHS1 = xc.*delta1(x,:) - xa.*qaz.*delta2(x,:);
         click1 = abs(LHS1 - RHS1);

         L2 =  alpha_d;
         lambda = L2;
         qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
         qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
         W_a_qbz =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
         W_c_qbz =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
         LHS2 = fbx_b.*(W_a_qbz - W_c_qbz);
         RHS2 = xc.*delta1(x,:) - xa.*qaz2.*delta2(x,:);
         click2 = abs(LHS2 - RHS2);

         count = 1;

         %need to make allowance for the fact that when not using the
         %correct x's the LHS1<RHS1 and LHS2<RHS2 is possible
         adjust = 0;
```

```
        if (LHS1 < RHS1) & (LHS2 < RHS2)
            L2 = L1;
            adjust = 1;
        end
        if (LHS1 > RHS1) & (LHS2 > RHS2)
            L1 = L2;
            adjust = 1;
        end

        click =
10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
        while click > 0.00000000001;
            if xaMAX - xa_min < 0.00001
                stop = 1;
            end


            if count > 100
                disp(['click is:  ' num2str(click)])
                keyboard
            end

            count = count + 1;


            L3 = (2/3)*L1 + L2/3;
            lambda = L3;
            qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-
1));
            qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
            W_a_qbz =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
            W_c_qbz =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
            LHS = fbx_b.*(W_a_qbz - W_c_qbz);
            RHS = xc.*delta1(x,:) - xa.*qaz3.*delta2(x,:);
            click3 = (1 - adjust).*(abs(LHS - RHS));

            L4 = (1/2)*L1 + L2/2;
            lambda = L4;
            qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-
1));
            qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
            W_a_qbz =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
            W_c_qbz =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
            LHS = fbx_b.*(W_a_qbz - W_c_qbz);
            RHS = xc.*delta1(x,:) - xa.*qaz4.*delta2(x,:);
            click4 = (1 - adjust).*(abs(LHS - RHS));

            L5 = (1/3)*L1 + L2*(2/3);
            lambda = L5;
            qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-
1));
            qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
```

```
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
            W_a_qbz =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
            W_c_qbz =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
            LHS = fbx_b.*(W_a_qbz - W_c_qbz);
            RHS = xc.*delta1(x,:) - xa.*qaz5.*delta2(x,:);
            click5 = (1 - adjust).*(abs(LHS - RHS));

            if click3 == min([click3; click4; click5])
                L1 = L1;
                L2 = L4;
                lambda = L3;
                click = click3;
                qaz = qaz3;
                qbz = qbz3;
            elseif click4 == min([click3; click4; click5])
                L1 = L3;
                L2 = L5;
                lambda = L4;
                click = click4;
                qaz = qaz4;
                qbz = qbz4;
            else
                L1 = L4;
                L2 = L2;
                lambda = L5;
                click = click5;
                qaz = qaz5;
                qbz = qbz5;
            end
        end

        W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        LHSx = W_a_qaz - (lambda./alpha_a).*qaz.*delta2(x,:);
        RHSx = fbW_c(x,:) - ((alpha_d - lambda)./alpha_c).*delta1(x,:);
        click_x = abs(LHSx - RHSx);

        if LHSx - RHSx > 0
            xa_min = xa;
        else
            xa_max = xa;
        end
        xa = (xa_min + xa_max)/2;

        if stop == 1
            click_x = 0;
            xa = xaMAX;
        end
        xc = (((1-(alpha_b)^N)./N) - (alpha_d*fbx_d) - xa.*alpha_a)./alpha_c;
    end
    lambdas(x,1) = lambda;
    qazs(x,1) = qaz;
    qbzs(x,1) = qbz;
    xazs(x,1) = xa;
    xczs(x,1) = xc;
```

```
end

W_a_qaz =  a*qazs.^(b) - t1_L - t2_H.*qazs;
W_b_qbz =  a*qbzs.^(b) - t1_H - t2_H.*qbzs;

% now work out the revenue

VWa11c = W_a_qaz - qazs.*delta2.*(lambdas)./alpha_a;
VWb11c = W_b_qbz - delta1.*(lambdas + alpha_a)./alpha_b - qbzs.*delta2.*(-lambdas
+ alpha_c + alpha_d)./alpha_b;
VWc11c = fbW_c - delta1.*(alpha_d-lambdas)./alpha_c;

x = (alpha_a.*xazs).*(VWa11c) + (alpha_b.*fbx_b).*(VWb11c) +
(alpha_c.*xczs).*(VWc11c);
rev_1_1_c = (x + (alpha_d.*fbx_d).*fbW_d);
OptimalRev = [OptimalRev rev_1_1_c.*sign(key11c)];
disp('rev_1_1_c is done')

%_____________________________________________________________________________
% Now work out solution 1.1.e

key21e = key11e + key21e;

%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
% PART 2 SENARIO 1

% condition1 = 1 if fbW_c > fbW_a
condition1 = sign(sign(fbW_c - fbW_a)+1);

% condition2 = 1 if W_a_fbq_b - W_c_fbq_b < 0
condition2 = sign(sign(W_c_fbq_b - W_a_fbq_b)+1);

% condition3 = 1 if fbx_a*(W_a(qa^2)-W_c(qa^2)) < fbx_b*(W_a(fbq_b)-W_c(fbq_b))
condition3 = sign(sign(fbx_b.*(W_a_fbq_b - W_c_fbq_b) - fbx_a.*(W_a_q_a2 -
W_c_q_a2))+1);

% condition4 = 1 if W_c_fbq_c - delta1*(alpha_d/alpha_c) > W_a_q_a2+(delta1-
q_a2*delta2)*(alpha_c + alpha_d)/alpha_a
condition4 = sign(sign(W_c_fbq_c - delta1.*(alpha_d/alpha_c) - W_a_q_a2 - (delta1
- q_a2.*delta2)*(alpha_c + alpha_d)./alpha_a)+1);

% condition5 = 1 if W_a_q_a2 + (delta1 - q_a2*delta2)*(alpha_c + alpha_d)/alpha_a
> fbW_b - delta1*(alpha_c + alpha_d + alpha_a)/alpha_b);
```

```
condition5 = sign(sign(W_a_q_a2 + (delta1 - q_a2.*delta2).*(alpha_c +
alpha_d)./alpha_a - fbW_b + delta1.*(alpha_c + alpha_d + alpha_a)./alpha_b)+1);


%*************************************************************************
*************************************************************************

% Start with Part II Senario 1 Solution 2.1.a
% condition for this applying is key(element) = 1;
key = condition1.*condition2.*condition3.*condition4.*condition5;
key21a = key + key21a;

% now work out the revenue

x = (alpha_a.*fbx_a).*(W_a_q_a2 + delta1.*(alpha_c + alpha_d)./alpha_a -
q_a2.*delta2*(alpha_c + alpha_d)./alpha_a);
x = x + (alpha_b.*fbx_b).*(fbW_b - delta1.*(alpha_c + alpha_d + alpha_a)./alpha_b
);
x = x + (alpha_c.*fbx_c).*(fbW_c - delta1.*alpha_d./alpha_c );
rev_2_1_a = (x + (alpha_d.*fbx_d).*fbW_d);
% this computes the revenue for solution 2.1.a for every point in the
% parameter set
% this is column 1 in OptimalRev

OptimalRev = [OptimalRev rev_2_1_a.*sign(key21a)];
disp('rev_II_1_a is done')

%*************************************************************************
*************************************************************************

% Part II Senario 1 Solution 2.1.b
% condition for this applying is key(element) = 1;

% need to work out the key to this solution (note this first key will also
% pick up 2.1.d and e)

key = condition1.*condition2.*condition3.*(1-condition4);
key21b = key + key21b;
% this gives the right rankings on the virtual welfares.

% find the new x's
Xbar = (1 - N*alpha_d*fbx_d - N*alpha_b*fbx_b)./(N*alpha_a + N*alpha_c);

% now work out the revenue

tag = find(key21b);
lambdas = zeros(simsize,1);
qazs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);
```

```
    L1 = 0;
    lambda = L1;
    LHS1 = W_a_q_a2(x,:) + ((alpha_c + lambda)./alpha_a).*delta1(x,:) - ((alpha_c
 + alpha_d)./alpha_a).*delta2(x,:).*q_a2(x,:);
    RHS1 = fbW_c(x,:) - lambda./alpha_c.*delta1(x,:) ;
    click1 = (abs(LHS1 - RHS1));

    L2 = alpha_d;
    lambda = L2;
    LHS2 = W_a_q_a2(x,:) + ((alpha_c + lambda)./alpha_a).*delta1(x,:) - ((alpha_c
 + alpha_d)./alpha_a).*delta2(x,:).*q_a2(x,:);
    RHS2 = fbW_c(x,:) - lambda./alpha_c.*delta1(x,:) ;
    click2 = (abs(LHS2 - RHS2));

    click =
 1000000000000000000000000000000000000000000000000000000000000000000000000000;
    while click > 0.00000000001;

        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        LHS = W_a_q_a2(x,:) + ((alpha_c + lambda)./alpha_a).*delta1(x,:) -
 ((alpha_c + alpha_d)./alpha_a).*delta2(x,:).*q_a2(x,:);
        RHS = fbW_c(x,:) - lambda./alpha_c.*delta1(x,:) ;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        LHS = W_a_q_a2(x,:) + ((alpha_c + lambda)./alpha_a).*delta1(x,:) -
 ((alpha_c + alpha_d)./alpha_a).*delta2(x,:).*q_a2(x,:);
        RHS = fbW_c(x,:) - lambda./alpha_c.*delta1(x,:) ;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        LHS = W_a_q_a2(x,:) + ((alpha_c + lambda)./alpha_a).*delta1(x,:) -
 ((alpha_c + alpha_d)./alpha_a).*delta2(x,:).*q_a2(x,:);
        RHS = fbW_c(x,:) - lambda./alpha_c.*delta1(x,:) ;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
            click = click5;
```

```
        end
    end
    lambdas(x,1) = lambda;
end

% check to see if VWc>VWb, if this is true then we can apply solution 2.1.b
VWb = fbW_b - delta1.*(alpha_c + alpha_d + alpha_a)./alpha_b;
VWc = fbW_c - lambdas./alpha_c.*delta1;
key21b = sign(sign(VWc - VWb)+1).*key21b;
% some of the points picked up are really 2.1.d and 2.1.e so we add to the key for
those cases:
key21de = (1-key21b).*key + key21de;

x = (alpha_a.*Xbar).*(VWc);
x = x + (alpha_b.*fbx_b).*(VWb);
x = x + (alpha_c.*Xbar).*(VWc);
rev_2_1_b = (x + (alpha_d.*fbx_d).*fbW_d);
% the above computes the revenue for all points that satisfy the welfare rankings
A>C in the part 2 senario 1 setting

OptimalRev = [OptimalRev rev_2_1_b.*sign(key21b)];
disp('rev_II_1_b is done')

%*************************************************************************
*************************************************************************

% Part II Senario 1 Solution 2.1.c
% condition for this applying is key(element) = 1;

key = condition1.*condition2.*condition3.*condition4.*(1-condition5);
key21c = key + key21c;
% this gives the right rankings on the virtual welfares for pt2 s1
% note that we will have to revisit this solution concept in pt2 s2
% also some of these points are going to devolve into solutions 2.1.d and
% 2.1.e

tag = find(key21c);
lambdas = zeros(simsize,1);
qazs = lambdas;
W_a_zs= lambdas;
tags = size(tag,1);
for i = 1:tags

    x = tag(i);
    L11 = 0;
    lambda = L11;
    qaz = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_z =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_b_z =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
    LHS1 = W_a_z + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz);
    RHS1 = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz;
```

```
    click1 = (abs(LHS1 - RHS1));

    L22 = alpha_c + alpha_d;
    lambda = L22;
    qaz2 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
    qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
    W_a_z =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
    W_b_z =  a*qbz2.^(b) - t1_H(x,:) - t2_H(x,:).*qbz2;
    LHS2 = W_a_z + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz2);
    RHS2 = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz2;
    click2 = (abs(LHS2 - RHS2));

    LAM = L11;
    clicked = click1;
    lam = (L22./200).*[1:200]';
    trueval = LHS1 - RHS1;
    for j = 1:200
        lambda = lam(j);
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1./(b-1));
        qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1./(b-1));
        W_a_z5 =  a.*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_b_z =  a.*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        LHS5 = W_a_z5 + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz5);
        RHS5 = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz;
        clicked = [clicked ; (abs(LHS5 - RHS5))];
        trueval = [trueval;(LHS5 - RHS5)];
        LAM = [LAM;lambda];
    end
    trueval = [trueval;LHS2 - RHS2];
    z1 = find(trueval == min(trueval));
    z2 = find(trueval == max(trueval));
    clicked = [clicked ; click2];
    LAM = [LAM;L22];
    clicked = clicked(min([min(z1) min(z2)]'):max([max(z1) max(z2)]'));
    z = find(clicked == min(clicked));
    L1 = LAM(max([1 min(z)-1]'));
    L2 = LAM(min([max(z)+1 202]'));

    count = 1;
    click =
10000000000000000000000000000000000000000000000000000000000000000000000000000000;
    while click > 0.0000000000001;

        if count > 100
            disp(['click is:   ' num2str(click)])
            disp(['At 4123'])
            keyboard
        end
        count = count +1;
```

```
        L3 = (2/3)*L1 + L2/3;
        lambda = L3;
        qaz3 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_z3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
        W_b_z =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        LHS = W_a_z3 + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz3);
        RHS = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz;
        click3 = (abs(LHS - RHS));

        L4 = (1/2)*L1 + L2/2;
        lambda = L4;
        qaz4 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_z4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
        W_b_z =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        LHS = W_a_z4 + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz4);
        RHS = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz;
        click4 = (abs(LHS - RHS));

        L5 = (1/3)*L1 + L2*(2/3);
        lambda = L5;
        qaz5 = ((t2_H(x,:) + lambda.*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
        qbz = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
        W_a_z5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
        W_b_z =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        LHS = W_a_z5 + (lambda./alpha_a).*(delta1(x,:) - delta2(x,:).*qaz5);
        RHS = W_b_z - ((alpha_a + lambda)./alpha_b).*delta1(x,:) - ((alpha_c +
alpha_d - lambda)./alpha_b).*delta2(x,:).*qbz;
        click5 = (abs(LHS - RHS));

        if click3 == min([click3; click4; click5])
            L1 = L1;
            L2 = L4;
            lambda = L3;
            click = click3;
            qaz = qaz3;
            W_a_z = W_a_z3;
        elseif click4 == min([click3; click4; click5])
            L1 = L3;
            L2 = L5;
            lambda = L4;
            click = click4;
            qaz = qaz4;
            W_a_z = W_a_z4;
        else
            L1 = L4;
            L2 = L2;
            lambda = L5;
```

```
            click = click5;
            qaz = qaz5;
            W_a_z = W_a_z5;
        end
    end
    lambdas(x,1) = lambda;
    qazs(x,1) = qaz;
    W_a_zs(x,1) = W_a_z;
end

% now need to check to see if we are really dealing with a solution 2.1.d
% or 2.1.e situation: this is just a check that VWc>VWb
VWb = W_a_zs + delta1.*(lambdas)./alpha_a - qazs.*delta2.*(lambdas)./alpha_a;
VWc = fbW_c - delta1.*alpha_d./alpha_c;
key21c = sign(sign(VWc - VWb)+1).*key21c;
% some of the points picked up are really 2.1.d and 2.1.e so we add to the key for
those cases:
key21de = (1-key21c).*key + key21de;
% need xb and xa that will work: note do not need them exactly since VWa = VWb
Xbar = (1 - N*alpha_d*fbx_d - N*alpha_c*fbx_c)./(N*alpha_a + N*alpha_b);

% now work out the revenue
x = ((alpha_a + alpha_b).*Xbar).*(VWb);
x = x + (alpha_c.*fbx_c).*(VWc);
rev_2_1_c = (x + (alpha_d.*fbx_d).*fbW_d);
% the above computes the revenue for all points that satisfy the welfare rankings
A>C in the part 2 senario 1 setting

OptimalRev = [OptimalRev rev_2_1_c.*sign(key21c)];
% this is column 3
disp('rev_II_1_c is done')

%*************************************************************************
*************************************************************************
% Part II Senario 1 Solution 2.1.d (and key for Solution 2.1.e)
% condition for this applying is key(element) = 1;

key = condition1.*condition2.*condition3.*(1-condition4).*(1-condition5);
% This gives the virtual welfare ranking for the cases not already tagged in pt2
s1
key21de = key21de + key;


keyTEST  = [key11a key11b key11c key11d key11e key12a key12b key12c key21a key21b
key21c key21d key21e key21de key12c21de key11c11d11e ];


%the logic for the solution in this section runs as follows:
% there are 5 potential constraints to use -
% 1. xa = xc
% 2. N(alpha_d*xd + xa(alpha_a+alpha_c) + xb(alpha_b))= 1
% 3. VWa = VWc
% 4. VWa = VWb
% 5. xa[Wa(qa)-Wc(qa)]=xb[Wa(qb)-Wc(qb)]
```

```
 % what we do is do a line search over the xb's, stopping when VWa = VWb, so
 % step 1: pick xb
 % step 2: use 1 and 2 to get xc and xa
 % step 3: use 5 to set lambda 3
 % step 4: use 3 to set lambda 5
 % compute the VW's, compare.
 % iterate untill the VW's are all equal

 tag = find(key21de);
 XB = zeros(simsize,1);
 VWa = XB;
 VWb = XB;
 lambdas3 = XB;
 lambdas5 = XB;
 tags = size(tag,1);
 for i = 1:tags
     x = tag(i);
     % Step 1: pick xb
     x_b1 = fbx_b;
     % Step 2: impute xa and xc (done in step 3)
     % Step 3: get the lamdba3
     % _____________________________
     % _____________________________ The code for the function
 "Solution21dlambdasearch" is essentially imported from
     % 1.2.c: it is included at the end of this code (line 3000 or so) so that if
 the called function is lost if
     % can be easily recreated.
     out = Solution21dlambdasearch(x_b1);
     % _____________________________
     % Step 4: impute lambda5
     lam31 = lambda;
     W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
     W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
     z = (fbW_c(x,:) - W_a_qaz) + ((alpha_c + alpha_d -
 lam31)./alpha_a).*delta2(x,:).*qaz;
     lam51 = (z - (alpha_c -
 lam31).*delta1(x,:)./alpha_a)./(delta1(x,:).*((1/alpha_a)+(1/alpha_c)));
     % _____________________________
     % Step 5: commpute VW's
     vwa1 = W_a_qaz - delta2(x,:).*qaz.*(alpha_c + alpha_d - lam31)./alpha_a +
 delta1(x,:).*(alpha_c + lam51 - lam31)./alpha_a;
     vwb1 =  W_b_qbz - qbz.*delta2(x,:).*(lam31)./alpha_b - delta1(x,:).*(-lam31
 +alpha_c + alpha_a + alpha_d)./alpha_b;
     vwc1 =  fbW_c(x,:) - delta1(x,:).*lam51./alpha_c ;
     % _____________________________
     click1 = abs(vwa1-vwb1);
     lam21 = alpha_c+lam51-lam31;
     click = 100;
     if sign(lam21)<0
         key21e_adj(x,:) = 1;
         click = 0;
         x_b = 0;
         vwa = 0;
         vwb = -10000;
```

```
        lam3 = 1;
        lam5 = 1;
    end

    x_b2 = (1 - N*alpha_d*fbx_d )./(N*alpha_b + N*alpha_a + N*alpha_c);
    out = Solution21dlambdasearch(x_b2);
    lam32 = lambda;
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
    z = (fbW_c(x,:) - W_a_qaz) + ((alpha_c + alpha_d -
lam32)./alpha_a).*delta2(x,:).*qaz;
    lam52 = (z - (alpha_c -
lam32).*delta1(x,:)./alpha_a)./(delta1(x,:).*((1/alpha_a)+(1/alpha_c)));
    vwa2 = W_a_qaz - delta2(x,:).*qaz.*(alpha_c + alpha_d - lam32)./alpha_a +
delta1(x,:).*(alpha_c + lam52 - lam32)./alpha_a;
    vwb2 =  W_b_qbz - qbz.*delta2(x,:).*(lam32)./alpha_b - delta1(x,:).*(-lam32
+alpha_c + alpha_a + alpha_d)./alpha_b;
    vwc2 =  fbW_c(x,:) - delta1(x,:).*lam52./alpha_c ;
    click2 = abs(vwa2-vwb2);

    if vwb2 > vwa2
        key21e_adj(x,:) = 1;
        click = 0;
        x_b = 0;
        lam3 = 1;
        lam5 = 1;
        vwa = 0;
        vwb = -10000;
    end

    while click > 0.000000000001;

        if count > 300
            disp(['click is:   ' num2str(click)])
            disp(['At 6776'])
            keyboard
        end
        count = count +1;
        x_b3 = (2/3)*x_b1 + x_b2/3;
        x_b4 = (1/2)*x_b1 + x_b2/2;
        x_b5 = (1/3)*x_b1 + x_b2*(2/3);

        out = Solution21dlambdasearch(x_b3);
        lam33 = lambda;
        W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        z = (fbW_c(x,:) - W_a_qaz) + ((alpha_c + alpha_d -
lam33)./alpha_a).*delta2(x,:).*qaz;
        lam53 = (z - (alpha_c -
lam33).*delta1(x,:)./alpha_a)./(delta1(x,:).*((1/alpha_a)+(1/alpha_c)));
        vwa3 = W_a_qaz - delta2(x,:).*qaz.*(alpha_c + alpha_d - lam33)./alpha_a +
delta1(x,:).*(alpha_c + lam53 - lam33)./alpha_a;
        vwb3 =  W_b_qbz - qbz.*delta2(x,:).*(lam33)./alpha_b - delta1(x,:).*(-
lam33 +alpha_c + alpha_a + alpha_d)./alpha_b;
```

```
        click3 = abs(vwa3-vwb3);

        out = Solution21dlambdasearch(x_b4);
        lam34 = lambda;
        W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        z = (fbW_c(x,:) - W_a_qaz) + ((alpha_c + alpha_d -
lam34)./alpha_a).*delta2(x,:).*qaz;
        lam54 = (z - (alpha_c -
lam34).*delta1(x,:)./alpha_a)./(delta1(x,:).*((1/alpha_a)+(1/alpha_c)));
        vwa4 = W_a_qaz - delta2(x,:).*qaz.*(alpha_c + alpha_d - lam34)./alpha_a +
delta1(x,:).*(alpha_c + lam54 - lam34)./alpha_a;
        vwb4 =  W_b_qbz - qbz.*delta2(x,:).*(lam34)./alpha_b - delta1(x,:).*(-
lam34 +alpha_c + alpha_a + alpha_d)./alpha_b;
        click4 = abs(vwa4-vwb4);

        out = Solution21dlambdasearch(x_b5);
        lam35 = lambda;
        W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        z = (fbW_c(x,:) - W_a_qaz) + ((alpha_c + alpha_d -
lam35)./alpha_a).*delta2(x,:).*qaz;
        lam55 = (z - (alpha_c -
lam35).*delta1(x,:)./alpha_a)./(delta1(x,:).*((1/alpha_a)+(1/alpha_c)));
        vwa5 = W_a_qaz - delta2(x,:).*qaz.*(alpha_c + alpha_d - lam35)./alpha_a +
delta1(x,:).*(alpha_c + lam55 - lam35)./alpha_a;
        vwb5 =  W_b_qbz - qbz.*delta2(x,:).*(lam35)./alpha_b - delta1(x,:).*(-
lam35 +alpha_c + alpha_a + alpha_d)./alpha_b;
        click5 = abs(vwa5-vwb5);

        if click3 == min([click3; click4; click5])
            x_b1 = x_b1;
            x_b2 = x_b4;
            x_b = x_b3;
            click = click3;
            vwa = vwa3;
            vwb = vwb3;
            lam3 = lam33;
            lam5 = lam53;
        elseif click4 == min([click3; click4; click5])
            x_b1 = x_b3;
            x_b2 = x_b5;
            x_b = x_b4;
            click = click4;
            vwa = vwa4;
            vwb = vwb4;
            lam3 = lam34;
            lam5 = lam54;
        else
            x_b1 = x_b4;
            x_b2 = x_b2;
            x_b = x_b5;
            click = click5;
            vwa = vwa5;
```

```
            vwb = vwb5;
            lam3 = lam35;
            lam5 = lam55;
        end
    end
    XB(x,1) = x_b;
    VWa(x,1) = vwa;
    VWb(x,1) = vwb;
    lambdas3 = lam3;
    lambdas5 = lam5;
end

%now check for 2.1.d vs 2.1.e
lambdas2 = alpha_c + lambdas5 - lambdas3;
key = sign(sign(lambdas2)+1); %positive lambda2 = 1
key21e = key21de - key21de.*key;
key21d = key21de.*key;

% now lets sort out the revenue of 2.1.d
% need xb and xa etc that will work: note do not need them exactly since VWa = VWb
= VWc
Xbar = (1 - N*alpha_d*fbx_d )./(N*alpha_c + N*alpha_a + N*alpha_b);
% now work out the revenue
x = ((alpha_a + alpha_b + alpha_c).*Xbar).*(VWb);
rev_2_1_d = (x + (alpha_d.*fbx_d).*fbW_d);
OptimalRev = [OptimalRev rev_2_1_d.*sign(key21d)];
% this is column 4
disp('rev_II_1_d is done')

%*****************************************************************************
*****************************************************************************
% Part II Senario 1 Solution 2.1.e
% condition for this applying is key(element) = 1;
% The key for this section is computed above

key21e = key21e;

% The algorithm used here is very similar to that used for 2.1.d except that the
virtual welfares have changed etc

tag = find(key21d);
XB = zeros(simsize,1);
VWa = XB;
VWb = XB;
lambdas3 = XB;
lambdas5 = XB;
tags = size(tag,1);
for i = 1:tags
    x = tag(i);
    % Step 1: pick xb
    x_b1 = fbx_b;
    % Step 3: get the lamdba3
    % _____________________________
    % _____________________________ The code for the function
```

```
 "Solution21dlambdasearch" is essentially imported from
    % 1.2.c: it is included at the end of this code (line 3000 or so) so that if
 the called function is lost if
    % can be easily recreated.
    out = Solution21elambdasearch(x_b1);
    % _____________________________
    % Step 4: impute lambda5
    lam41 = lambda;
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
    lam51 = ( (fbW_c(x,:) - W_a_qaz) + lam41*delta2(x,:)*qaz./alpha_a
 )./(delta1(x,:)/alpha_c);
    % _____________________________
    % Step 5: commpute VW's
    vwa1 = W_a_qaz - delta2(x,:).*qaz.*(lam41)./alpha_a;
    vwb1 =  W_b_qbz - qbz.*delta2(x,:).*(alpha_c + alpha_d - lam41)./alpha_b -
 delta1(x,:).*(-lam51 + alpha_a + alpha_d)./alpha_b;
    vwc1 =  fbW_c(x,:) - delta1(x,:).*lam51./alpha_c ;
    % _____________________________
    click1 = abs(vwa1-vwb1);


    x_b2 = (1 - N*alpha_d*fbx_d )./(N*alpha_b + N*alpha_a + N*alpha_c);
    out = Solution21elambdasearch(x_b2);
    lam42 = lambda;
    W_a_qaz =  a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
    W_b_qbz =  a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
    lam52 = ( (fbW_c(x,:) - W_a_qaz) + lam42*delta2(x,:)*qaz./alpha_a
 )./(delta1(x,:)/alpha_c);
    vwa2 = W_a_qaz - delta2(x,:).*qaz.*(lam42)./alpha_a;
    vwb2 =  W_b_qbz - qbz.*delta2(x,:).*(alpha_c + alpha_d - lam42)./alpha_b -
 delta1(x,:).*(-lam52 + alpha_a + alpha_d)./alpha_b;
    vwc2 =  fbW_c(x,:) - delta1(x,:).*lam52./alpha_c ;
    click2 = abs(vwa2-vwb2);

    click = 100;
    while click > 0.0000000001;

        if count > 300
            disp(['click is:   ' num2str(click)])
            disp(['At 12345'])
            keyboard
        end
        count = count +1;
        x_b3 = (2/3)*x_b1 + x_b2/3;
        x_b4 = (1/2)*x_b1 + x_b2/2;
        x_b5 = (1/3)*x_b1 + x_b2*(2/3);

        out = Solution21elambdasearch(x_b3);
        lam43 = lambda;
        W_a_qaz = a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz = a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        lam53 = ( (fbW_c(x,:) - W_a_qaz) + lam43*delta2(x,:)*qaz./alpha_a
 )./(delta1(x,:)/alpha_c);
```

```
        vwa3 = W_a_qaz - delta2(x,:).*qaz.*(lam43)./alpha_a;
        vwb3 = W_b_qbz - qbz.*delta2(x,:).*(alpha_c + alpha_d - lam43)./alpha_b -
 delta1(x,:).*(-lam53 + alpha_a + alpha_d)./alpha_b;
        vwc3 = fbW_c(x,:) - delta1(x,:).*lam53./alpha_c ;
        click3 = abs(vwa3-vwb3);

        out = Solution21elambdasearch(x_b4);
        lam44 = lambda;
        W_a_qaz = a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz = a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        lam54 = ( (fbW_c(x,:) - W_a_qaz) + lam44*delta2(x,:)*qaz./alpha_a
 )./(delta1(x,:)/alpha_c);
        vwa4 = W_a_qaz - delta2(x,:).*qaz.*(lam44)./alpha_a;
        vwb4 = W_b_qbz - qbz.*delta2(x,:).*(alpha_c + alpha_d - lam44)./alpha_b -
 delta1(x,:).*(-lam54 + alpha_a + alpha_d)./alpha_b;
        vwc4 = fbW_c(x,:) - delta1(x,:).*lam54./alpha_c ;
        click4 = abs(vwa4-vwb4);

        out = Solution21elambdasearch(x_b5);
        lam45 = lambda;
        W_a_qaz = a*qaz.^(b) - t1_L - t2_H(x,:).*qaz;
        W_b_qbz = a*qbz.^(b) - t1_H(x,:) - t2_H(x,:).*qbz;
        lam55 = ( (fbW_c(x,:) - W_a_qaz) + lam45*delta2(x,:)*qaz./alpha_a
 )./(delta1(x,:)/alpha_c);
        vwa5 = W_a_qaz - delta2(x,:).*qaz.*(lam45)./alpha_a;
        vwb5 = W_b_qbz - qbz.*delta2(x,:).*(alpha_c + alpha_d - lam45)./alpha_b -
 delta1(x,:).*(-lam53 + alpha_a + alpha_d)./alpha_b;
        vwc5 = fbW_c(x,:) - delta1(x,:).*lam55./alpha_c ;
        click5 = abs(vwa5-vwb5);

        if click3 == min([click3; click4; click5])
            x_b1 = x_b1;
            x_b2 = x_b4;
            x_b = x_b3;
            click = click3;
            vwa = vwa3;
            vwb = vwb3;
            lam4 = lam43;
            lam5 = lam53;
        elseif click4 == min([click3; click4; click5])
            x_b1 = x_b3;
            x_b2 = x_b5;
            x_b = x_b4;
            click = click4;
            vwa = vwa4;
            vwb = vwb4;
            lam4 = lam44;
            lam5 = lam54;
        else
            x_b1 = x_b4;
            x_b2 = x_b2;
            x_b = x_b5;
            click = click5;
            vwa = vwa5;
```

```
            vwb = vwb5;
            lam4 = lam45;
            lam5 = lam55;
        end
    end

    VWa(x,1) = vwa;
    VWb(x,1) = vwb;
    lambdas3 = lam3;
    lambdas5 = lam5;
end

% now lets sort out the revenue of 2.1.e
% need xb and xa etc that will work: note do not need them exactly since VWa = VWb
= VWc
Xbar = (1 - N*alpha_d*fbx_d )./(N*alpha_c + N*alpha_a + N*alpha_b);
% now work out the revenue
x = ((alpha_a + alpha_b + alpha_c).*Xbar).*(VWb);
rev_2_1_e = (x + (alpha_d.*fbx_d).*fbW_d);
OptimalRev = [OptimalRev rev_2_1_e.*sign(key21e)];
% this is column 5
disp('rev_II_1_e is done')

%*****************************************************************************
*****************************************************************************
%*****************************************************************************
*****************************************************************************
% Now do the graphing thing
test = sum(OptimalRev,2);
test2 = sum(rev_BOEM,2);
x = ceil(simsize./2);
test2(x,:) = test2(x,:)./2;
output = [delta1 test test2];
plot([test test2],'-')
disp('Done')

%*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&
%*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&
% Code for the function: Solution21dlambdasearch

% function [] = Solution21dlambdasearch(input)
% global  lambda x qaz N qbz  t2_H t1_H t1_L t2_L alpha_c alpha_d alpha_b alpha_a
delta2 delta2 a b fbx_d
%
% x_b = input;
%     x_bar = (1 - N*alpha_d*fbx_d - N*alpha_b*x_b)./(N*alpha_a + N*alpha_c);
% L1 = 0;
% lambda = L1;
% qaz1 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz1 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
% W_a_qaz1 =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
% W_c_qaz1 =  a*qaz1.^(b) - t1_H(x,:) - t2_L.*qaz1;
% W_a_qbz1 =  a*qbz1.^(b) - t1_L - t2_H(x,:).*qbz1;
```

```
% W_c_qbz1 =  a*qbz1.^(b) - t1_H(x,:) - t2_L.*qbz1;
% LHS1 = x_bar.*(W_a_qaz1-W_c_qaz1);
% RHS1 = x_b.*(W_a_qbz1-W_c_qbz1);
% click1 = (abs(LHS1 - RHS1));
%
% L2 = alpha_c + alpha_d;
% lambda = L2;
% qaz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz2 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
% W_a_qaz2 =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
% W_c_qaz2 =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
% W_a_qbz2 =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
% W_c_qbz2 =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
% LHS2 = x_bar.*(W_a_qaz2-W_c_qaz2);
% RHS2 = x_b.*(W_a_qbz2-W_c_qbz2);
% click2 = (abs(LHS2 - RHS2));
%
% count = 1;
% click =
1000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000;
% while click > 0.0000000000001;
%
%     if count > 100
%         disp(['click is:   ' num2str(click)])
%         disp(['At line 1073'])
%         keyboard
%     end
%     count = count +1;
%
%     L3 = (2/3)*L1 + L2/3;
%     lambda = L3;
%     qaz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
%     qbz3 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
%     W_a_qaz3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
%     W_c_qaz3 =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
%     W_a_qbz3 =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
%     W_c_qbz3 =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
%     LHS = x_bar.*(W_a_qaz3-W_c_qaz3);
%     RHS = x_b.*(W_a_qbz3-W_c_qbz3);
%     click3 = (abs(LHS - RHS));
%
%     L4 = (1/2)*L1 + L2/2;
%     lambda = L4;
%     qaz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
%     qbz4 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
%     W_a_qaz4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
%     W_c_qaz4 =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
%     W_a_qbz4 =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
%     W_c_qbz4 =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
%     LHS = x_bar.*(W_a_qaz4-W_c_qaz4);
%     RHS = x_b.*(W_a_qbz4-W_c_qbz4);
```

```
%     click4 = (abs(LHS - RHS));
%
%     L5 = (1/3)*L1 + L2*(2/3);
%     lambda = L5;
%     qaz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
%     qbz5 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
%     W_a_qaz5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
%     W_c_qaz5 =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
%     W_a_qbz5 =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
%     W_c_qbz5 =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
%     LHS = x_bar.*(W_a_qaz5-W_c_qaz5);
%     RHS = x_b.*(W_a_qbz5-W_c_qbz5);
%     click5 = (abs(LHS - RHS));
%
%     if click3 == min([click3; click4; click5])
%         L1 = L1;
%         L2 = L4;
%         lambda = L3;
%         click = click3;
%         qaz = qaz3;
%         qbz = qbz3;
%     elseif click4 == min([click3; click4; click5])
%         L1 = L3;
%         L2 = L5;
%         lambda = L4;
%         click = click4;
%         qaz = qaz4;
%         qbz = qbz4;
%     else
%         L1 = L4;
%         L2 = L2;
%         lambda = L5;
%         click = click5;
%         qaz = qaz5;
%         qbz = qbz5;
%     end
% end



%*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&
%*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&*&
% Code for the function: Solution21elambdasearch

% function x_b = Solution21elambdasearch(input)
% global lambda x qaz qbz N t2_H t1_H t1_L t2_L alpha_c alpha_d alpha_b alpha_a
delta2 delta2 a b fbx_d
%
% x_b = input;
% x_c = x_b;
% x_a = (1 - N*alpha_d*fbx_d - N*alpha_b*x_b - N*alpha_c*x_c)./(N*alpha_a );

% L1 = 0;
```

```
% lambda = L1;
% qaz1 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz1 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
% W_a_qaz1 =  a*qaz1.^(b) - t1_L - t2_H(x,:).*qaz1;
% W_c_qaz1 =  a*qaz1.^(b) - t1_H(x,:) - t2_L.*qaz1;
% W_a_qbz1 =  a*qbz1.^(b) - t1_L - t2_H(x,:).*qbz1;
% W_c_qbz1 =  a*qbz1.^(b) - t1_H(x,:) - t2_L.*qbz1;
% LHS1 = x_a*qaz1 ;
% RHS1 = x_b.*qbz1;
% click1 = (abs(LHS1 - RHS1));
%
% L2 = alpha_c + alpha_d;
% lambda = L2;
% qaz2 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz2 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
% W_a_qaz2 =  a*qaz2.^(b) - t1_L - t2_H(x,:).*qaz2;
% W_c_qaz2 =  a*qaz2.^(b) - t1_H(x,:) - t2_L.*qaz2;
% W_a_qbz2 =  a*qbz2.^(b) - t1_L - t2_H(x,:).*qbz2;
% W_c_qbz2 =  a*qbz2.^(b) - t1_H(x,:) - t2_L.*qbz2;
% LHS2 = x_a*qaz2;
% RHS2 = x_b.*qbz2;
% click2 = (abs(LHS2 - RHS2));
%
% count = 1;
% click =
100000000000000000000000000000000000000000000000000000000000000000000000000000;
% while click > 0.0000000000001;
%
%     if count > 100
%         disp(['click is:   ' num2str(click)])
%         disp(['At line 1073'])
%         keyboard
%     end
%     count = count +1;
%
%     L3 = (2/3)*L1 + L2/3;
%     lambda = L3;
%      qaz3 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz3 = ((t2_H(x,:) + (alpha_c + alpha_d -
lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
% W_a_qaz3 =  a*qaz3.^(b) - t1_L - t2_H(x,:).*qaz3;
% W_c_qaz3 =  a*qaz3.^(b) - t1_H(x,:) - t2_L.*qaz3;
% W_a_qbz3 =  a*qbz3.^(b) - t1_L - t2_H(x,:).*qbz3;
% W_c_qbz3 =  a*qbz3.^(b) - t1_H(x,:) - t2_L.*qbz3;
% LHS3 = x_a*qaz3 ;
% RHS3 = x_b.*qbz3;
%     click3 = (abs(LHS - RHS));
%
%     L4 = (1/2)*L1 + L2/2;
%     lambda = L4;
% qaz4 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
% qbz4 = ((t2_H(x,:) + (alpha_c + alpha_d -
```

```
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
 % W_a_qaz4 =  a*qaz4.^(b) - t1_L - t2_H(x,:).*qaz4;
 % W_c_qaz4 =  a*qaz4.^(b) - t1_H(x,:) - t2_L.*qaz4;
 % W_a_qbz4 =  a*qbz4.^(b) - t1_L - t2_H(x,:).*qbz4;
 % W_c_qbz4 =  a*qbz4.^(b) - t1_H(x,:) - t2_L.*qbz4;
 % LHS4 = x_a*qaz4 ;
 % RHS4 = x_b.*qbz4;
 %     click4 = (abs(LHS - RHS));
 %
 %     L5 = (1/3)*L1 + L2*(2/3);
 %     lambda = L5;
 % qaz5 = ((t2_H(x,:) + (lambda).*delta2(x,:)./alpha_a)./(a.*b)).^(1/(b-1));
 % qbz5 = ((t2_H(x,:) + (alpha_c + alpha_d -
 lambda).*delta2(x,:)./alpha_b)./(a.*b)).^(1/(b-1));
 % W_a_qaz5 =  a*qaz5.^(b) - t1_L - t2_H(x,:).*qaz5;
 % W_c_qaz5 =  a*qaz5.^(b) - t1_H(x,:) - t2_L.*qaz5;
 % W_a_qbz5 =  a*qbz5.^(b) - t1_L - t2_H(x,:).*qbz5;
 % W_c_qbz5 =  a*qbz5.^(b) - t1_H(x,:) - t2_L.*qbz5;
 % LHS5 = x_a*qaz5 ;
 % RHS5 = x_b.*qbz5;
 %     click5 = (abs(LHS - RHS));
 %
 %     if click3 == min([click3; click4; click5])
 %         L1 = L1;
 %         L2 = L4;
 %         lambda = L3;
 %         click = click3;
 %         qaz = qaz3;
 %         qbz = qbz3;
 %     elseif click4 == min([click3; click4; click5])
 %         L1 = L3;
 %         L2 = L5;
 %         lambda = L4;
 %         click = click4;
 %         qaz = qaz4;
 %         qbz = qbz4;
 %     else
 %         L1 = L4;
 %         L2 = L2;
 %         lambda = L5;
 %         click = click5;
 %         qaz = qaz5;
 %         qbz = qbz5;
 %     end
 % end
```

```
% This code is used to construct the sequential mechanism simulations with
% recall in table 7 and 6

clear all
global q11 alpha_a alpha_b alpha_c alpha_d fbq_a fbq_b fbq_c fbq_d a b t1_L t2_L t1_H↙
t2_H delta1 delta2 V_last

rand('state',201)
simsize = 101;

% Recall costs are theta1 + theta2*q
% The valuation of the quality for the buyer is V(q)=3*q^(1/2)
a = 3;
b = 1/2;
% V = a*q^(b);

% need to pick the probs of individual types
alpha_a = 40/100;
alpha_b = 10/100;
alpha_c = 10/100;
alpha_d = 40/100;

%need to set the number of bidders
N = 2; %This is hard coded at 2 in the recall model

% Pick the types
t1_L = 1;
t2_L = 1;
delta1 = [0.00001; (1/(simsize-1)).*[1:simsize-1]'.*1.125];
delta2 = 1.*ones(simsize,1);
t1_H = 1 + delta1;
t2_H = 1 + delta2;

% define first best qualities

fbq_a = (t2_H/(a*b)).^(1/(b-1));
fbq_b = fbq_a;
fbq_c = (t2_L/(a*b)).^(1/(b-1));
fbq_d = fbq_c;

% define the first best welfares

fbW_a =  a*fbq_a.^(b) - t1_L - t2_H.*fbq_a;
fbW_b =  a*fbq_b.^(b) - t1_H - t2_H.*fbq_b;
fbW_c =  a*fbq_c.^(b) - t1_H - t2_L.*fbq_c;
fbW_d =  a*fbq_d.^(b) - t1_L - t2_L.*fbq_d;

%*************************************************************************
%*************************************************************************
%*************************************************************************
%*************************************************************************
```

```
% 1. Standard Sequential Mechanism with no Recall [hard coded to N=2]

% Final Round Continuation value

alphaH = alpha_a + alpha_b;
alphaL = alpha_d + alpha_c;
Z = alphaL./alphaH;
q11 = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
W_hH = a*q11.^(b) - t1_H - t2_H.*q11;
V_last = alphaH.*W_hH + alphaL.*(fbW_c - delta2.*q11);


% Next Round Continuations
for i = 1:N-1
    % Case 1 lL = Kn
    r1 = alpha_d.*fbW_d;
    Case1 = r1 + (1-alpha_d).*V_last;

    % Case 2 lL lH = Kn
    alphaH = alpha_a;
    alphaL = alpha_d;
    Z = alphaL./alphaH;
    qstar = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
    W_lH = a*qstar.^(b) - t1_L - t2_H.*qstar;
    condition = ([delta1 - delta2.*qstar < 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000000000000000000)) ;
    r2 = alpha_d.*(fbW_d - delta2.*qstar) + alpha_a.*(W_lH) + condition;
    Case2 = r2 + (alpha_c + alpha_b).*V_last;
    Case2 = Case2 ;

    % Case 3 lL hL = Kn
    r3 = (alpha_c + alpha_d).*fbW_c;
    Case3 = r3 + (alpha_a + alpha_b).*V_last;

    %     This is code for the old case 4 solution in the earlier draft
    %
    %     Case 4 lL hL lH = Kn
    %     alphaH = alpha_a;
    %     alphaL = alpha_d + alpha_c;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H.*qstar;
    %     condition = ([delta1 - delta2.*qstar > 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000000000000000000)) ;
    %     r4 = (alpha_d + alpha_c).*(fbW_d - delta2.*qstar) + alpha_a.*(W_lH) +↙
condition;
    %     Case4 = r4 + alpha_b.*V_last;
    %     Case4 = Case4 + condition;

    % Case 4 lL hL lH = Kn [This is the new code: NEED TO CONFIRM IT IS THE
```

```
    % SAME AS WHAT IS IN THE FINAL VERSION OF THE PAPER
    qstar1 = [(t2_H + ((alpha_d + alpha_c)/alpha_a).*delta2)./(a*b)].^(1/(b-1));
    pstar1 = t1_L + t2_H.*qstar1;
    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
    p2star = t1_H + t2_L.*fbq_c;
    q2star1 = min([ (delta1./delta2), fbq_a],[],2);
    p2star1 =  t1_L + t2_H.*q2star1;
    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d) + (a*qstar1.^(b) - pstar1).*↙
(alpha_a );
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d) + (a*q2star1.^(b) - p2star1).*↙
(alpha_a );
    s = pstar > p2star;
    ss = pstar <= p2star;
    Case4 = Vstar.*s + V2star.*ss + alpha_b.*V_last;

    RevenueNoRecall = max([Case1 Case2 Case3 Case4],[],2);
    zz = [Case1 Case2 Case3 Case4]==[RevenueNoRecall RevenueNoRecall RevenueNoRecall↙
RevenueNoRecall];
    zz = zz*[1 3 2 4]';


    V_last = RevenueNoRecall ;
end
ContractRecorderNR = zz;
RevenueNoRecall = RevenueNoRecall./2;

% Now to do the case with recall.

% MAIN CODING

% First Step: Set up the matric that will record everything about the game
% and allow quick solving at the end. Most of this code is about filling in
% this matrix. This matrix is called GAME_SUMMARY.

% The first three columns of GAME_SUMMARY record the choices at each info set of the↙
buyer.
% The first column is the offer in the first info set, the second the second offer and↙
the
% third is the choice of supplier in the recall round

% The offers are coded 1,2,3,4,5 according to who accepts. The code is:
% 1 = lL
% 2 = lL hL
% 3 = lL lH
% 4 = lL hL lH
% 5 = lL hL lH hH
% 6 = none

% The next three columns GAME_SUMMARY(:,4:6) are the payoffs at each terminal node
% corresponding to acceptance of the offers made at each info set
```

```
% The next three columns GAME_SUMMARY(:,7:9) are the expected payoffs, conditional at↙
being at
% each info set

% The last six columns GAME_SUMMARY(:,10:12) and GAME_SUMMARY(:,13:15) are there to↙
record which choices are sequentially
% rational at each info set. The equilibrium choices are read off this.

EXPECTED_BUYER_UTILITY = zeros(simsize,1); % This is what gives buyer revenue per seller
ContractRecorder = zeros(simsize,3); %This records the contracts chose along the↙
eqilibrium path for each set of parameter values
CurveLook = zeros(simsize,5);
Offer4Tracking =zeros(simsize,10);

% HAVE TO LOOP THIS WHOLE THING....
for i = 1:simsize

    GAME_SUMMARY = zeros(72,12);
    x = ones(6,1);
    y = cumsum(ones(6,1));
    z = [kron(x,y) kron(y,x)];
    z = [z ones(size(z,1),1) ; z ones(size(z,1),1)*2];
    z = sortrows(z,[1 2]);
    GAME_SUMMARY(:,1:3) = z;

    % The above block of code just sets up a complete set of offers and who
    % to recall. Given Lemma 5 in the solution for this game it may look
    % strange to be explicitly computing everything for when supplier 2
    % is recalled. However, this provides a helpful check on the code, so
    % it actually is helpful. The revenue solutions should have all be the
    % same at stage 2 and, since the code used is the same as for supplier 1,
    % this helps check that the code not buggy.

    % For each of the six type of offer work out the continuation value in the
    % recall round if that offer is recalled.

    % Offer 1 = lL [This offer refers to the offer rejected in stage 1 or 2,
    % depending on who is ultimately recalled]

    alphaALL = alpha_a + alpha_b + alpha_c;
    alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    %     This next line of code works out where this contract is offered given
    %     the pattern of previous offers

    z = [GAME_SUMMARY(:,1) == 1 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 1↙
```

```
].*[GAME_SUMMARY(:,3) == 2 ];
    VV =  V.*z;

    % Offer 2 = lL hL

    V = fbW_b(i,:);
    z = [GAME_SUMMARY(:,1) == 2 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 2↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 3 = lL lH

    alphaALL =  alpha_b + alpha_c;
    alphaH =  alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 3 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 3↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 4 = lL hL lH

    V = fbW_b(i,:);
    z = [GAME_SUMMARY(:,1) == 4 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 4↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 5 = lL hL lH hH

    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 5 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 5↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

 % Offer 6 = none [This offer refers to the offer rejected in stage 1 or 2,
    % depending on who is ultimately recalled]

    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
```

```
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 6 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 6↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Now update the GAME SUMMARY

    GAME_SUMMARY(:,6) = VV;
    GAME_SUMMARY(:,9) = VV;

    %     % NOW NEED TO DEAL WITH THE NEXT INFO SET: ie. Stage 2.
    %
    %     % Dealing with all those stage 2 offers where there is no recall chance: This
    %     % section of code is really only providing a check on the next section
    %     % where the matrix GSPr is introduced and actually does the real work.
    %     % This section is not used other than feeding into GS_CHECK (for
    %     % checking GSPr)
    %
    %
    %     % Offer 1 = lL
    %
    %     V = fbW_d;
    %     zz = [GAME_SUMMARY(:,1) == 1 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = V.*zz;
    %     zzz =  [GAME_SUMMARY(:,2) == 1 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = V.*zzz;
    %
    %     % Offer 2 = lL hL
    %
    %     V = fbW_c(i,:);
    %     zz = [GAME_SUMMARY(:,1) == 2 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 2 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 3 = lL lH
    %     alphaALL = alpha_a + alpha_d;
    %     alphaH = alpha_a/alphaALL;
    %     alphaL = alpha_d/alphaALL;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H(i,:).*qstar;
    %     condition = ([delta1(i,:) - delta2(i,:).*qstar < 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000)) ;
    %     V = alphaL.*(fbW_d - delta2(i,:).*qstar) + alphaH.*(W_lH) + condition;
    %     zz = [GAME_SUMMARY(:,1) == 3 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
```

```
    %     zzz = [GAME_SUMMARY(:,2) == 3 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 4 = lL hL lH
    %     alphaALL = alpha_a + alpha_d + alpha_c;
    %     alphaH = alpha_a;
    %     alphaL = alpha_d + alpha_c;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H(i,:).*qstar;
    %     condition = ([delta1(i,:) - delta2(i,:).*qstar > 0].*↙
(-10000000000000000000000000000000000000000000000000000000000000000)) ;
    %     V = alphaL.*(fbW_d - delta2(i,:).*qstar) + alphaH.*(W_lH) + condition;
    %     zz = [GAME_SUMMARY(:,1) == 4 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 4 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 5 = lL hL lH hH (here there is no distinction between recall and
    %     % not as if get recalled just gest offered the same thing again)
    %
    %     alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    %     alphaH =  (alpha_a + alpha_b)./alphaALL;
    %     alphaL = (alpha_c + alpha_d)./alphaALL;
    %     Z = alphaL./alphaH;
    %     qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) -↙
delta2(i,:).*qr1);
    %
    %     zz = [GAME_SUMMARY(:,1) == 5 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 5 ] ;
    %     VVV = VVV + V.*zzz;
    %
    %     GAME_SUMMARY(:,4) = VV;
    %     GAME_SUMMARY(:,5) = VVV;


    % NOW NEED TO DEAL WITH THE OFFERS THAT MAY GET RECALLED

    %     What is done here is that a matrix called GSPr is introduced. The
    %     whole point is to construct the transition probabilities of moving
    %     from one stage to another given the offers (this will be given by
    %     GSPr(:,4:7)) and the returns from stage 1 and 2, GSPr(:,8:9) which will
    %     ultimately be fed back into GAME_SUMMARY(:,4:5). Note, GSPr(:,5) and
    %     GSPr(:,6) will give the endogenous discount factors for supplier 1
    %     and 2 respectively.

    GSPr = GAME_SUMMARY(:,1:3);

    reject1 = 1-alpha_d; reject2 = 1-alpha_c - alpha_d; reject3 = 1-alpha_a - alpha_d;
```

```
    reject4 = 1-alpha_a - alpha_d - alpha_c; reject5 = 0; reject6 = 1;

    z1 = [GSPr(:,1) == 1] ; z2 = [GSPr(:,1) == 2] ; z3 = [GSPr(:,1) == 3] ; z4 = [GSPr↙
(:,1) == 4] ;
    z5 = [GSPr(:,1) == 5] ; z6 = [GSPr(:,1) == 6] ; zz1 = [GSPr(:,2) == 1] ; zz2 = [GSPr↙
(:,2) == 2] ; zz3 = [GSPr(:,2) == 3] ;
    zz4 = [GSPr(:,2) == 4] ; zz5 = [GSPr(:,2) == 5] ;zz6 = [GSPr(:,2) == 6] ;

    GSPr(:,4) = z1.*reject1 + z2.*reject2 + z3.*reject3 + z4.*reject4 + z5.*reject5 +↙
z6.*reject6 ;
    GSPr(:,5) = zz1.*reject1 + zz2.*reject2 + zz3.*reject3 + zz4.*reject4 + zz5.*reject5↙
+ zz6.*reject6;

    zzz = [GSPr(:,3) == 1];
    GSPr(:,6) = zzz.*GSPr(:,5);
    zzz = [GSPr(:,3) == 2];
    GSPr(:,7) = zzz;
    GSPr(:,8) = 0;
    GSPr(:,9) = 0;


    % Offer 1 = lL

    alphaALL = alpha_a + alpha_b + alpha_c;
    alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    pr2 = t1_H(i,:) + delta2(i,:).*qr1 + t2_L.*fbq_c;

    z = [GSPr(:,1) == 1];
    zz = [GSPr(:,2) == 1];

    GSPr(:,8) = GSPr(:,9)+[a*fbq_c.^(b) - pr2.*GSPr(:,6) - (t1_L + t2_L.*fbq_c).*(1-GSPr↙
(:,6))].*z;
    GSPr(:,9) = GSPr(:,9)+[a*fbq_c.^(b) - pr2.*GSPr(:,7) - (t1_L + t2_L.*fbq_c).*(1-GSPr↙
(:,7))].*zz;

    %     The above 2 lines look a little different from the lemmas in the
    %     solution but follow from rearranging the IIR constraint of the lL
    %     type. It's just easier to code up this way as it fits easier with
    %     existing batches of code.

    %     Also there is a somewhat subtle theory point here that is worth
    %     talkng about: When the stage 2 player is recalled with certainty
    %     there is a discontinuity at the limit in that the expected revenue
    %     at stage 2 has to be equal to the one-shot take it or leave it
    %     environment (see lemma 5 of the technical not that accompanies this).
    %     This is not how things are coded here, resulting in revenue being
    %     lower than that from contract 5 when stage 2 supplier recalled with
    %     certainty. The "correct" way to code would be to note that it is
```

```
    %     impossible to exclude the hL type in this setting and disallow the
    %     lL type contract in stgae 2 if stage 2 is getting recalled with certainty.
    %     However the results in the main paper tell us that such coding is not
    %     needed as the "5" contract will always do better than the "1" contract
    %     in stage 2 with certain recall anyway. So easier to just run the code as is.
    %     This same issue arises elsewhere in the code, but this is where it
    %     turns up first

    % Offer 2 = lL hL

    qr1 = fbq_a(i,:);

    z = [GSPr(:,1) == 2];
    zz = [GSPr(:,2) == 2];

    GSPr(:,8) = GSPr(:,8)+ [a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPr(:,6) - (t1_H(i,:) +↙
t2_L.*fbq_c)].*z;
    GSPr(:,9) = GSPr(:,9)+[a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPr(:,7) - (t1_H(i,:) +↙
t2_L.*fbq_c)].*zz;


    % Offer 3 = lL lH

    alphaALL = alpha_c + alpha_b;
    alphaH = alpha_b/alphaALL;
    alphaL = alpha_c/alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
    qr2 = fbq_c;
    pr2 = pr1 - t2_L.*qr1 + t2_L.*qr2;
    qstar1 = [(t2_H(i,:) + (alpha_d/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-1));

    gamma = GSPr(:,6);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

    q2star1 = qr1.*gamma;
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b) -↙
p2star1).*alpha_a./(alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;
```

```
    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1 <↙
0].*(-1000000000000000000000000000000000000000000000000000000000000000)) ;

    % The above line severely penalises if IIR(hL) is not violated.

    V = V + condition;

    z = [GSPr(:,1) == 3];
    GSPr(:,8) = GSPr(:,8)+[V].*z;

    %     Now we just do the same thing for supplier 2, but changing the
    %     discount rate

    gamma = GSPr(:,7);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

    q2star1 = qr1.*gamma;
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b) -↙
p2star1).*alpha_a./(alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1 <↙
0].*(-1000000000000000000000000000000000000000000000000000000000000000)) ;

    V = V + condition;

    zz = [GSPr(:,2) == 3];
    GSPr(:,9) =GSPr(:,9)+ V.*zz;

    % Offer 4 = lL lH hL

    qr1 = fbq_b(i,:);
    pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
    %%%%%%%%
    qstar1 = [(t2_H(i,:) + ((alpha_d + alpha_c)/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-↙
```

```
1));

    gamma = GSPr(:,6);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;

    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = [pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)].*gamma + [t1_H(i,:) + t2_L.*fbq_c];

    q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones↙
(size(gamma,1),1)],[],2);
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) +↙
(a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a)↙
+ (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    z = [GSPr(:,1) == 4];
    GSPr(:,8) = GSPr(:,8)+[V].*z;

    % Repeat for supplier 2

    gamma = GSPr(:,7);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
    p2star = (pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)).*gamma + [t1_H(i,:) + t2_L.*fbq_c];

    q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones↙
(size(gamma,1),1)],[],2);
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) +↙
(a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a)↙
+ (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;
```

```
    z = [GSPr(:,2) == 4];
    GSPr(:,9) =GSPr(:,9)+ [V].*z;

    %     The block of code below is for looking inside all the code above and
    %     seeing how its bits are moving together. So just for debugging
    %     V = [V.*z Vstar V2star s ss];
    margin1 = a*qstar1.^(b) - pstar1;
    margin2 = a*q2star1.^(b) - p2star1;
    Offer4Tracking(i,1:8) = [pstar1(7,1) pstar(7,1) p2star(7,1) q2star1(7,1) p2star1↙
(7,1) Vstar(7,1) V2star(7,1) s(7,1)];
    Offer4Tracking(i,9:12) = [gamma(7,1) V(7,1) margin1(7,1) margin2(7,1) ];


    % Offer 5 = lL lH hL hH

    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    zz = [GSPr(:,1) == 5 ] ;
    zzz = [GSPr(:,2) == 5 ] ;

    GSPr(:,8) =GSPr(:,8)+ [V].*zz;
    GSPr(:,9) =GSPr(:,9)+ [V].*zzz;


        % Offer 6 = none

    z = [GSPr(:,1) == 2];
    zz = [GSPr(:,2) == 2];

    GSPr(:,8) = GSPr(:,8)+ 0.*z;
    GSPr(:,9) = GSPr(:,9)+ 0.*zz;


    % In this section the code basically follows the construction of GSPr but
    % deals specifically with the contract sequences that give rise to the
    % possibility of mixing over whom to recall. The relevant sequences are
    % [1 1]
    % [2 2]
    % [2 4]
    % [3 3]
    % [4 2]
    % [4 4]
    % Once we work out the optimal mixing we plug it back into the relevant bit
    % in GSPr
```

```
    mixingsequences = [ [1 1]; [2 2]; [2 4]; [3 3]; [4 2];[4 4] ];
    mixsummaryrec = [];
    for mixline = 1:6
        GSPmix = [ones(1001,1)*mixingsequences(mixline,:) [linspace(0,1,1001)]'];

        % GSPmix(i,3) = 0 means supplier 1 gets recalled with certainty

        reject1 = 1-alpha_d; reject2 = 1-alpha_c - alpha_d; reject3 = 1-alpha_a -↙
alpha_d;
        reject4 = 1-alpha_a - alpha_d - alpha_c; reject5 = 0;

        z1 = [GSPmix(:,1) == 1] ; z2 = [GSPmix(:,1) == 2] ; z3 = [GSPmix(:,1) == 3] ; z4↙
= [GSPmix(:,1) == 4] ;
        z5 = [GSPmix(:,1) == 5] ; zz1 = [GSPmix(:,2) == 1] ; zz2 = [GSPmix(:,2) == 2] ;↙
zz3 = [GSPmix(:,2) == 3] ;
        zz4 = [GSPmix(:,2) == 4] ; zz5 = [GSPmix(:,2) == 5] ;

        GSPmix(:,4) = z1.*reject1 + z2.*reject2 + z3.*reject3 + z4.*reject4 + z5.↙
*reject5 ;
        GSPmix(:,5) = zz1.*reject1 + zz2.*reject2 + zz3.*reject3 + zz4.*reject4 + zz5.↙
*reject5 ;

        zzz = 1-GSPmix(:,3);
        GSPmix(:,6) = zzz.*GSPmix(:,5);
        zzz = GSPmix(:,3);
        GSPmix(:,7) = zzz;
        GSPmix(:,8) = 0;
        GSPmix(:,9) = 0;

        % Offer 1 = lL

        alphaALL = alpha_a + alpha_b + alpha_c;
        alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
        alphaL = alpha_c./alphaALL;
        Z = alphaL./alphaH;
        qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
        pr2 = t1_H(i,:) + delta2(i,:).*qr1 + t2_L.*fbq_c;

        z = [GSPmix(:,1) == 1];
        zz = [GSPmix(:,2) == 1];

        GSPmix(:,8) = GSPmix(:,9)+[a*fbq_c.^(b) - pr2.*GSPmix(:,6) - (t1_L + t2_L.↙
*fbq_c).*(1-GSPmix(:,6))].*z;
        GSPmix(:,9) = GSPmix(:,9)+[a*fbq_c.^(b) - pr2.*GSPmix(:,7) - (t1_L + t2_L.↙
*fbq_c).*(1-GSPmix(:,7))].*zz;

        %     The above 2 lines look a little different from the lemmas in the
        %     solution but follow from rearranging the IIR constraint of the lL
        %     type. It's just easier to code up this way as it fits easier with
        %     existing batches of code.
```

```
        % Offer 2 = lL hL

        qr1 = fbq_a(i,:);

        z = [GSPmix(:,1) == 2];
        zz = [GSPmix(:,2) == 2];

        GSPmix(:,8) = GSPmix(:,8)+ [a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPmix(:,6) - (t1_H(i,:) + t2_L.*fbq_c)].*z;
        GSPmix(:,9) = GSPmix(:,9)+[a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPmix(:,7) - (t1_H(i,:) + t2_L.*fbq_c)].*zz;


        % Offer 3 = lL lH

        alphaALL = alpha_c + alpha_b;
        alphaH = alpha_b/alphaALL;
        alphaL = alpha_c/alphaALL;
        Z = alphaL./alphaH;
        qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
        pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
        qr2 = fbq_c;
        pr2 = pr1 - t2_L.*qr1 + t2_L.*qr2;
        qstar1 = [(t2_H(i,:) + (alpha_d/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-1));

        gamma = GSPmix(:,6);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

        p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

        q2star1 = qr1.*gamma;
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) - pstar1).*alpha_a./(alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b) - p2star1).*alpha_a./(alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1 < 0].*(-10000000000000000000000000000000000000000000000000000000000000000)) ;

        % The above line severely penalises if IIR(hL) is not violated.
```

```
        V = V + condition;

        z = [GSPmix(:,1) == 3];
        GSPmix(:,8) = GSPmix(:,8)+[V].*z;

        %     Now we just do the same thing for supplier 2, but changing the
        %     discount rate

        gamma = GSPmix(:,7);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

        p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

        q2star1 = qr1.*gamma;
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b)↙
- p2star1).*alpha_a./(alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1↙
< 0].*(-1000000000000000000000000000000000000000000000000000000000000000)) ;

        V = V + condition;

        zz = [GSPmix(:,2) == 3];
        GSPmix(:,9) =GSPmix(:,9)+ V.*zz;

        % Offer 4 = lL lH hL

        qr1 = fbq_b(i,:);
        pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
        %%%%%%%%
        qstar1 = [(t2_H(i,:) + ((alpha_d + alpha_c)/alpha_a).*delta2(i,:))./(a*b)].^(1/↙
(b-1));

        gamma = GSPmix(:,6);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;

        pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
```

```
        p2star = [pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)].*gamma + [t1_H(i,:) + t2_L.*fbq_c];

        q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones(size(gamma,1),1)],[],2);
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) + (a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) + (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        z = [GSPmix(:,1) == 4];
        GSPmix(:,8) = GSPmix(:,8)+[V].*z;

        % Repeat for supplier 2

        gamma = GSPmix(:,7);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
        p2star = (pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)).*gamma + [t1_H(i,:) + t2_L.*fbq_c];

        q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones(size(gamma,1),1)],[],2);
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) + (a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) + (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        z = [GSPmix(:,2) == 4];
        GSPmix(:,9) =GSPmix(:,9)+ [V].*z;

%         this is for debuigging
        z = find(z);
```

```
        Lookit = [GSPmix(z,3) V(z,:) s(z,:) ss(z,:) q1(z,:)];



        GAME_SUMMARY(:,4:5) = GSPr(:,8:9);
        GAME_SUMMARY(:,9) = GAME_SUMMARY(:,6);
        GAME_SUMMARY(:,8) = GSPr(:,5).*GAME_SUMMARY(:,6) + (1-GSPr(:,5)).*GAME_SUMMARY↙
(:,5);
        GAME_SUMMARY(:,7) = GSPr(:,4).*GAME_SUMMARY(:,8) + (1-GSPr(:,4)).*GAME_SUMMARY↙
(:,4);

        mixsummary = GSPmix(:,1:3);
        mixsummary(:,4:5) = GSPmix(:,8:9);
        s = find([GAME_SUMMARY(:,1) == GSPmix(1,1)].*[GAME_SUMMARY(:,2) == GSPmix↙
(1,2)]);
        mixsummary(:,6) = min(GAME_SUMMARY(s,6));
        mixsummary(:,9) = mixsummary(:,6);
        mixsummary(:,8) = GSPmix(:,5).*mixsummary(:,6) + (1-GSPmix(:,5)).*mixsummary(:,↙
5);
        mixsummary(:,7) = GSPmix(:,4).*mixsummary(:,8) + (1-GSPmix(:,4)).*mixsummary(:,↙
4);

        % mixsummary(:,7:9) are the expected payoffs, conditional at being at
        % each info set ie. Stage 1 2 & 3 resp.
        mixsummaryrec = [mixsummaryrec;mixsummary];

    end
    mixsummaryrec(:,10:17) = 0;
    mixsummaryrec(:,17) = 1;
    % NOW TO SOLVE THE GAME

    %     GS_CHECK = GAME_SUMMARY(:,4:5);
    GAME_SUMMARY(:,4:5) = GSPr(:,8:9);
    GAME_SUMMARY(:,9) = GAME_SUMMARY(:,6);
    GAME_SUMMARY(:,8) = GSPr(:,5).*GAME_SUMMARY(:,6) + (1-GSPr(:,5)).*GAME_SUMMARY(:,5);
    GAME_SUMMARY(:,7) = GSPr(:,4).*GAME_SUMMARY(:,8) + (1-GSPr(:,4)).*GAME_SUMMARY(:,4);
    nGS = find([GAME_SUMMARY(:,1) == 1].*[GAME_SUMMARY(:,2) == 1] + [GAME_SUMMARY(:,1)↙
== 2].*[GAME_SUMMARY(:,2) == 2] + [GAME_SUMMARY(:,1) == 2].*[GAME_SUMMARY(:,2) == 4] +↙
[GAME_SUMMARY(:,1) == 4].*[GAME_SUMMARY(:,2) == 2] + [GAME_SUMMARY(:,1) == 4].*↙
[GAME_SUMMARY(:,2) == 4]  + [GAME_SUMMARY(:,1) == 3].*[GAME_SUMMARY(:,2) == 3]);
    GAME_SUMMARY(nGS,:) = [];
    GAME_SUMMARY(:,13:17) = 0;

    Game_Summary = [GAME_SUMMARY;mixsummaryrec];

    % GAME_SUMMARY(:,7:9) are the expected payoffs, conditional at being at
    % each info set ie. Stage 1 2 & 3 resp.

    %     I'm just going to solve everything backwards this will tell me what
    %     is optimal at each info set/stage. GAME_SUMMARY(:,13:15) will be
```

```
    %     columns of ones and zeros with a one corresponding to a offer that is
    %     a best response at the info set.

    % note here i take advantage of how the first three columns are sorted
    Game_Summary(:,10:12) = Game_Summary(:,7:9);
    ContractsList = unique(Game_Summary(:,1:2),'rows');
    [nCL kCL] = size(ContractsList);
    for j = 1:nCL
        y = find([ContractsList(j,1) == Game_Summary(:,1)].*[ContractsList(j,2) ==↙
Game_Summary(:,2)]);
        x = max([Game_Summary(y,9)]);
        ny = size(y,1);
        x = x.*ones(ny,1);
        x = [x == Game_Summary(y,9)];
        Game_Summary(y,15) = x;
    end
    Game_Summary(:,11) = Game_Summary(:,8) + (1-Game_Summary(:,15)).*↙
(-10000000000000000000000000000);
    %     What the line above does is penalise any sequence of offers that would lead to↙
a
    %     subsequent offer violating sequential rationality

    % now need to think about what is best at stage 2, this gest tricky
    % when dealing with the sequences that invite mixing as they can'y be
    % compared to themselves as this would involve comparing different
    % equilibria
    ContractsList = unique(Game_Summary(:,1));
    [nCL kCL] = size(ContractsList);
    for j = 1:nCL
        if ContractsList(j,1) == 2
        olp2 = find([2 == Game_Summary(:,1)].*[Game_Summary(:,17) == 1].*[2 ==↙
Game_Summary(:,2)]);
        olp4 = find([2 == Game_Summary(:,1)].*[Game_Summary(:,17) == 1].*[4 ==↙
Game_Summary(:,2)]);
        test = [max(Game_Summary(olp2,11)) < min(Game_Summary(olp4,11))] ;
        if test == 1
            Game_Summary(olp2,17) = 0;
        end
        test =  [max(Game_Summary(olp4,11)) < min(Game_Summary(olp2,11))];
        if test == 1
            Game_Summary(olp4,17) = 0;
        end
        end

        ynm = find([ContractsList(j,1) == Game_Summary(:,1)].*[Game_Summary(:,17) ==↙
0]);
        xnm = max([Game_Summary(ynm,11)]);
        ym = find([ContractsList(j,1) == Game_Summary(:,1)].*[Game_Summary(:,17) == 1]);
        xmin = min([Game_Summary(ym,11)]);

        if xmin > xnm
```

```
            Game_Summary(ym,14) = 1;
        else
            y = find(ContractsList(j,1) == Game_Summary(:,1));
            ny = size(y,1);
            x = xnm.*ones(ny,1);
            x = [x <= Game_Summary(y,11)];
            Game_Summary(y,14) = x;
        end
    end
    Game_Summary(:,10) = Game_Summary(:,7) + (1-Game_Summary(:,14)).*↙
(-10000000000000000000000000000);
    x = max(Game_Summary(:,10));
    nGS = size(Game_Summary,1);
    x = x.*ones(nGS,1);
    x = [x == Game_Summary(:,10)];
    Game_Summary(:,13) = x;


    % Now find the =m path: it'll be a row of ones in GAME_SUMMARY(:13:15).
    x = sum(Game_Summary(:,13:15),2);
    [n k] = find(x == 3);
    if isempty(n) == 1
        EXPECTED_BUYER_UTILITY(i,1) = -100000;
    else
        n = n(1);
        EXPECTED_BUYER_UTILITY(i,1) = Game_Summary(n,7);
        ContractRecorder(i,1:3) = Game_Summary(n,1:3);
        %         This next bit of code records the Expected Buyer Utility from the↙
following contracts
        %         (in this order): (1,2,1) (1,4,1) (1,1,1) (2,4,1) (3,3,1)
        CurveLook(i,1:5) = [Game_Summary(3,7) Game_Summary(7,7) Game_Summary(1,7)↙
Game_Summary(17,7) Game_Summary(25,7)];
        CurveLook(i,1:5) = max([CurveLook(i,1:5);zeros(1,5)]);

    end
end
RevenueRecall = EXPECTED_BUYER_UTILITY./2;

subplot(1,2,2)
plot(RevenueRecall,'b-')
subplot(1,2,1)
x = delta1;
plot(x,CurveLook(:,1),'b',x,CurveLook(:,2),'r+-',x,CurveLook(:,3),'k-',x,CurveLook(:,↙
4),'k:',x,CurveLook(:,5),'k-.')
legend('(1 2)','(1 4)','(1 1)','(2 4)','(3 3)')

cr = [ContractRecorder ContractRecorderNR 5.*ones(101,1)];
n = size(cr,1);
for i = 1:n-1
    if cr(n-i,:) == cr(n+1-i,:)
        cr(n+1-i,:) = [];
```

```
    end
end

return
```

```
% This code is used to construct the sequential mechanism simulations with recall in table 8

clear all
global q11 alpha_a alpha_b alpha_c alpha_d fbq_a fbq_b fbq_c fbq_d a b t1_L t2_L t1_H t2_H delta1 delta2 V_last

rand('state',201)
simsize = 101;

% Recall costs are theta1 + theta2*q
% The valuation of the quality for the buyer is V(q)=3*q^(1/2)
a = 3; %3
b = 1/2; %1/2
% V = a*q^(b);

% need to pick the probs of individual types
alpha_a = 40/100;  %to get more exclusion 5, 65, 5, 25
alpha_b = 10/100;
alpha_c = 10/100;
alpha_d = 40/100;

%need to set the number of bidders
N = 2; %This is hard coded at 2 in the recall model

% Pick the types
t1_L = 1;
t2_L = 1;
delta1 = [0.00001; (1/(simsize-1)).*[1:simsize-1]'.*1.125];
delta2 = 1.*ones(simsize,1);
t1_H = t1_L + delta1;
t2_H = t2_L + delta2;

% define first best qualities

fbq_a = (t2_H/(a*b)).^(1/(b-1));
fbq_b = fbq_a;
fbq_c = (t2_L/(a*b)).^(1/(b-1));
fbq_d = fbq_c;

% define the first best welfares

fbW_a =  a*fbq_a.^(b) - t1_L - t2_H.*fbq_a;
fbW_b =  a*fbq_b.^(b) - t1_H - t2_H.*fbq_b;
fbW_c =  a*fbq_c.^(b) - t1_H - t2_L.*fbq_c;
fbW_d =  a*fbq_d.^(b) - t1_L - t2_L.*fbq_d;

%*************************************************************************
%*************************************************************************
%*************************************************************************
%*************************************************************************
```

```
% 1. Standard Sequential Mechanism with no Recall [hard coded to N=2]

% Final Round Continuation value

alphaH = alpha_a + alpha_b;
alphaL = alpha_d + alpha_c;
Z = alphaL./alphaH;
q11 = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
W_hH = a*q11.^(b) - t1_H - t2_H.*q11;
V_last = alphaH.*W_hH + alphaL.*(fbW_c - delta2.*q11);


% Next Round Continuations
for i = 1:N-1
    % Case 1 lL = Kn
    r1 = alpha_d.*fbW_d;
    Case1 = r1 + (1-alpha_d).*V_last;

    % Case 2 lL lH = Kn
    alphaH = alpha_a;
    alphaL = alpha_d;
    Z = alphaL./alphaH;
    qstar = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
    W_lH = a*qstar.^(b) - t1_L - t2_H.*qstar;
    condition = ([delta1 - delta2.*qstar < 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000)) ;
    r2 = alpha_d.*(fbW_d - delta2.*qstar) + alpha_a.*(W_lH) + condition;
    Case2 = r2 + (alpha_c + alpha_b).*V_last;
    Case2 = Case2 ;

    % Case 3 lL hL = Kn
    r3 = (alpha_c + alpha_d).*fbW_c;
    Case3 = r3 + (alpha_a + alpha_b).*V_last;

    %     This is code for the old case 4 solution in the earlier draft
    %
    %     Case 4 lL hL lH = Kn
    %     alphaH = alpha_a;
    %     alphaL = alpha_d + alpha_c;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H + Z.*delta2)./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H.*qstar;
    %     condition = ([delta1 - delta2.*qstar > 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000)) ;
    %     r4 = (alpha_d + alpha_c).*(fbW_d - delta2.*qstar) + alpha_a.*(W_lH) +↙
condition;
    %     Case4 = r4 + alpha_b.*V_last;
    %     Case4 = Case4 + condition;

    % Case 4 lL hL lH = Kn [This is the new code: NEED TO CONFIRM IT IS THE
```

```
    % SAME AS WHAT IS IN THE FINAL VERSION OF THE PAPER
    qstar1 = [(t2_H + ((alpha_d + alpha_c)/alpha_a).*delta2)./(a*b)].^(1/(b-1));
    pstar1 = t1_L + t2_H.*qstar1;
    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
    p2star = t1_H + t2_L.*fbq_c;
    q2star1 = min([ (delta1./delta2), fbq_a],[],2);
    p2star1 =  t1_L + t2_H.*q2star1;
    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d) + (a*qstar1.^(b) - pstar1).*↙
(alpha_a );
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d) + (a*q2star1.^(b) - p2star1).*↙
(alpha_a );
    s = pstar > p2star;
    ss = pstar <= p2star;
    Case4 = Vstar.*s + V2star.*ss + alpha_b.*V_last;

    RevenueNoRecall = max([Case1 Case2 Case3 Case4],[],2);
    zz = [Case1 Case2 Case3 Case4]==[RevenueNoRecall RevenueNoRecall RevenueNoRecall↙
RevenueNoRecall];
    zz = zz*[1 3 2 4]';


    V_last = RevenueNoRecall ;
end
ContractRecorderNR = zz;
RevenueNoRecall = RevenueNoRecall./2;

% Now to do the case with recall.

% MAIN CODING

% First Step: Set up the matric that will record everything about the game
% and allow quick solving at the end. Most of this code is about filling in
% this matrix. This matrix is called GAME_SUMMARY.

% The first three columns of GAME_SUMMARY record the choices at each info set of the↙
buyer.
% The first column is the offer in the first info set, the second the second offer and↙
the
% third is the choice of supplier in the recall round

% The offers are coded 1,2,3,4,5 according to who accepts. The code is:
% 1 = lL
% 2 = lL hL
% 3 = lL lH
% 4 = lL hL lH
% 5 = lL hL lH hH
% 6 = none

% The next three columns GAME_SUMMARY(:,4:6) are the payoffs at each terminal node
% corresponding to acceptance of the offers made at each info set
```

```
% The next three columns GAME_SUMMARY(:,7:9) are the expected payoffs, conditional at↙
being at
% each info set

% The last six columns GAME_SUMMARY(:,10:12) and GAME_SUMMARY(:,13:15) are there to↙
record which choices are sequentially
% rational at each info set. The equilibrium choices are read off this.

EXPECTED_BUYER_UTILITY = zeros(simsize,1); % This is what gives buyer revenue per seller
ContractRecorder = zeros(simsize,3); %This records the contracts chose along the↙
eqilibrium path for each set of parameter values
CurveLook = zeros(simsize,5);
Offer4Tracking =zeros(simsize,10);

% HAVE TO LOOP THIS WHOLE THING....
for i = 1:simsize

    GAME_SUMMARY = zeros(72,12);
    x = ones(6,1);
    y = cumsum(ones(6,1));
    z = [kron(x,y) kron(y,x)];
    z = [z ones(size(z,1),1) ; z ones(size(z,1),1)*2];
    z = sortrows(z,[1 2]);
    GAME_SUMMARY(:,1:3) = z;

    % The above block of code just sets up a complete set of offers and who
    % to recall. Given Lemma 5 in the solution for this game it may look
    % strange to be explicitly computing everything for when supplier 2
    % is recalled. However, this provides a helpful check on the code, so
    % it actually is helpful. The revenue solutions should have all be the
    % same at stage 2 and, since the code used is the same as for supplier 1,
    % this helps check that the code not buggy.

    % For each of the six type of offer work out the continuation value in the
    % recall round if that offer is recalled.

    % Offer 1 = lL [This offer refers to the offer rejected in stage 1 or 2,
    % depending on who is ultimately recalled]

    alphaALL = alpha_a + alpha_b + alpha_c;
    alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    %     This next line of code works out where this contract is offered given
    %     the pattern of previous offers

    z = [GAME_SUMMARY(:,1) == 1 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 1↙
```

```
].*[GAME_SUMMARY(:,3) == 2 ];
    VV =  V.*z;

    % Offer 2 = lL hL

    V = fbW_b(i,:);
    z = [GAME_SUMMARY(:,1) == 2 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 2↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 3 = lL lH

    alphaALL =  alpha_b + alpha_c;
    alphaH =  alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 3 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 3↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 4 = lL hL lH

    V = fbW_b(i,:);
    z = [GAME_SUMMARY(:,1) == 4 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 4↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Offer 5 = lL hL lH hH

    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 5 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 5↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

     % Offer 6 = none [This offer refers to the offer rejected in stage 1 or 2,
    % depending on who is ultimately recalled]

    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
```

```
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    z = [GAME_SUMMARY(:,1) == 6 ].*[GAME_SUMMARY(:,3) == 1 ] + [GAME_SUMMARY(:,2) == 6↙
].*[GAME_SUMMARY(:,3) == 2 ];
    VV = VV + V.*z;

    % Now update the GAME SUMMARY

    GAME_SUMMARY(:,6) = VV;
    GAME_SUMMARY(:,9) = VV;

    %     % NOW NEED TO DEAL WITH THE NEXT INFO SET: ie. Stage 2.
    %
    %     % Dealing with all those stage 2 offers where there is no recall chance: This
    %     % section of code is really only providing a check on the next section
    %     % where the matrix GSPr is introduced and actually does the real work.
    %     % This section is not used other than feeding into GS_CHECK (for
    %     % checking GSPr)
    %
    %
    %     % Offer 1 = lL
    %
    %     V = fbW_d;
    %     zz = [GAME_SUMMARY(:,1) == 1 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = V.*zz;
    %     zzz =  [GAME_SUMMARY(:,2) == 1 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = V.*zzz;
    %
    %     % Offer 2 = lL hL
    %
    %     V = fbW_c(i,:);
    %     zz = [GAME_SUMMARY(:,1) == 2 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 2 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 3 = lL lH
    %     alphaALL = alpha_a + alpha_d;
    %     alphaH = alpha_a/alphaALL;
    %     alphaL = alpha_d/alphaALL;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H(i,:).*qstar;
    %     condition = ([delta1(i,:) - delta2(i,:).*qstar < 0].*↙
(-1000000000000000000000000000000000000000000000000000000000000000)) ;
    %     V = alphaL.*(fbW_d - delta2(i,:).*qstar) + alphaH.*(W_lH) + condition;
    %     zz = [GAME_SUMMARY(:,1) == 3 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
```

```
    %     zzz = [GAME_SUMMARY(:,2) == 3 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 4 = lL hL lH
    %     alphaALL = alpha_a + alpha_d + alpha_c;
    %     alphaH = alpha_a;
    %     alphaL = alpha_d + alpha_c;
    %     Z = alphaL./alphaH;
    %     qstar = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     W_lH = a*qstar.^(b) - t1_L - t2_H(i,:).*qstar;
    %     condition = ([delta1(i,:) - delta2(i,:).*qstar > 0].*↙
(-10000000000000000000000000000000000000000000000000000000000000000)) ;
    %     V = alphaL.*(fbW_d - delta2(i,:).*qstar) + alphaH.*(W_lH) + condition;
    %     zz = [GAME_SUMMARY(:,1) == 4 ].*[GAME_SUMMARY(:,3) == 2 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 4 ].*[GAME_SUMMARY(:,3) == 1 ];
    %     VVV = VVV + V.*zzz;
    %
    %     % Offer 5 = lL hL lH hH (here there is no distinction between recall and
    %     % not as if get recalled just gest offered the same thing again)
    %
    %     alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    %     alphaH =  (alpha_a + alpha_b)./alphaALL;
    %     alphaL = (alpha_c + alpha_d)./alphaALL;
    %     Z = alphaL./alphaH;
    %     qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    %     V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) -↙
delta2(i,:).*qr1);
    %
    %     zz = [GAME_SUMMARY(:,1) == 5 ] ;
    %     VV = VV + V.*zz;
    %     zzz = [GAME_SUMMARY(:,2) == 5 ] ;
    %     VVV = VVV + V.*zzz;
    %
    %     GAME_SUMMARY(:,4) = VV;
    %     GAME_SUMMARY(:,5) = VVV;


    % NOW NEED TO DEAL WITH THE OFFERS THAT MAY GET RECALLED

    %     What is done here is that a matrix called GSPr is introduced. The
    %     whole point is to construct the transition probabilities of moving
    %     from one stage to another given the offers (this will be given by
    %     GSPr(:,4:7)) and the returns from stage 1 and 2, GSPr(:,8:9) which will
    %     ultimately be fed back into GAME_SUMMARY(:,4:5). Note, GSPr(:,5) and
    %     GSPr(:,6) will give the endogenous discount factors for supplier 1
    %     and 2 respectively.

    GSPr = GAME_SUMMARY(:,1:3);

    reject1 = 1-alpha_d; reject2 = 1-alpha_c - alpha_d; reject3 = 1-alpha_a - alpha_d;
```

```
    reject4 = 1-alpha_a - alpha_d - alpha_c; reject5 = 0; reject6 = 1;

    z1 = [GSPr(:,1) == 1] ; z2 = [GSPr(:,1) == 2] ; z3 = [GSPr(:,1) == 3] ; z4 = [GSPr↙
(:,1) == 4] ;
    z5 = [GSPr(:,1) == 5] ; z6 = [GSPr(:,1) == 6] ; zz1 = [GSPr(:,2) == 1] ; zz2 = [GSPr↙
(:,2) == 2] ; zz3 = [GSPr(:,2) == 3] ;
    zz4 = [GSPr(:,2) == 4] ; zz5 = [GSPr(:,2) == 5] ;zz6 = [GSPr(:,2) == 6] ;

    GSPr(:,4) = z1.*reject1 + z2.*reject2 + z3.*reject3 + z4.*reject4 + z5.*reject5 +↙
z6.*reject6 ;
    GSPr(:,5) = zz1.*reject1 + zz2.*reject2 + zz3.*reject3 + zz4.*reject4 + zz5.*reject5↙
+ zz6.*reject6;

    zzz = [GSPr(:,3) == 1];
    GSPr(:,6) = zzz.*GSPr(:,5);
    zzz = [GSPr(:,3) == 2];
    GSPr(:,7) = zzz;
    GSPr(:,8) = 0;
    GSPr(:,9) = 0;


    % Offer 1 = lL

    alphaALL = alpha_a + alpha_b + alpha_c;
    alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
    alphaL = alpha_c./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    pr2 = t1_H(i,:) + delta2(i,:).*qr1 + t2_L.*fbq_c;

    z = [GSPr(:,1) == 1];
    zz = [GSPr(:,2) == 1];

    GSPr(:,8) = GSPr(:,9)+[a*fbq_c.^(b) - pr2.*GSPr(:,6) - (t1_L + t2_L.*fbq_c).*(1-GSPr↙
(:,6))].*z;
    GSPr(:,9) = GSPr(:,9)+[a*fbq_c.^(b) - pr2.*GSPr(:,7) - (t1_L + t2_L.*fbq_c).*(1-GSPr↙
(:,7))].*zz;

    %     The above 2 lines look a little different from the lemmas in the
    %     solution but follow from rearranging the IIR constraint of the lL
    %     type. It's just easier to code up this way as it fits easier with
    %     existing batches of code.


    % Offer 2 = lL hL

    qr1 = fbq_a(i,:);

    z = [GSPr(:,1) == 2];
    zz = [GSPr(:,2) == 2];
```

```
    GSPr(:,8) = GSPr(:,8)+ [a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPr(:,6) - (t1_H(i,:) +↙
t2_L.*fbq_c)].*z;
    GSPr(:,9) = GSPr(:,9)+[a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPr(:,7) - (t1_H(i,:) +↙
t2_L.*fbq_c)].*zz;


    % Offer 3 = lL lH

    alphaALL = alpha_c + alpha_b;
    alphaH = alpha_b/alphaALL;
    alphaL = alpha_c/alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
    qr2 = fbq_c;
    pr2 = pr1 - t2_L.*qr1 + t2_L.*qr2;
    qstar1 = [(t2_H(i,:) + (alpha_d/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-1));

    gamma = GSPr(:,6);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

    q2star1 = qr1.*gamma;
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b) -↙
p2star1).*alpha_a./(alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1 <↙
0].*(-100000000000000)) ;

    % The above line severely penalises if IIR(hL) is not violated.

    V = V + condition;

    z = [GSPr(:,1) == 3];
    GSPr(:,8) = GSPr(:,8)+[V].*z;

    %     Now we just do the same thing for supplier 2, but changing the
    %     discount rate
```

```
    gamma = GSPr(:,7);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

    q2star1 = qr1.*gamma;
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b) -↙
p2star1).*alpha_a./(alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1 <↙
0].*(-100000000000000)) ;

    V = V + condition;

    zz = [GSPr(:,2) == 3];
    GSPr(:,9) = GSPr(:,9)+ V.*zz;

    % Offer 4 = lL lH hL

    qr1 = fbq_b(i,:);
    pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
    %%%%%%%
    qstar1 = [(t2_H(i,:) + ((alpha_d + alpha_c)/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-↙
1));

    gamma = GSPr(:,6);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;

    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;

    p2star = [pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)].*gamma + [t1_H(i,:) + t2_L.*fbq_c];

    q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones↙
(size(gamma,1),1)],[],2);
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) +↙
```

```
(a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a)↙
+ (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    z = [GSPr(:,1) == 4];
    GSPr(:,8) = GSPr(:,8)+[V].*z;

    % Repeat for supplier 2

    gamma = GSPr(:,7);

    pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
    pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
    p2star = (pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)).*gamma + [t1_H(i,:) + t2_L.*fbq_c];

    q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).*ones↙
(size(gamma,1),1)],[],2);
    p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

    Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a) +↙
(a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
    V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d + alpha_a)↙
+ (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

    s = pstar > p2star;
    ss = pstar <= p2star;

    V = Vstar.*s + V2star.*ss;
    q1 = qstar1.*s + q2star1.*ss;

    z = [GSPr(:,2) == 4];
    GSPr(:,9) =GSPr(:,9)+ [V].*z;

    %     The block of code below is for looking inside all the code above and
    %     seeing how its bits are moving together. So just for debugging
    %     V = [V.*z Vstar V2star s ss];
    margin1 = a*qstar1.^(b) - pstar1;
    margin2 = a*q2star1.^(b) - p2star1;
    Offer4Tracking(i,1:8) = [pstar1(7,1) pstar(7,1) p2star(7,1) q2star1(7,1) p2star1↙
(7,1) Vstar(7,1) V2star(7,1) s(7,1)];
    Offer4Tracking(i,9:12) = [gamma(7,1) V(7,1) margin1(7,1) margin2(7,1) ];


    % Offer 5 = lL lH hL hH
```

```
    alphaALL = alpha_d + alpha_a + alpha_b + alpha_c;
    alphaH =  (alpha_a + alpha_b)./alphaALL;
    alphaL = (alpha_c + alpha_d)./alphaALL;
    Z = alphaL./alphaH;
    qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
    V = alphaH.*(a*qr1.^(b) - t1_H(i,:) - t2_H(i,:).*qr1) + alphaL.*(fbW_c(i,:) - delta2↙
(i,:).*qr1);

    zz = [GSPr(:,1) == 5 ] ;
    zzz = [GSPr(:,2) == 5 ] ;

    GSPr(:,8) =GSPr(:,8)+ [V].*zz;
    GSPr(:,9) =GSPr(:,9)+ [V].*zzz;

            % Offer 6 = none

    z = [GSPr(:,1) == 2];
    zz = [GSPr(:,2) == 2];

    GSPr(:,8) = GSPr(:,8)+ 0.*z;
    GSPr(:,9) = GSPr(:,9)+ 0.*zz;




    % In this section the code basically follows the construction of GSPr but
    % deals specifically with the contract sequences that give rise to the
    % possibility of mixing over whom to recall. The relevant sequences are
    % [1 1]
    % [2 2]
    % [2 4]
    % [3 3]
    % [4 2]
    % [4 4]
    % Once we work out the optimal mixing we plug it back into the relevant bit
    % in GSPr

    mixingsequences = [ [1 1]; [2 2]; [2 4]; [3 3]; [4 2];[4 4] ];
    mixsummaryrec = [];
    for mixline = 1:6
        GSPmix = [ones(1001,1)*mixingsequences(mixline,:) [linspace(0,1,1001)]'];

        % GSPmix(i,3) = 0 means supplier 1 gets recalled with certainty

        reject1 = 1-alpha_d; reject2 = 1-alpha_c - alpha_d; reject3 = 1-alpha_a -↙
alpha_d;
        reject4 = 1-alpha_a - alpha_d - alpha_c; reject5 = 0;

        z1 = [GSPmix(:,1) == 1] ; z2 = [GSPmix(:,1) == 2] ; z3 = [GSPmix(:,1) == 3] ; z4↙
= [GSPmix(:,1) == 4] ;
        z5 = [GSPmix(:,1) == 5] ; zz1 = [GSPmix(:,2) == 1] ; zz2 = [GSPmix(:,2) == 2] ;↙
```

```
zz3 = [GSPmix(:,2) == 3] ;
        zz4 = [GSPmix(:,2) == 4] ; zz5 = [GSPmix(:,2) == 5] ;

        GSPmix(:,4) = z1.*reject1 + z2.*reject2 + z3.*reject3 + z4.*reject4 + z5.↙
*reject5 ;
        GSPmix(:,5) = zz1.*reject1 + zz2.*reject2 + zz3.*reject3 + zz4.*reject4 + zz5.↙
*reject5 ;

        zzz = 1-GSPmix(:,3);
        GSPmix(:,6) = zzz.*GSPmix(:,5);
        zzz = GSPmix(:,3);
        GSPmix(:,7) = zzz;
        GSPmix(:,8) = 0;
        GSPmix(:,9) = 0;

        % Offer 1 = lL

        alphaALL = alpha_a + alpha_b + alpha_c;
        alphaH = alpha_a./alphaALL + alpha_b./alphaALL;
        alphaL = alpha_c./alphaALL;
        Z = alphaL./alphaH;
        qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
        pr2 = t1_H(i,:) + delta2(i,:).*qr1 + t2_L.*fbq_c;

        z = [GSPmix(:,1) == 1];
        zz = [GSPmix(:,2) == 1];

        GSPmix(:,8) = GSPmix(:,9)+[a*fbq_c.^(b) - pr2.*GSPmix(:,6) - (t1_L + t2_L.↙
*fbq_c).*(1-GSPmix(:,6))].*z;
        GSPmix(:,9) = GSPmix(:,9)+[a*fbq_c.^(b) - pr2.*GSPmix(:,7) - (t1_L + t2_L.↙
*fbq_c).*(1-GSPmix(:,7))].*zz;

        %     The above 2 lines look a little different from the lemmas in the
        %     solution but follow from rearranging the IIR constraint of the lL
        %     type. It's just easier to code up this way as it fits easier with
        %     existing batches of code.


        % Offer 2 = lL hL

        qr1 = fbq_a(i,:);

        z = [GSPmix(:,1) == 2];
        zz = [GSPmix(:,2) == 2];

        GSPmix(:,8) = GSPmix(:,8)+ [a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPmix(:,6) - (t1_H↙
(i,:) + t2_L.*fbq_c)].*z;
        GSPmix(:,9) = GSPmix(:,9)+[a*fbq_c.^(b) - delta2(i,:).*qr1.*GSPmix(:,7) - (t1_H↙
(i,:) + t2_L.*fbq_c)].*zz;
```

```
        % Offer 3 = lL lH

        alphaALL = alpha_c + alpha_b;
        alphaH = alpha_b/alphaALL;
        alphaL = alpha_c/alphaALL;
        Z = alphaL./alphaH;
        qr1 = [(t2_H(i,:) + Z.*delta2(i,:))./(a*b)].^(1/(b-1));
        pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
        qr2 = fbq_c;
        pr2 = pr1 - t2_L.*qr1 + t2_L.*qr2;
        qstar1 = [(t2_H(i,:) + (alpha_d/alpha_a).*delta2(i,:))./(a*b)].^(1/(b-1));

        gamma = GSPmix(:,6);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;

        p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

        q2star1 = qr1.*gamma;
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b)↙
- p2star1).*alpha_a./(alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1↙
< 0].*(-10000000000000000000000000000000000000000000000000000000000000000)) ;

        % The above line severely penalises if IIR(hL) is not violated.

        V = V + condition;

        z = [GSPmix(:,1) == 3];
        GSPmix(:,8) = GSPmix(:,8)+[V].*z;

        %     Now we just do the same thing for supplier 2, but changing the
        %     discount rate

        gamma = GSPmix(:,7);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1  - t2_L.*qstar1 + t2_L.*fbq_c;
```

```
        p2star = pr2.*gamma + [t1_L + t2_L.*fbq_c].*(1-gamma);

        q2star1 = qr1.*gamma;
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*alpha_d./(alpha_d + alpha_a) + (a*qstar1.^(b) -↙
pstar1).*alpha_a./(alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*alpha_d./(alpha_d + alpha_a) + (a*q2star1.^(b)↙
- p2star1).*alpha_a./(alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        condition = ([delta1(i,:).*(1-gamma) + delta2(i,:).*qr1.*gamma - delta2(i,:).*q1↙
< 0].*(-1000000000000000000000000000000000000000000000000000000000000000)) ;

        V = V + condition;

        zz = [GSPmix(:,2) == 3];
        GSPmix(:,9) =GSPmix(:,9)+ V.*zz;

        % Offer 4 = lL lH hL

        qr1 = fbq_b(i,:);
        pr1 = t1_H(i,:) + t2_H(i,:).*qr1;
        %%%%%%%
        qstar1 = [(t2_H(i,:) + ((alpha_d + alpha_c)/alpha_a).*delta2(i,:))./(a*b)].^(1/↙
(b-1));

        gamma = GSPmix(:,6);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;

        pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;

        p2star = [pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)].*gamma + [t1_H(i,:) + t2_L.↙
*fbq_c];

        q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).↙
*ones(size(gamma,1),1)],[],2);
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d +↙
alpha_a) + (a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d +↙
alpha_a) + (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

        s = pstar > p2star;
```

```
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        z = [GSPmix(:,1) == 4];
        GSPmix(:,8) = GSPmix(:,8)+[V].*z;

        % Repeat for supplier 2

        gamma = GSPmix(:,7);

        pstar1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*qstar1;
        pstar = pstar1 - t2_L.*qstar1 + t2_L.*fbq_c;
        p2star = (pr1 - t1_H(i,:) - t2_L.*fbq_a(i,:)).*gamma + [t1_H(i,:) + t2_L.↙
*fbq_c];

        q2star1 = min([qr1.*gamma + (delta1(i,:)/delta2(i,:)).*(1-gamma), fbq_a(i,:).↙
*ones(size(gamma,1),1)],[],2);
        p2star1 = delta1(i,:).*gamma + t1_L + t2_H(i,:).*q2star1;

        Vstar = (a*fbq_c.^(b) - pstar).*(alpha_c + alpha_d)./(alpha_c + alpha_d +↙
alpha_a) + (a*qstar1.^(b) - pstar1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);
        V2star = (a*fbq_c.^(b) - p2star).*(alpha_c + alpha_d)./(alpha_c + alpha_d +↙
alpha_a) + (a*q2star1.^(b) - p2star1).*(alpha_a )./(alpha_c + alpha_d + alpha_a);

        s = pstar > p2star;
        ss = pstar <= p2star;

        V = Vstar.*s + V2star.*ss;
        q1 = qstar1.*s + q2star1.*ss;

        z = [GSPmix(:,2) == 4];
        GSPmix(:,9) =GSPmix(:,9)+ [V].*z;

%         this is for debuigging
        z = find(z);
        Lookit = [GSPmix(z,3) V(z,:) s(z,:) ss(z,:) q1(z,:)];




        GAME_SUMMARY(:,4:5) = GSPr(:,8:9);
        GAME_SUMMARY(:,9) = GAME_SUMMARY(:,6);
        GAME_SUMMARY(:,8) = GSPr(:,5).*GAME_SUMMARY(:,6) + (1-GSPr(:,5)).*GAME_SUMMARY↙
(:,5);
        GAME_SUMMARY(:,7) = GSPr(:,4).*GAME_SUMMARY(:,8) + (1-GSPr(:,4)).*GAME_SUMMARY↙
(:,4);

        mixsummary = GSPmix(:,1:3);
        mixsummary(:,4:5) = GSPmix(:,8:9);
```

```
        s = find([GAME_SUMMARY(:,1) == GSPmix(1,1)].*[GAME_SUMMARY(:,2) == GSPmix↙
(1,2)]);
        mixsummary(:,6) = min(GAME_SUMMARY(s,6));
        mixsummary(:,9) = mixsummary(:,6);
        mixsummary(:,8) = GSPmix(:,5).*mixsummary(:,6) + (1-GSPmix(:,5)).*mixsummary(:,↙
5);
        mixsummary(:,7) = GSPmix(:,4).*mixsummary(:,8) + (1-GSPmix(:,4)).*mixsummary(:,↙
4);

        % mixsummary(:,7:9) are the expected payoffs, conditional at being at
        % each info set ie. Stage 1 2 & 3 resp.
        mixsummaryrec = [mixsummaryrec;mixsummary];

    end
    mixsummaryrec(:,10:17) = 0;
    mixsummaryrec(:,17) = 1;
    % NOW TO SOLVE THE GAME

    %     GS_CHECK = GAME_SUMMARY(:,4:5);
    GAME_SUMMARY(:,4:5) = GSPr(:,8:9);
    GAME_SUMMARY(:,9) = GAME_SUMMARY(:,6);
    GAME_SUMMARY(:,8) = GSPr(:,5).*GAME_SUMMARY(:,6) + (1-GSPr(:,5)).*GAME_SUMMARY(:,5);
    GAME_SUMMARY(:,7) = GSPr(:,4).*GAME_SUMMARY(:,8) + (1-GSPr(:,4)).*GAME_SUMMARY(:,4);
    nGS = find([GAME_SUMMARY(:,1) == 1].*[GAME_SUMMARY(:,2) == 1] + [GAME_SUMMARY(:,1)↙
== 2].*[GAME_SUMMARY(:,2) == 2] + [GAME_SUMMARY(:,1) == 2].*[GAME_SUMMARY(:,2) == 4] +↙
[GAME_SUMMARY(:,1) == 4].*[GAME_SUMMARY(:,2) == 2] + [GAME_SUMMARY(:,1) == 4].*↙
[GAME_SUMMARY(:,2) == 4]  + [GAME_SUMMARY(:,1) == 3].*[GAME_SUMMARY(:,2) == 3]);
    GAME_SUMMARY(nGS,:) = [];
    GAME_SUMMARY(:,13:17) = 0;

    Game_Summary = [GAME_SUMMARY;mixsummaryrec];

    % GAME_SUMMARY(:,7:9) are the expected payoffs, conditional at being at
    % each info set ie. Stage 1 2 & 3 resp.

    %     I'm just going to solve everything backwards this will tell me what
    %     is optimal at each info set/stage. GAME_SUMMARY(:,13:15) will be
    %     columns of ones and zeros with a one corresponding to a offer that is
    %     a best response at the info set.

    % note here i take advantage of how the first three columns are sorted
    Game_Summary(:,10:12) = Game_Summary(:,7:9);
    ContractsList = unique(Game_Summary(:,1:2),'rows');
    [nCL kCL] = size(ContractsList);
    for j = 1:nCL
        y = find([ContractsList(j,1) == Game_Summary(:,1)].*[ContractsList(j,2) ==↙
Game_Summary(:,2)]);
        x = max([Game_Summary(y,9)]);
        ny = size(y,1);
        x = x.*ones(ny,1);
        x = [x == Game_Summary(y,9)];
```

```
        Game_Summary(y,15) = x;
    end
    Game_Summary(:,11) = Game_Summary(:,8) + (1-Game_Summary(:,15)).*↙
(-10000000000000000000000000000);
    %     What the line above does is penalise any sequence of offers that would lead to↙
a
    %     subsequent offer violating sequential rationality

    % now need to think about what is best at stage 2, this gest tricky
    % when dealing with the sequences that invite mixing as they can'y be
    % compared to themselves as this would involve comparing different
    % equilibria
    ContractsList = unique(Game_Summary(:,1));
    [nCL kCL] = size(ContractsList);

    %HERE IS WHERE THE NEW CODE IS

    for j = 1:nCL

        if ContractsList(j,1) == 2
        olp2 = find([2 == Game_Summary(:,1)].*[Game_Summary(:,17) == 1].*[2 ==↙
Game_Summary(:,2)]);
        olp4 = find([2 == Game_Summary(:,1)].*[Game_Summary(:,17) == 1].*[4 ==↙
Game_Summary(:,2)]);
        test = [max(Game_Summary(olp2,11)) < min(Game_Summary(olp4,11))] ;
        if test == 1
            Game_Summary(olp2,17) = 0;
        end
        test =  [max(Game_Summary(olp4,11)) < min(Game_Summary(olp2,11))];
        if test == 1
            Game_Summary(olp4,17) = 0;
        end
        end


        ynm = find([ContractsList(j,1) == Game_Summary(:,1)].*[Game_Summary(:,17) ==↙
0]);
        xnm = max([Game_Summary(ynm,11)]);
        ym = find([ContractsList(j,1) == Game_Summary(:,1)].*[Game_Summary(:,17) == 1]);
        xmin = min([Game_Summary(ym,11)]);
        xmin10 = min([Game_Summary(ym,10)]);

        if xmin > xnm
            ymmm = find([xmin10 == Game_Summary(:,10)].*[ContractsList(j,1) ==↙
Game_Summary(:,1)].*[Game_Summary(:,17) == 1]);
            Game_Summary(ymmm,14) = 1;
        else
            y = find(ContractsList(j,1) == Game_Summary(:,1));
            ny = size(y,1);
            x = xnm.*ones(ny,1);
            x = [x <= Game_Summary(y,11)];
```

```
            xyz = Game_Summary(y,10) + (1-x).*↙
(1000000000000000000000000000000000000000000000000000000000000000000000000);

            xx = min(xyz);

            x = ones(ny,1) - [(xx.*ones(ny,1)) < (Game_Summary(y,10)+ (1-x).*↙
(100000000000000000000000000000000000000000000000000000000000000000000000000))];

            Game_Summary(y,14) = x;

        end
    end

    %HERE IS WHERE THE NEW CODE STOPS

    Game_Summary(:,10) = Game_Summary(:,7) + (1-Game_Summary(:,14)).*↙
(-1000000000000000000000000000);
    x = max(Game_Summary(:,10));
    nGS = size(Game_Summary,1);
    x = x.*ones(nGS,1);
    x = [x == Game_Summary(:,10)];
    Game_Summary(:,13) = x;


    % Now find the =m path: it'll be a row of ones in GAME_SUMMARY(:13:15).
    x = sum(Game_Summary(:,13:15),2);
    [n k] = find(x == 3);
    if isempty(n) == 1
        EXPECTED_BUYER_UTILITY(i,1) = -100000;
    else
        n = n(1);
        EXPECTED_BUYER_UTILITY(i,1) = Game_Summary(n,7);
        ContractRecorder(i,1:3) = Game_Summary(n,1:3);
        %         This next bit of code records the Expected Buyer Utility from the↙
following contracts
        %         (in this order): (1,2,1) (1,4,1) (1,1,1) (2,4,1) (3,3,1)
        CurveLook(i,1:5) = [Game_Summary(3,7) Game_Summary(7,7) Game_Summary(1,7)↙
Game_Summary(17,7) Game_Summary(25,7)];
        CurveLook(i,1:5) = max([CurveLook(i,1:5);zeros(1,5)]);

    end
end
RevenueRecall = EXPECTED_BUYER_UTILITY./2;
subplot(1,2,2)
plot(RevenueRecall,'b-')
subplot(1,2,1)
x = delta1;
plot(x,CurveLook(:,1),'b',x,CurveLook(:,2),'r+-',x,CurveLook(:,3),'k-',x,CurveLook(:,↙
4),'k:',x,CurveLook(:,5),'k-.')
legend('(1 2)','(1 4)','(1 1)','(2 4)','(3 3)')
```

```
cr = [ContractRecorder ContractRecorderNR 5.*ones(101,1)];
n = size(cr,1);
for i = 1:n-1
    if cr(n-i,:) == cr(n+1-i,:)
        cr(n+1-i,:) = [];
    end
end

return
```