

Technical Appendix: Computing solution of the game for different levels of inequality and graphical representation of the best response functions, Matlab code

Anna V. Yurko

1 Introduction

This document contains Matlab code for solving the three stage game for different levels of inequality, as measured by the Gini coefficient. The distribution function used is lognormal, specified by parameters μ and σ . The values of these parameters are derived using the Gini coefficient and the mean of the distribution. There are also four other programmes:

- Two programs that plot price best response functions of firms for any given quality levels for the number of firms equal to two (section 12) and the number of firms equal to three (section 13) and
- Two programs that plot quality best response functions of firms for the case of two firms (section 14) and for the case of three firms (section 15).

Section 2 contains the program. This program calls on several functions. The first function is `NE_qual_iter.m`. This function computes Nash equilibrium qualities for the given number of firms. Its text is given in section 3. The next function called upon is `NE_prices_iter.m`. It computes equilibrium prices taking as given firms' qualities. It is presented in section 4. Function `F.m` contains the lognormal distribution function (see section 11) and is used to compute firms' profits.

Function `NE_qual_iter.m` calls upon function `BR_qual.m`. Its input is the vector of qualities of all firms and the output is the quality best response of a particular firm. The code is presented in section 5. Depending on the index of the firm, function `BR_qual.m` uses function `ProfitQual_mid.m` (see section 6) or function `ProfitQual_last.m` (see section 7) to compute the profit as function of quality. Both of these functions use function `NE_prices_iter.m` to find optimal prices for these qualities.

Function `NE_prices_iter.m` calls on function `BR_price.m` (see section 8). This function takes as given prices of all firms and computes the price best response of a

particular firm. Function BR_price.m, depending on firm's index, uses function ProfitPrice_mid.m (see section 9) or function ProfitPrice_last.m (section 10) to compute firm's profit as function of price.

The programmes that graph price and quality best response functions to perform visual tests of the existence of equilibria also call upon all (for the quality best responses) or some (for the price best responses) of the functions listed above and described below.

2 Main code

```

global mu sig cc u_h

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% I. Parametrization

% Mean of the distribution:
m=15; % mean

%-----
% Set some qualities (derived in stage 2)
% The products are labeled in increasing order of quality: u_0<...<u_N,
% where N is the number of firms.

% Two cases: 1) Zero costs of quality improvement, 2) Quadratic fixed
% costs of quality improvement
% Case 1: Grid for qualities is (u_0,u_h].

% Normalize u_0=1, let u_h=10:
u_0=1;
u_h=10;

u_min=u_0+0.0001; % the lowest quality a firm can choose is slightly
% above u_0

% Case 2: Define the quality - cost function as cc*((u_k)^2).
% If cc=0 we have the zero costs case (case 1), and the upper bound u_h
% is active. Otherwise, it is not relevant.

cc=0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

```

% II. Load distribution values:

load sigma_gini_vecs.mat; % contains sig_vec (1 by 68) and gini_vec
% (same dimensions)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% III. Find solutions for different values of inequality

% Initialize solution matrices:
N_max=10;
NumFirms_gini=[gini_vec' zeros(length(sig_vec),2)];
QualFirms_gini=[gini_vec' zeros(length(sig_vec),N_max)];
PriceFirms_gini=[gini_vec' zeros(length(sig_vec),N_max)];
ProfitFirms_gini=[gini_vec' zeros(length(sig_vec),N_max)];

N=1; % initial number of firms
u_vec_0=[u_0 u_h];

for sig_ind=1:length(sig_vec)
    sig=sig_vec(sig_ind);
    mu=log(m)-((sig^2)/2);

    % Create initial matrices for qualities, prices, profits:
    Qual_mat=zeros(N_max,N_max+1);
    Pr_mat=zeros(N_max,N_max);
    Prof_mat=zeros(N_max,N_max);

    %-----
    prof_min=1; %to initialize the loop

    while prof_min>0
        % 1) Find optimal qualities:
        [uq_vec,I_conv]=NE_qual_iter(u_vec_0); % I_conv indicates
        % convergency. If it is not achieved for candidate solution,
        % check it graphically later on.
        % 2) Find optimal prices:
        N=length(uq_vec)-1;
        % Create C_k vector (the vector of relative quality
        % differences of firms:
        % C_k(k)=uq_vec(k)/(uq_vec(k)-uq_vec(k-1)):
        ut_dif=uq_vec(2:N+1)-uq_vec(1:N);
        C_k_int=(uq_vec(2:N+1))./ut_dif;
    end
end

```

```

        clear ut_dif;
price_vec=NE_prices_iter(uq_vec);
% 3) Find profits:
for j=1:N
    if j<N
        profit_vec(j)=(((price_vec(j+1))*(F((price_vec(j+1))*
            (1-C_k_int(j+1))+(price_vec((j+1)+1))*
            (C_k_int(j+1))))-F((price_vec((j-1)+1))*
            (1-C_k_int(j))+(price_vec(j+1))*(C_k_int(j)))))-
            cc*((uq_vec(j+1))^2));
    else
        profit_vec(j)=(((price_vec(j+1))*
            (1-F((price_vec((j-1)+1))*(1-C_k_int(j))+
            (price_vec(j+1))*(C_k_int(j)))))-
            cc*((uq_vec(j+1))^2));
    end
end
end
% 4) Is any firm making negative or zero profits?
prof_min=min(profit_vec);
% 5) Save results:
Qual_mat(N,1:N)=uq_vec(2:N+1);
Qual_mat(N,N_max+1)=I_conv;
Pr_mat(N,1:N)=price_vec(2:N+1);
Prof_mat(N,1:N)=profit_vec;
% 6) Re-initiate:
N=N+1;
u_vec_0=[uq_vec(1) (uq_vec(1)+0.0001) uq_vec(2:N)];
% 7) Clear variables:
clear uq_vec price_vec profit_vec C_k_int I_conv j;
end % for "while prof_min>0"
%-----

% Save results:
N=N-2;
NumFirms_gini(sig_ind,2)=N;
NumFirms_gini(sig_ind,3)=Qual_mat(N,N_max+1); % indicator of
% convergence
QualFirms_gini(sig_ind,2:N+1)=Qual_mat(N,1:N);
PriceFirms_gini(sig_ind,2:N+1)=Pr_mat(N,1:N);
ProfitFirms_gini(sig_ind,2:N+1)=Prof_mat(N,1:N);

% Set initial values for the next level of inequality:

```

```

N=1;
u_vec_0=[u_0 u_h];

clear prof_min Qual_mat Pr_mat Prof_mat
clear global sig mu;
global sig mu

end % for "for sig_ind=1:length(sig_vec)"

```

3 Function NE_qual_iter.m

```

function [D,Indic]=NE_qual_iter(u_vec0)

% Calculate NE qualities for N firms. Stage 2 of the game function.

global mu sig u_h cc ind_firm

u_vec_int=u_vec0;

N=length(u_vec_int)-1; %the first element is u_0

%-----
dist=1;
max_dist=0.0001;

iter=1;
maxiter=300;

Indic=0;

while dist>max_dist
    %iter
    u_vector(1)=u_vec_int(1); %the quality of the outside option is
    % always the same, u_0
    for j=1:N
        ind_firm=j;
        u_vector(ind_firm+1)=BR_qual(u_vec_int);
        clear global ind_firm;
        global ind_firm;
    end
end

```

```

    dist=max(abs(u_vector-u_vec_int));
    u_vec_int=u_vector;
    if iter>maxiter
        %disp('Convergence in qualities not achieved.');
```

Indic=1;

break

end

iter=iter+1;

clear u_vector;

end

D=u_vec_int;

4 Function NE_prices_iter.m

```

function D=NE_prices_iter(u_vec0)

% Calculate NE prices for N firms with given qualities.

global mu sig u_vec k

% Let u_vec=[u_0, u_1, ..., u_(N-1) u_N] and
% p_vec=[p_0, p_1, ..., p_(N-1), p_N].

N=length(u_vec0)-1;
u_vec=u_vec0;

% 1) Initialize:
dist=1;
max_dist=0.0001;

iter=1;
maxiter=500;

p_max=20;
price_vec_int=[0:((p_max)/N):p_max]; % initial price vector

% 2) Find equilibrium:
while dist>max_dist
    pr_vec(1)=price_vec_int(1); %the price of outside option is always
```

```

% the same, 0
for ind_k=1:N
    k=ind_k;
    pr_vec(k+1)=BR_price(price_vec_int);
    clear global k;
    global k;
end
dist=max(abs(pr_vec-price_vec_int));
price_vec_int=pr_vec;
if iter>maxiter
    %disp('Convergence in prices not achieved.');
```

```

    break
end
iter=iter+1;
end
D=pr_vec;
```

5 Function BR_qual.m

```

function D=BR_qual(u_vec0)

% Best Response function for any firm k=1,...,N in qualities

global mu sig ut_vec cc u_h ind_firm

options=optimset('MaxFunEvals',100000,'LargeScale','on','MaxIter',5000,
    'TolX',1e-5,'Display','off'); % for nonlinear cases

% Let u_vec=[u_0, u_1, ..., u_(N-1) u_N] and
% p_vec=[p_0, p_1, ..., p_(N-1), p_N].

ut_vec=u_vec0;
N=length(ut_vec)-1;

if ind_firm<N
    q=fminbnd(@ProfitQual_mid,(ut_vec(ind_firm)),(ut_vec(ind_firm+2)),
        options);
    Prof_q=ProfitQual_mid(q);
    if Prof_q>0
```

```

        D=ut_vec(ind_firm)+0.0001;
    else
        D=q;
    end
else
    if cc>0
        q=fminbnd(@ProfitQual_last,(ut_vec(ind_firm)),
            (50*(ut_vec(ind_firm+1))),options);
        Prof_q=ProfitQual_last(q);
        if Prof_q>0
            D=ut_vec(ind_firm)+0.0001;
        else
            D=q;
        end
    else
        D=u_h;
    end
end
end

```

6 Function ProfitQual_mid.m

```

function D=ProfitQual_mid(u_int);

% Specify profit as a function of quality for firms k=1,...,N-1, so
% that prices are NE.

global mu sig cc ut_vec ind_firm

u_vec0=ut_vec;

% 1) u_vec0 has all the firms, including ind_firm. Replace element with
% u_int
u_vec0(ind_firm+1)=u_int;

% 2) Write C_k:
N=length(u_vec0)-1;
% Create C_k vector:
ut_dif=u_vec0(2:N+1)-u_vec0(1:N);
C_k=(u_vec0(2:N+1))./ut_dif;
clear ut_dif;

```

```

% 2) Find optimal prices with these qualities
pr_vec=NE_prices_iter(u_vec0);

% 4) Write profit (minus):
D=-(((pr_vec(ind_firm+1))*(F((pr_vec(ind_firm+1))*(1-C_k(ind_firm+1))+
    (pr_vec((ind_firm+1)+1))*(C_k(ind_firm+1))))-
    F((pr_vec((ind_firm-1)+1))*(1-C_k(ind_firm))+(pr_vec(ind_firm+1))*
    (C_k(ind_firm))))))-cc*(u_int)^2);

```

7 Function ProfitQual_last.m

```

function D=ProfitQual_last(u_int);

% Specify profit as a function of quality for firm k=N, so that
% prices are NE.

global mu sig cc ut_vec ind_firm

u_vec0=ut_vec;

% 1) u_vec0 has all the firms, including k. Replace element with u_int
u_vec0(ind_firm+1)=u_int;

% 2) Write C_k:
N=length(u_vec0)-1;
% Create C_k vector:
ut_dif=u_vec0(2:N+1)-u_vec0(1:N);
C_k=(u_vec0(2:N+1))./ut_dif;
clear ut_dif;

% 2) Find optimal prices with these qualities
pr_vec=NE_prices_iter(u_vec0);

% 4) Write profit (minus):
D=-(((pr_vec(ind_firm+1))*(1-F((pr_vec((ind_firm-1)+1))*(1-C_k(ind_firm))
    +(pr_vec(ind_firm+1))*(C_k(ind_firm))))))-cc*(u_int)^2);

```

8 Function BR_price.m

```
function D=BR_price(p_vec0)

% Best Response function for any firm k=1,...,N

global u_vec mu sig p_vec k

options=optimset('MaxFunEvals',100000,'LargeScale','on','MaxIter',5000,
    'TolX',1e-4,'Display','off'); % for nonlinear cases

% Let u_vec=[u_0, u_1, ..., u_(N-1) u_N] and
% p_vec=[p_0, p_1, ..., p_(N-1), p_N].

p_vec=p_vec0;
N=length(u_vec)-1;

if k<N
    D=fminsearch(@ProfitPrice_mid,p_vec0(k+1));
else
    D=fminsearch(@ProfitPrice_last,p_vec0(k+1));
end
```

9 Function ProfitPrice_mid.m

```
function D=ProfitPrice_mid(x0)

% Compute profit as function of own price for any firm k=1,...,N-1

% Let u_vec=[u_0, u_1, ..., u_(N-1) u_N] and
% p_vec=[p_0, p_1, ..., p_(N-1), p_N]. Here p_k is
% a choice variable

% In this price function we take into account the option of firm k to
% undercut firm (k-1), that is, set low price p_k, such that
%  $t_k \leq t_{(k-1)}$ . Then, firm k competes with firm (k-2) more directly.

global mu sig u_vec p_vec k
```

```

if k<2
    D=-((x0)*(F((u_vec((k+1)+1))*(p_vec((k+1)+1))-(u_vec(k+1))*
        (x0))/(u_vec((k+1)+1)-(u_vec(k+1))))-F((u_vec(k+1))*
        (x0)-(u_vec((k-1)+1))*(p_vec((k-1)+1)))/(u_vec(k+1))-
        (u_vec((k-1)+1)))));
else
    % Can undercut firm (k-1) by setting a price below:
    p_cutoff=(((u_vec((k-1)+1))*(p_vec((k-1)+1))-(u_vec((k-2)+1))
        *(p_vec((k-2)+1)))/(u_vec((k-1)+1)-(u_vec((k-2)+1))))*
        ((u_vec(k+1)-(u_vec((k-1)+1)))+(u_vec((k-1)+1))*
        (p_vec((k-1)+1)))/(u_vec(k+1)));
    if x0>p_cutoff
        D=-((x0)*(F((u_vec((k+1)+1))*(p_vec((k+1)+1))-(u_vec(k+1))
            *(x0))/(u_vec((k+1)+1)-(u_vec(k+1))))-F((u_vec(k+1))*
            (x0)-(u_vec((k-1)+1))*(p_vec((k-1)+1)))/(u_vec(k+1))-
            (u_vec((k-1)+1)))));
    else
        D=-((x0)*(F((u_vec((k+1)+1))*(p_vec((k+1)+1))-(u_vec(k+1))*
            (x0))/(u_vec((k+1)+1)-(u_vec(k+1))))-F((u_vec(k+1))*
            (x0)-(u_vec((k-2)+1))*(p_vec((k-2)+1)))/(u_vec(k+1))-
            (u_vec((k-2)+1)))));
    end
end
end

```

10 Function ProfitPrice_last.m

```

function D=ProfitPrice_last(x0)

% Compute profit as function of own price for top firm k=N

% Let u_vec=[u_0, u_1, ..., u_(N-1) u_N] and
% p_vec=[p_0, p_1, ..., p_(N-1), p_N]. Here p_N is
% a choice variable

global mu sig u_vec p_vec k

% Here k=N

if k<2
    D=-((x0)*(1-F((u_vec(k+1))*(x0)-(u_vec((k-1)+1))*

```

```

        (p_vec((k-1)+1))/((u_vec(k+1))-(u_vec((k-1)+1)))));
else
    % Can undercut firm (k-1) by setting a price below:
    p_cutoff=(((u_vec((k-1)+1))*(p_vec((k-1)+1))-(u_vec((k-2)+1))*
        (p_vec((k-2)+1)))/((u_vec((k-1)+1))-(u_vec((k-2)+1))))*
        ((u_vec(k+1))-(u_vec((k-1)+1)))+(u_vec((k-1)+1))*
        (p_vec((k-1)+1)))/(u_vec(k+1));
    if x0>p_cutoff
        D=-((x0)*(1-F((u_vec(k+1))*(x0)-(u_vec((k-1)+1))*
            (p_vec((k-1)+1)))/((u_vec(k+1))-(u_vec((k-1)+1))))));
    else
        D=-((x0)*(1-F((u_vec(k+1))*(x0)-(u_vec((k-2)+1))*
            (p_vec((k-2)+1)))/((u_vec(k+1))-(u_vec((k-2)+1))))));
    end
end
end

```

11 Function F.m: Income distribution

```

function D=F(v)

% This is cdf

global mu sig

D=logncdf(v,mu,sig);

```

12 Graphing price best response functions for the case of 2 firms and given qualities

```

global mu sig u_vec k

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PARAMETRIZATION

% Mean of the distribution:
m=15;

```

```

%-----
% Set some qualities (derived in stage 2)
% Grid for qualities is (u_0,u_h].

u_0=1;
u_h=10;

u_min=u_0+0.0001; % the lower bound is slightly above u_0

%-----
% Define the quality - cost function as cc*((u_k)^2). cc=0 means zero
% costs, and the upper bound u_h is active. It should not be otherwise.

cc=0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load sigma_gini_vecs.mat; % contains sig_vec (1 by 68) and gini_vec
% (same dimensions)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pick parameters of the distribution:
sig_ind=1; % index 1 through 68
sig=sig_vec(1);
mu=log(m)-((sig^2)/2);

%-----
N=2; % number of firms
L_vec=300; % size of the grid

% Set any qualities:
u_vec=[u_0 2 9];

% 1. Find the best response function p1(p2):

k=1;

p_ub=10;
p_lb=0.0001;

p2_grid=[p_lb:((p_ub-p_lb)/(L_vec-1)):p_ub];

```

```

for ind_p2=1:length(p2_grid)
    p_vec_int=[0 (mean([0 (p2_grid(ind_p2))])) (p2_grid(ind_p2))];
    BR_p1(ind_p2)=BR_price(p_vec_int);
    clear p_vec_int;
end
clear ind_p2;
clear global k
global k

% 2. Find the best response function p2(p1):

k=2;

p1_grid=[p_lb:((p_ub-p_lb)/(L_vec-1)):p_ub];

for ind_p1=1:length(p1_grid)
    p_vec_int=[0 (p1_grid(ind_p1)) ((p1_grid(ind_p1))+1)];
    BR_p2(ind_p1)=BR_price(p_vec_int);
    clear p_vec_int;
end
clear ind_u1 p_lb p_ub L_vec;

% 3. Plot these functions:
plot(p1_grid,BR_p2,'r','LineWidth',2);
hold on;
plot(BR_p1,p2_grid,'-.','LineWidth',2);

```

13 Graphing price best response functions for the case of 3 firms and given qualities

```

global mu sig u_vec k

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PARAMETRIZATION

```

```

% Mean of the distribution:
m=15;

```

```

%-----
% Set some qualities (derived in stage 2)
% Grid for qualities is (u_0,u_h].

u_0=1;
u_h=10;

u_min=u_0+0.0001; % the lower bound is slightly above u_0

%-----
% Define the quality - cost function as cc*((u_k)^2). cc=0 means zero
% costs, and the upper bound u_h is active. It should not be otherwise.

cc=0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load sigma_gini_vecs.mat; % contains sig_vec (1 by 68) and gini_vec
% (same dimensions)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pick parameters of the distribution:
sig_ind=60;
sig=sig_vec(1);
mu=log(m)-((sig^2)/2);

%-----
N=3; % number of firms
L_vec=100; % size of the grid

% Set any qualities:
u_vec=[u_0 2 4 9];

max_dist=0.0001;
max_iter=200;

% 1. Find the best response function p1(p2(p1,p3),p3):
p_ub=10;
p_lb=0.0001;
p3_grid=[p_lb:((p_ub-p_lb)/(L_vec-1)):p_ub];

for ind_p3=1:length(p3_grid)

```

```

p_vec_int0=[0 ((1/3)*(p3_grid(ind_p3))) ((2/3)*(p3_grid(ind_p3)))
(p3_grid(ind_p3))];
dist=1;
iter=0;
p_vec_int=p_vec_int0;
while dist>max_dist
    for j=1:2
        k=j;
        p_vec_int(k+1)=BR_price(p_vec_int0);
        clear global k;
        global k;
    end
    dist=max(abs(p_vec_int-p_vec_int0));
    p_vec_int0=p_vec_int;
    if iter>max_iter
        %disp('p3=');
        %disp((p3_grid(ind_p3)));
        %disp('Convergence for p1 and p2 not achieved.')
```

```

        break
    end
    iter=iter+1;
end % for "while dist>max_dist"
BR_p1(ind_p3)=p_vec_int(2);
clear p_vec_int p_vec_int0 dist iter j;
end
clear ind_p3;

% 2. Find the best response function p3(p1,p2(p1,p3)):
p1_grid=[p_lb:((p_ub-p_lb)/(L_vec-1)):p_ub];

for ind_p1=1:length(p1_grid)
    p_vec_int0=[0 (p1_grid(ind_p1)) ((p1_grid(ind_p1))+1)
((p1_grid(ind_p1))+2)];
    dist=1;
    iter=0;
    p_vec_int=p_vec_int0;
    while dist>max_dist
        for j=1:2
            k=j+1; % firms 2 and 3!
            p_vec_int(k+1)=BR_price(p_vec_int0);
            clear global k;
            global k;
        end
    end
end

```

```

end
dist=max(abs(p_vec_int-p_vec_int0));
p_vec_int0=p_vec_int;
if iter>max_iter
    %disp('p1=');
    %disp((p1_grid(ind_p1)));
    %disp('Convergence for p2 and p3 not achieved.');
```

break

```

end
iter=iter+1;
end % for "while dist>max_dist"
BR_p3(ind_p1)=p_vec_int(4);
clear p_vec_int0 p_vec_int dist iter j;
end
clear ind_p1 p_lb p_ub L_vec;

% 3. Plot these functions:
plot(p1_grid,BR_p3,'r','LineWidth',2);
hold on;
plot(BR_p1,p3_grid,'-.','LineWidth',2);
```

14 Graphing quality best response functions for the case of 2 firms

```

% Find and plot best response functions in qualities for two firms
% u1(u2) and u2(u1). Check the intersections - NE.
```

```

global mu sig cc u_h ind_firm
```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PARAMETRIZATION
```

```

% Mean of the distribution:
m=15;
```

```

%-----
% Set some qualities (derived in stage 2)
% Grid for qualities is (u_0,u_h).
```

```

u_0=1;
u_h=10;

u_min=u_0+0.0001; % the lower bound is slightly above u_0

%-----
% Define the quality - cost function as cc*((u_k)^2). cc=0 means zero
% costs, and the upper bound u_h is active. It should not be otherwise.

cc=0.01;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load sigma_gini_vecs.mat; % contains sig_vec (1 by 68) and gini_vec
% (same dimensions)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Pick parameters of the distribution:
sig=sig_vec(1);
mu=log(m)-((sig^2)/2);

%-----
N=2;
L_vec=100;

% 1. Find the best response function u1(u2):
ind_firm=1;
u_ub=12;
u_lb=(u_0+0.0002);
u2_grid=[u_lb:((u_ub-u_lb)/(L_vec-1)):u_ub];

for ind_u2=1:length(u2_grid)
    u_vec=[u_0 u_min (u2_grid(ind_u2))];
    BR_u1(ind_u2)=BR_qual(u_vec);
    clear u_vec;
end
clear ind_u2 u_lb;
clear global ind_firm
global ind_firm

% 2. Find the best response function u2(u1):
ind_firm=2;
u_lb=u_min;

```

```

u1_grid=[u_lb:((u_ub-u_lb)/(L_vec-1)):u_ub];

for ind_u1=1:length(u1_grid)
    u_vec=[u_0 (u1_grid(ind_u1)) ((u1_grid(ind_u1))+0.0002)];
    BR_u2(ind_u1)=BR_qual(u_vec);
    clear u_vec;
end
clear ind_u1 u_lb u_ub L_vec;

% 3. Plot these functions:
plot(u1_grid,BR_u2,'r','LineWidth',2);
hold on;
plot(BR_u1,u2_grid,'-.','LineWidth',2);

```

15 Graphing quality best response functions for the case of 3 firms

```

global mu sig cc u_h ind_firm

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PARAMETRIZATION

% Mean of the distribution:
m=15;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set some qualities (derived in stage 2)
% Grid for qualities is (u_0,u_h].

u_0=1;
u_h=10;

u_min=u_0+0.0001; % the lower bound is slightly above u_0

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Define the quality - cost function as cc*((u_k)^2). cc=0 means zero
% costs, and the upper bound u_h is active. It should not be otherwise.

cc=0.01;

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load sigma_gini_vecs.mat; % contains sig_vec (1 by 68) and gini_vec
% (same dimensions)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pick parameters of the distribution:
sig=sig_vec(6);
mu=log(m)-((sig^2)/2);

%-----
N=3;
L_vec=50;

max_dist=0.0001;
max_iter=200;

% 1. Find the best response function u1(u2(u1,u3),u3):
u_ub=12;
u_lb=(u_0+0.0003);
u3_grid=[u_lb:((u_ub-u_lb)/(L_vec-1)):u_ub];

for ind_u3=1:length(u3_grid)
    u_vec0=[u_0 u_min (u_min+0.0001) (u3_grid(ind_u3))];
    dist=1;
    iter=0;
    u_vec=u_vec0;
    while dist>max_dist
        for j=1:2
            ind_firm=j;
            u_vec(ind_firm+1)=BR_qual(u_vec0);
            clear global ind_firm;
            global ind_firm;
        end
        dist=max(abs(u_vec-u_vec0));
        u_vec0=u_vec;
        if iter>max_iter
            %disp('u3=');
            %disp((u3_grid(ind_u3)));
            %disp('Convergence for u1 and u2 not achieved.');
```

```

        end
        iter=iter+1;
    end % for "while dist>max_dist"
    BR_u1(ind_u3)=u_vec(2);
    clear u_vec u_vec0 dist iter j;
end
clear ind_u3 u_lb;

% 2. Find the best response function u3(u1,u2(u1,u3)):
u_lb=u_min;
u_ub=4;
L_vec=16;
u1_grid=[u_lb:((u_ub-u_lb)/(L_vec-1)):u_ub];

for ind_u1=1:length(u1_grid)
    u_vec0=[u_0 (u1_grid(ind_u1)) ((u1_grid(ind_u1))+0.0001)
            ((u1_grid(ind_u1))+0.0002)];
    dist=1;
    iter=0;
    u_vec=u_vec0;
    while dist>max_dist
        for j=1:2
            ind_firm=j+1; % firms 2 and 3!
            u_vec(ind_firm+1)=BR_qual(u_vec0);
            clear global ind_firm;
            global ind_firm;
        end
        dist=max(abs(u_vec-u_vec0));
        u_vec0=u_vec;
        if iter>max_iter
            %disp('u1=');
            %disp((u1_grid(ind_u1)));
            %disp('Convergence for u2 and u3 not achieved.');
```

```

            break
        end
        iter=iter+1;
    end % for "while dist>max_dist"
    BR_u3(ind_u1)=u_vec(4);
    clear u_vec u_vec0 dist iter j;
end
clear ind_u1 u_lb u_ub L_vec;

```

```
% 3. Plot these functions:  
plot(u1_grid,BR_u3,'r','LineWidth',2);  
hold on;  
plot(BR_u1,u3_grid,'-.','LineWidth',2);
```