Advanced Econometrics in Labor and IO
Week 3 - Static Discrete Choice in IO

Hannes Ullrich
DIW Berlin and University of Copenhagen

DIW Office 5.2.020, email: hullrich@diw.de

12 May 2022, DIW

Organisation: next session

- Next sessions:
  - May 19, 14:15 - 17:00.
  - June 2, 14:15 - 17:00.
- Single-Agent Dynamic Discrete Choice
  - Read *Rust (Ecta 1987, HoEx 1994)*

Organisation: next session

- Next sessions:
  - May 19, 14:15 - 17:00.
  - June 2, 14:15 - 17:00.
- Single-Agent Dynamic Discrete Choice
  - Read *Rust (Ecta 1987, HoEx 1994)*
- Hand in problem set 3 before May 26

Organisation: next session

- Next sessions:
  - May 19, 14:15 - 17:00.
  - June 2, 14:15 - 17:00.
- Single-Agent Dynamic Discrete Choice
  - Read *Rust (Ecta 1987, HoEx 1994)*
- Hand in problem set 3 before May 26
- Related advertisement: special IO BB seminar with Reinhold Kesler (U Zurich) GDPR And The Lost Generation Of Innovative Apps

Organisation: next session

- Next sessions:
  - May 19, 14:15 - 17:00.
  - June 2, 14:15 - 17:00.
- Single-Agent Dynamic Discrete Choice
  - Read *Rust (Ecta 1987, HoEx 1994)*
- Hand in problem set 3 before May 26
- Related advertisement: special IO BB seminar with Reinhold Kesler (U Zurich) GDPR And The Lost Generation Of Innovative Apps
- Problem set 4 (dynamic discrete choice) will be graded, due June 2, 14:15.

Plan for today

- Recap BLP: The Random Coefficient Logit Model of Demand
- Work through BLP code
- Discuss along the way
  - Numerical integration
  - Contraction mapping
  - Supply side moments

Recap BLP

- Individual choice model using market-level data with
  - horizontal (ϵ, μ) and vertical (δ, ξ) product differentiation
  - (price) endogeneity
    - Isolate mean utility to estimate β coefficients by linear IV estimation
  - unobserved heterogeneity
  - flexible substitution patterns
    - In homogenous logit, only market shares matter
    - In heterogenous logit, closeness in characteristics space
  - static pricing game for improved identification
  - quasilinear preferences allow computing changes in consumer welfare

Recap BLP

- Identification
  - exogenous variation in observed characteristics, price
  - choice set variation
  - exogenous variation in preferences leading to variation in choice probabilities / market shares
  - formal positive nonparametric identification results by Berry and Haile (Ecta 2014, ARE 2016), Fox and Ghandi (Rand 2016), Fox, Kim, Ryan, and Bajari (JoE 2012)
    $\rightarrow$ functional forms and distributional assumption not necessary for identification, standard IV conditions are sufficient

BLP widely used but problems

- Google Scholar citations: 6665 (BLP 1995), 17755 (Train 2009)
- Econometric
  - Identifying unobserved heterogeneity with aggregate data,
    Petrin (JPE 2002), BLP (JPE 2004) add microdata
  - Weak instrumental variables, lack of cost shifters -
    Armstrong (Ecta 2016), Reynaert and Verboven (JoE 2014), Ghandi and Houde (2019)
  - "Logit" assumption / Welfare analysis with many products, entry/exit,
    Ackerberg and Rysman (Rand 2005), Berry and Pakes (IER 2007)
  - Measurement error in market shares (small T, large J), moment inequalities, Ghandi, Lu, and Shi (2020)
  - Asymptotics, small sample behavior -
    Freyberger (JoE 2015), Skrainka (2011)

BLP widely used but problems

- Numerical
  - Error tolerance in inner loop δ and premature 'convergence', Dubé, Fox, and Su (Ecta 2012), Knittel and Metaxoglu (ReStat 2012)
  - Reynaerts, Varadhan, and Nash (2012): fast and robust solving for δ, Li (2018): restate δ as solution to a convex optimization problem
  - Quality of Matlab solvers not state-of-the-art, Dubé, Fox, and Su (Ecta 2012), Knittel and Metaxoglu (ReStats 2014)
  - Importance of starting values, Knittel and Metaxoglu (ReStats 2014)
  - Numerical integration techniques and simulation error, Judd and Skrainka (2011), Chiou and Walker (JoE 2007)
- Implications for elasticity estimates and, in consequence, for measure of market power, merger evaluation, welfare gains from new products/technologies, ...?
- Progress: Conlon and Gortmaker (2020) best practice paper incl. python code: https://github.com/jeffgortmaker/pyblp

BLP Code

Nested fixed point algorithm (BLP 1995)

Matlab functions to be called (directly and indirectly) from main script

- Market shares: $s_{jt}(\delta_t, \sigma) \approx \sum_{i=1}^{n} \phi_i \frac{\exp(\delta_{jt}+\mu_{jt}(\nu_i))}{1+\sum_{l=1}^{J} \exp(\delta_{lt}+\mu_{lt}(\nu_i))}$
- Market share derivatives Δ_t:

$$\sum_{i=1}^{n} \phi_i \left[-\alpha s_{ijt}(1 - s_{ijt})\right] \ \forall\ j = k, \qquad \sum_{i=1}^{n} \phi_i \left[\alpha s_{ijt} s_{ikt}\right] \ \forall\ j \neq k$$

- Equilibrium prices, FOC: $0 = c_t - p_t - \Delta_t^{-1} s_t$
- Contraction mapping: $\delta^{h+1} = \delta^h + ln(S) - ln(s(\delta^h, \sigma))$

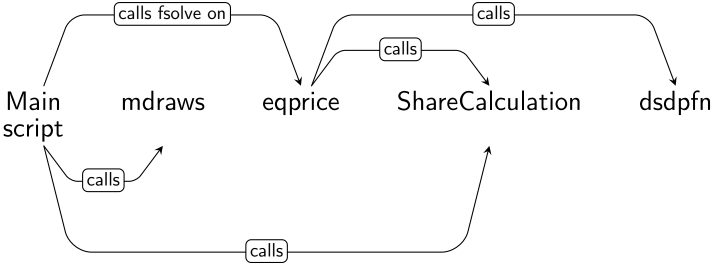
Nested fixed point algorithm (BLP 1995)

- GMM objective function,
  minimization problem based on moment conditions $E[g_{jt}(z_{jt})\xi_{jt}] = 0$:

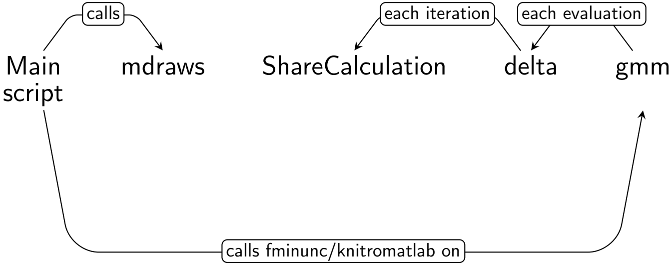
$$\min_{\theta} \xi(\theta)' g(z)' A g(z) \xi(\theta)$$

- For simplicity, no analytical gradients. In practice, use if possible!

Construct data



BLP estimation - only demand side moments



Main script: Creating fake data

```
% Size/index of dataset
% number of markets
nmkt = 25;
% number of brands per market
nbrn = 10;
% number of observations
nobs=nmkt*nbrn;
% market number for each obs
cdid = kron((1:nmkt)',ones(nbrn,1));
% vector with last obs per market
cdindex = (nbrn:nbrn:nbrn*nmkt)';
% dummies for each market
dummarket=dummyvar(cdid);

% Single-product firm ownership matrix
owner=repmat(diag(ones(nbrn,1)),nmkt,1);
```

Specify basic dimensions of data set: $T = 25$, $J = 10$
Ownership matrix O

Main script

```
% True model parameter values
% mean tastes on constant, x, p
betatrue = [2 2 -2]';
% random coefficient standard error
rc_true = 1;
% parameters of the model
thetatrue = [betatrue;rc_true];

% Number of linear parameters
nlin=length(betatrue);
% Number of instruments
ninst = 3;

% Price Equation Parameters
zparamtrue=[ones(ninst,1);0.7*ones(nlin-1,1)];

% Degree of endogeneity
% (omega, ksi covariance)
truecovomegksi=[1 0.7; 0.7 1];
mu=zeros(nobs,2);
```

Set true parameters: $\beta, \alpha, \sigma, \gamma, cov(\omega, \xi)$

Notes on numerical integration

Stochastic / Monte Carlo

- "Monte Carlo is the art of approximating an expectation by the sample mean of a function of simulated random variables" - Statistical Genetics lecture notes, UC Berkeley.
- Pseudo-random - standard random number generator
- Quasi-random - more uniform coverage, e.g. Halton, modified Latin hypercube sampling (Hess, Train, and Polak, TR Part B 2006)
- Importance sampling - higher weights to "important" draws
- Simulation bias in MSL and MSS (ln in simulated $\ln P_n(\theta)$ is a nonlinear transformation), not in MSM

Notes on numerical integration

Non-stochastic / Quadrature

- Gaussian Hermite product rule
  - Difficult for high-dimensional distributions and complicated integrands
- Sparse grid integration (Heiss and Winschel, JoE 2008)
  - Subset of nodes from product rule

Pseudo-/quasi-random draws

```
function[v, quadweight]=mdraws(nishares,Ktheta,ndr)
rng(0);
quadweight=(1/(ndr*Ktheta))*ones(ndr*Ktheta,1);
if nishares == 1
    % Pseudo-random draws
    v = randn(Ktheta,ndr);
elseif nishares == 2
    % Modified Latin Hypercube Sampling
    shift = rand(1,1)/ndr;
    if Ktheta==1
        p = (0:ndr-1)./ndr + shift;
        v = norminv(p,0,1);
    else
        v = zeros(Ktheta,ndr);
        for k=1:Ktheta
            % unidimensional draws
            draws = (0:ndr-1)./ndr + shift;
            % Shuffle unidimensional draws, append
            [~,rrid] = sort(rand(ndr,1));
            v(k,:) = norminv(draws(rrid'),0,1);
        end
    end
```

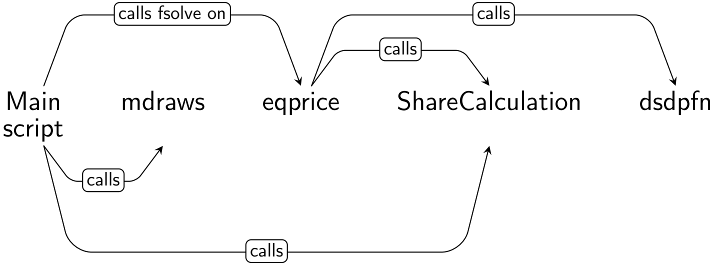
Function drawing random numbers: pseudo random number generator or MLHS.

Quasi-random draws / Quadrature

```
elseif nishares == 3
    % Scrambled Halton draws
    p = net(...
        scramble(...
        haltonset(Ktheta,'Skip',1e3,'Leap',1e2),...
        'RR2'),...
        ndr)';
    v = norminv(p,0,1);
elseif nishares == 4
    % Sparse Grid Integration, Kronrod-Patterson rule
    [v,quadweight]=nwspgr('KPN', Ktheta, ndr);
end
```

Scrambled Halton draws or Sparse Grid Nodes.

Construct data



Main script

```
% Individual unobserved heterogeneity
% Integration of market shares using
% pseudo Monte Carlo (1)
% MLHS (2)
% Scrambled Halton (3)
% Sparse Grid Integration (4)
drawsintegration=4;
% MC: Draws, Quadrature: Precision (KPN: 7)
ndraws = 7;

% Draw v or choose nodes/weights
[qv, qweight] = mdraws(drawsintegration,1,ndraws);

nodes=length(qweight);
% make row vector and duplicate nobs times
qv=ones(nobs,1)*qv';
qweight=ones(nobs,1)*qweight';
```

Draw random numbers / fix nodes for market share integral.

Main script

```
% Set Seed
rng(1)

% Unobserved characteristics: omeg and Ksi
omegksi=mvnrnd(mu,truecovomegksi);
ksi=omegksi(:,1);
omeg=omegksi(:,2);

% Constant and One Product Attribute, U(1,2)
A = 1+rand(nobs,nlin-2);
A = [ones(nobs,1) A];

% Cost Shifters, U(0,1)
z =  rand(nobs,ninst);
```

Draw data:
ξ, ω, x, w.

Main script

```
% X-Matrices for Estimation - part I
Xrandom=A(:,2); % Random coefficient X vector
% Compute individual specific contribution x*mu
xv=(Xrandom*ones(1,nodes)).*qv;
```

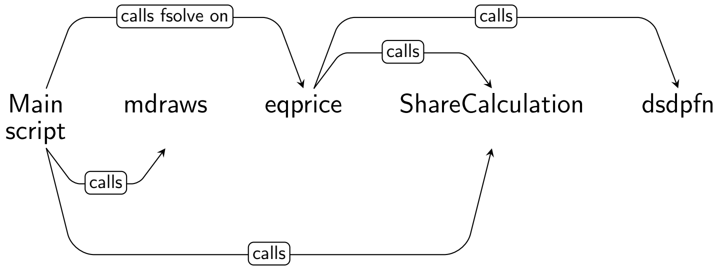
Arrange nonlinear part of utility function: $x_{jt}\nu_i$

Main script

```
% Price with perfect competition
% a function of z, A and omeg = mc
price = [z A]*zparamtrue + omeg;

% Price with imperfect competition
% Compute Nash equilibrium
PData.nmkt=nmkt;PData.nbrn=nbrn;PData.cdid=cdid;
PData.dummarket=dummarket;PData.owner=owner;
PData.betatrue=betatrue;PData.rc_true=rc_true;
PData.nodes=nodes;PData.qweight=qweight;
PData.xv=xv;PData.A=A;PData.ksi=ksi;PData.mc=price;

options=optimset('Display','iter',...
    'TolFun',1e-6,'TolX',1e-6);
aneqprice = @(price)eqprice(price,PData);
[eprice,fval,exitflag] = ...
    fsolve(aneqprice,price,options);
```

Compute equilibrium prices.

Construct data



Supply side equilibrium function

```
% This function computes
% Bertrand-Nash equilibrium prices,
% given a price starting vector and data
function root = eqprice(price,data)

betatrue=data.betatrue;rc_true=data.rc_true;
A=data.A;ksi=data.ksi;owner=data.owner;
mc=data.mc;

% Make parts for market share calculation
deltatrue=[A price]*betatrue+ksi;

[share, sij,~]=...
    ShareCalculation(rc_true,deltatrue,data);

dsdp = dsdpfn(sij,data);

root = mc - price - share./sum((owner.*dsdp),2);
```

Function solving the system of supply side FOC:
$0 = c_t - p_t - \Delta_t^{-1} s_t$.

Market share function

```
function [sh,sij,wsij] = ...
    ShareCalculation(theta,delta,data)

%% Unpack
cdid=data.cdid;dummarket=data.dummarket;
xv=data.xv;qweight=data.qweight;
nodes=data.nodes;

%% Market Share
mu = xv.* theta;
mudel=kron(ones(1,nodes),delta)+mu;

numer1 = exp(mudel);
sumMS=(dummarket'*numer1);
denom1=1+sumMS(cdid,:);
sij=(numer1./denom1);
wsij=qweight.*sij;
sh=sum(qweight.*sij,2);
```

Function computing the market share integral numerically.

Market share derivative function

```
function dsdp = dsdpfn(sij,data)

nbrn=data.nbrn;nmkt=data.nmkt;
qweight = data.qweight;
pcoeff = data.betatrue(length(data.betatrue));

dsdp = zeros(nbrn*nmkt,nbrn);
dsdpjj=sum(qweight.*(pcoeff.*sij.*(1-sij)),2);
dsdphelp=zeros(nbrn,nbrn);
for market=1:nmkt
    xind = (market-1)*nbrn+1:market*nbrn;
    sik=sij(xind,:);
    for k=1:nbrn
        dsdphelp(:,k) = ...
            -1.*sum(qweight(xind,:).*...
            (pcoeff.*sij(xind,:).*...
            repmat(sik(k,:),nbrn,1)),2);
    end
    dsdphelp(1:nbrn+1:nbrn^2) = ...
        dsdpjj((market-1)*nbrn+1:market*nbrn);
    dsdp(xind,:) = dsdphelp;
end
```

Function computing market share derivative matrix.

Own:
$\sum_{i=1}^{n} \phi_i \left[-\alpha s_{ijt}(1 - s_{ijt})\right]$,

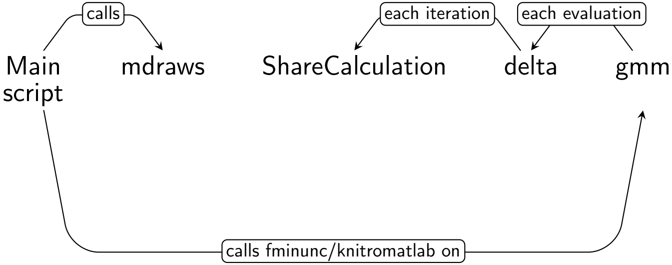
Cross: $\sum_{i=1}^{n} \phi_i \left[\alpha s_{ijt} s_{ikt}\right]$

Main script

```
% X-Matrices for Estimation - part II
Xexo=[A eprice]; % RHS X vector

% True mean utility for market share calculation
deltatrue=Xexo*betatrue+ksi;

% Calculate the True/Observed Market shares
[share,~,~] = ShareCalculation(rc_true,deltatrue,PData);
logobsshare=log(share);
```

Final steps creating the data: market shares.

BLP estimation - only demand side moments



Main script: Setting the stage for estimation

```
%% 3. Data Structure, Instruments,
%% and Weighting Matrix
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Data.nmkt=nmkt;Data.nbrn=nbrn;Data.cdid=cdid;
Data.dummarket=dummarket;Data.owner=owner;
Data.nlin=nlin;
Data.nodes=nodes;Data.qweight=qweight;
Data.logobsshare=logobsshare;
Data.share=share;Data.xv=xv;
Data.qv=qv;Data.nobs=nobs;

% Create sum of rival characteristics
xcomp=dummarket'*A(:,2); % Sum per market
xcomp=xcomp(cdid,:); % Expand to JxT vector
xcomp=xcomp-A(:,2);
```

Construct data structure for estimation. Compute BLP instruments.

Main script: Setting the stage for estimation

```
% Choose set of instruments
if i==1
    Z=[A A(:,2).^2 xcomp];
elseif i==2
    Z=[A A(:,2).^2 z xcomp];
end
% Weighting Matrix - homoscedasticity
norm=mean(mean(Z'*Z),2);
W=inv((Z'*Z)/norm)/norm;

% Some Data to speed up gmm computations
xzwz=Xexo'*Z*W*Z';
Data.xzwz=xzwz;
xzwzx=xzwz*Xexo;
locnorm=mean(mean(xzwzx),2);
Data.invxzwzx=inv(xzwzx/locnorm)/locnorm;

Data.Z = Z;
Data.W = W;
Data.Xrandom=Xrandom;
Data.Xexo = Xexo;
```

Define sets of instruments. Compute initial weighting matrix for GMM estimation. Construct auxiliary matrices.

Main script

```
% Integration of market shares using
% pseudo Monte Carlo integration (1)
% MLHS (2)
% Scrambled Halton (3)
% Quadrature Rule (4)
drawsintegration=4;
% MC: Draws, Quadrature: Precision (KPN: 7)
ndraws = 7;

% Draw v or choose nodes/weights
[qv, qweight] = mdraws(drawsintegration,1,ndraws);

nodes=length(qweight);
% make row vector and duplicate nobs times
qv=ones(nobs,1)*qv';
qweight=ones(nobs,1)*qweight';
```

Draw random numbers / nodes for market share integral.

Main script: Linear OLS, $\sigma = 0$

```
%% 4. Linear Estimation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%OLS
ou=1-sum(reshape(share,nbrn,nmkt),1)';
y=log(share)-log(ou(cdid,:));

bols=(Xexo'*Xexo)\(Xexo'*y);

est=y-Xexo*bols;
dgf=(size(Xexo,1)-size(Xexo,2));
ser=(est'*est)./dgf;
sst=inv(Xexo'*Xexo);
seols=sqrt(ser*diag(sst));
```

Estimate logit with fixed coefficients only by OLS.

Main script: Linear IV, $\sigma = 0$

```
%GMM
%STAGE I: INITIAL WEIGHTING MATRIX*/

mid=Z*W*Z';
btsls=(Xexo'*mid*Xexo)\(Xexo'*mid*y);

xi=y-Xexo*btsls;
sst=inv(Xexo'*mid*Xexo);
ser=(xi'*xi)./dgf;
setsls=sqrt(ser*diag(sst));
```

Estimate logit with fixed coefficients by linear IV, using a set of instruments Z.

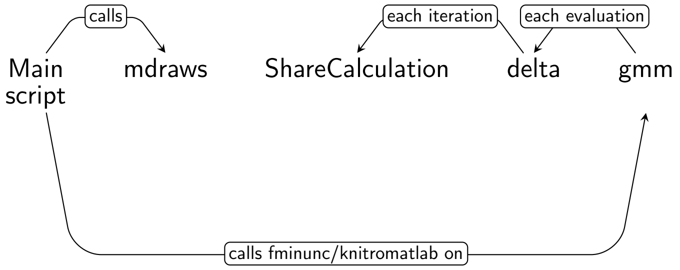
Main script: Estimate full model using NFP algorithm

```
%% 5. NFP algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pass Data into gmm using anonymous functions
theta20=abs(randn(1,1));
angmm = @(theta20)gmm(theta20,Data);

options = ...
    optimset( 'Display','iter',...
              'GradObj','off','TolCon',1E-6,...
              'TolFun',1E-6,'TolX',1E-6,...
              'Hessian', 'off','DerivativeCheck','off');

t1 = cputime;
[theta, fval, exitflag, output, lambda] = ...
fminunc(angmm,theta20, options);

load bet; th12gmm=[bet; theta];
se12gmm=seblp(th12gmm,Data);
cputimegmm=cputime-t1;
```

Full model:

GMM objective function minimized by a solver such as *fminunc* or *Knitro*.

BLP estimation - only demand side moments



GMM objective function

```
function f = gmm(theta,BLPdata)

%% Contraction Mapping
d=delta(theta,BLPdata);

%% GMM
if max(isnan(d)) == 1
    f = 1e10;
else
    % Precomputed:
    % W = inv(Z'*Z);
    % Data.xzwz = Xexo'*Z*W*Z'
    % Data.invxzwzx = inv(Xexo'*Z*W*Z'*Xexo)
    bet = BLPdata.invxzwzx*(BLPdata.xzwz*d);
    csi = d-BLPdata.Xexo*bet;
    f = csi'*BLPdata.Z*BLPdata.W*BLPdata.Z'*csi;
    % Pass estimated beta vector to main script
    save bet bet;
end
```

Nonlinear: δ, contraction mapping

Linear IV: β, by $E[\xi Z] = 0$

δ contraction mapping

```
function delta1 = delta(theta,data)

delta0=zeros(data.nobs,1);
k=100;
km=1e-14;
loshare=data.logobsshare;

while k > km
    [sh, ~, ~] = ...
        ShareCalculation(theta,delta0,data);
    delta1 = delta0 + loshare - log(sh);
           if max(isnan(delta1))==1
              disp('No Convergence')
              break
           end
    k=max(abs(delta1-delta0));
    delta0 = delta1;
end
```

Contraction mapping to find mean utilities:

$\delta^{h+1} =$
$\delta^{h} + ln(S) - ln(s(\cdot))$

GMM objective function

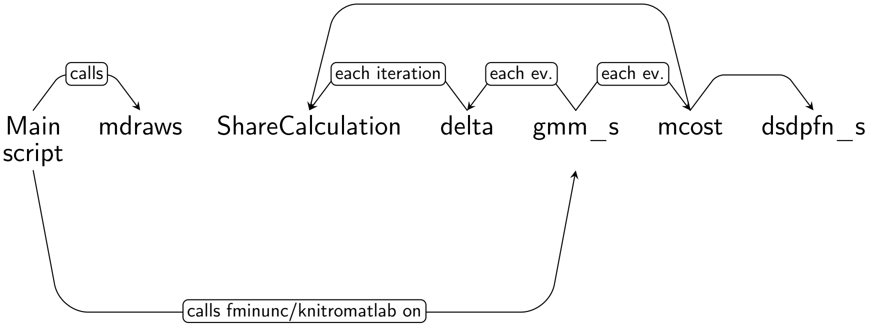
```
function f = gmm(theta,BLPdata)

%% Contraction Mapping
d=delta(theta,BLPdata);

%% GMM
if max(isnan(d)) == 1
    f = 1e10;
else
    % Precomputed:
    % W = inv(Z'*Z);
    % Data.xzwz = Xexo'*Z*W*Z'
    % Data.invxzwzx = inv(Xexo'*Z*W*Z'*Xexo)
    bet = BLPdata.invxzwzx*(BLPdata.xzwz*d);
    csi = d-BLPdata.Xexo*bet;
    f = csi'*BLPdata.Z*BLPdata.W*BLPdata.Z'*csi;
    % Pass estimated beta vector to main script
    save bet bet;
end
```

Nonlinear: δ

Linear IV: β, by $E[\xi Z] = 0$

$\xi = \delta(s, \sigma) - X[\beta, \alpha]$

BLP estimation - demand and supply side moments



Adding supply side moments

```
function f = gmm_s(theta,BLPdata)

%% Contraction Mapping
d=delta(theta,BLPdata);
%% GMM
if max(isnan(d)) == 1
 f = 1e10;
else
    % Demand
    bet = BLPdata.invxzwzx*(BLPdata.xzwz*d);
    csi = d-BLPdata.Xexo*bet;
    % Supply
    mc=mcost(theta,d,bet(3),BLPdata);
    gam = BLPdata.invwwzwzww*(BLPdata.wwzwz*mc);
    omeg = mc-BLPdata.Az*gam;
    % Stack moment conditions
    f = ones(1,2)*...
        [csi'*BLPdata.Z*BLPdata.W*BLPdata.Z'*csi;...
        omeg'*BLPdata.Z*BLPdata.W*BLPdata.Z'*omeg];
    % Pass estimated beta and gamma vectors
    save bet bet;save gam gam;
end
```

Nonlinear: δ

Linear IV: β, by $E[\xi Z] = 0$

$\xi = \delta(s, \sigma) - X[\beta, \alpha]$

$\omega = mc - [X, W]\gamma$

Adding supply side moments

```
% This function computes
% equilibrium-implied marginal cost
function mc = mcost(theta,delta,alpha,data)

owner = data.owner;
price = data.Xexo(:,size(data.Xexo,2));

[share, sij,~]=...
    ShareCalculation(theta,delta,data);

% dsdpfn_s same as dsdpfn, use est. alpha
dsdp = dsdpfn_s(sij,alpha,data);

markup = - sum((owner.*dsdp),2)\share;

mc = price - markup;
```

Compute marginal cost implied by equilibrium markups and observed prices: $mc = p + \Delta^{-1}s$