Matlab codes for Bobenrieth, J.R.A.; Bobenrieth, E.S.A.; Villegas, A.F.; Wright, B.D. (2022), Estimation of Endogenous Volatility Models With Exponential Trends. This document provides the Matlab codes we use in the paper to run Monte Carlo experiments. Codes are divided into the following two cases:

1. Linear case:

    - **dem.m** consumption demand.
    - **invdem.m** inverse consumption demand.
    - **Estimatemodelprices.m** Solve the storage model and generate prices.
    - **onestep.m** Econometric estimation

    dem.m

```
function q = dem(p,a,b)
q=(p-a)./b;
end
```

    invdem.m

```
function p = invdem(q,a,b)
p=a+b.*q;
end
```

    Estimatemodelprices.m

```
function []=eststoragetables(MM,lamda,r,a,b,maxA)
d=0;
h = [-1.755 -1.045 -0.677 -0.386 -0.126 0.126 0.386 0.677 1.045 1.755];
probh= 0.1.*ones(1,10);
h=10*h+100;
tol=10^-10;
minA=min(h);
pointsA=3000;
gridA = linspace(minA,maxA,pointsA);
newA=zeros(length(h),pointsA);
f0=spline(gridA,invdem(gridA,a,b));
discrepancy = 1;
iter = 1;
while (iter < 20000) && ( discrepancy > tol)
f = f0;
for j = 1:length(h)
newA(j,:) = h(j) + max(0, ((1-d)/lamda).*(gridA - dem(ppval(f,gridA),a,b)));
end
p = max(invdem(gridA,a,b), (lamda*(1-d)/(1+r)).* (probh*ppval(f,newA)));
f0 = spline(gridA,p);
discrepancy=max(abs(ppval(f0,gridA) - ppval(f,gridA)));
iter = iter + 1;
end
if iter == 19999;
warning('failure to converge in storage');
end
pstar =((1-d)*lamda/(1+r))*(ppval(f,h)*probh');
A_star=dem(pstar,a,b);
randn('seed',80);
hh=min(max(randn(MM,1),-5),5);
hh=hh*10+100;
AA=zeros(MM,1);
qq=zeros(MM,1);
ss=zeros(MM,1);
pp=zeros(MM,1);
```

```
AA(1,1)=A_star;
pp(1,1)=ppval(f,AA(1,1));
t=0;
for i=2:MM
t = t+1;
if t == 1000
clc; disp(strcat('Generating prices...', num2str(i)));
t = 0;
end
qq(i-1,1)=dem(pp(i-1),a,b);
ss(i-1,1)=max(0,AA(i-1,1)-qq(i-1,1));
AA(i,1)=hh(i,1)+((1-d)/lamda).*ss(i-1,1);
pp(i,1)=max(0,ppval(f,AA(i,1)));
end
save(strcat('simuT=',num2str(MM),'r=',num2str(100*r),'%lamda=',num2str(lamda),'.mat'))
```

onestep.m

```
function [lamda,gama,pstar,exitflag,perstk,maxp,sigma1,sigma2,sigma3] = ...
    onestept(pdat,param0)
T=length(pdat);
P=pdat(2:T)./pdat(1:T-1);
options = optimset(optimset('fminsearch'),...
'Display', 'off',...
'TolFun', 1e-06);
[param0,¬,exitflag]= fminsearch(@(x) ...
    sum((P-x(1).*min(x(2).*((x(3).^(1:T-1)')./pdat(1:T-1)),1)).^2),param0,options);
options = optimset(optimset('fmincon'), 'Display', 'off',...
'LargeScale', 'off', 'Algorithm', 'active-set','TolFun',1e-6, 'MaxFunEval', 5000);
[param] = fmincon(@(x) ...
    sum((P-x(1).*min(x(2).*((x(3).^(1:T-1)')./pdat(1:T-1)),1)).^2),param0,[],[],[],[],param0 ...
    - abs(param0.*0.5),param0 + abs(param0.*0.5),[],options);
gama=param(1);
pstar=param(2);
lamda=param(3);
maxp=max(pdat(1:T-1));
minp=min(pdat(1:T-1));
et1g=P-gama.*min(pstar.*((lamda.^(1:T-1)')./pdat(1:T-1)),1);
lamdas=zeros(T-1,1);
for i=1:T-1
lamdas(i)=lamda^(i);
end
ptg=pdat(1:T-1)./lamdas;
et1ptg=et1g./ptg;
et1pt=zeros(length(et1ptg),1);
for i=1:length(et1ptg)
if ptg(i)>pstar
et1pt(i)=et1ptg(i)*1;
else
et1pt(i)=et1ptg(i)*0;
end
end
invpt=zeros(length(ptg),1);
for i=1:length(ptg)
if ptg(i)>pstar
invpt(i)=(1/ptg(i))*1;
else
invpt(i)=(1/ptg(i))*0;
end
end
et1=zeros(length(et1g),1);
for i=1:length(et1g)
if ptg(i)≤pstar
et1(i)=et1g(i)*1;
else
et1(i)=et1g(i)*0;
end
end
```

```
A=sum((et1pt).^2)/(T-1);
B=sum(et1.^2)/(T-1);
C=sum((invpt).^2)/(T-1);
D=sum((min((pstar./ptg),1)).^2)/(T-1);
LAMDA=zeros(3,3);
SIGMA=zeros(3,3);
LAMDA(1,1)=(4*A*(pstar^2)*gama^2)/(3*lamda^2);
LAMDA(1,2)=(2*A*pstar*gama^2)/lamda;
LAMDA(1,3)=(2*A*(pstar^2)*gama)/lamda;
LAMDA(2,1)=(2*A*pstar*gama^2)/lamda;
LAMDA(2,2)=4*A*gama^2;
LAMDA(2,3)=4*A*pstar*gama;
LAMDA(3,1)=(2*A*(pstar^2)*gama)/lamda;
LAMDA(3,2)=4*A*pstar*gama;
LAMDA(3,3)=4*(B+A*pstar^2);
SIGMA(1,1)=(2*C*(pstar^2)*(gama^2))/(3*lamda^2);
SIGMA(1,2)=(C*pstar*gama^2)/lamda;
SIGMA(1,3)=(C*(pstar^2)*gama)/lamda;
SIGMA(2,1)=(C*pstar*gama^2)/lamda;
SIGMA(2,2)=2*C*gama^2;
SIGMA(2,3)=2*C*pstar*gama;
SIGMA(3,1)=(C*(pstar^2)*gama)/lamda;
SIGMA(3,2)=2*C*pstar*gama;
SIGMA(3,3)=2*D;
OMEGA=SIGMA\LAMDA/SIGMA;
omega1=OMEGA(1,1);
omega2=OMEGA(2,2);
omega3=OMEGA(3,3);
sigma1=sqrt(omega1/T^3);
sigma2=sqrt(omega2/T);
sigma3=sqrt(omega3/T);
trt=zeros(T,1);
for i=1:T
trt(i,1)=lamda^(i);
end
lamdaP=trt.*pstar;
perstk=(length(find(pdat(1:T-1)>lamdaP(1:T-1)))/T);
end
```

2. Iso-elastic case:

    - **dem.m** consumption demand.
    - **invdem.m** inverse consumption demand.
    - **Estimatemodelprices.m** Solve the storage model and generate prices.
    - **onestep.m** Econometric estimation

dem.m

```
function q = dem(p,rho)
q=p.^(-1/rho);
end
```

invdem.m

```
function p = invdem(q,rho)
p=q.^(-rho);
end
```

Estimatemodelprices.m

```
function []=Estimatemodelprices(MM,lamda,r,rho,maxA)
d=0.05;
h0 = [-1.755 -1.045 -0.677 -0.386 -0.126 0.126 0.386 0.677 1.045 1.755];
probh= 0.1.*ones(1,10);
mu1=0;
sigma1=0.5;
h1=(sigma1.*h0)+mu1;
h=exp(h1);
tol=10^-10;
minA=min(h);
pointsA=500;
gridA = linspace(minA,maxA,pointsA);
newA=zeros(length(h),pointsA);
f0=spline(gridA,invdem(gridA,rho));
discrepancy = 1;
iter = 1;
while (iter < 20000) && ( discrepancy > tol)
f = f0;
for j = 1:length(h)
newA(j,:) = h(j) + max(0, ((1-d)/(lamda^(-1/rho))).*(gridA - dem(ppval(f,gridA),rho)));
end
p = max(invdem(gridA,rho), (lamda*(1-d)/(1+r)).* (probh*ppval(f,newA)));
f0 = spline(gridA,p);
discrepancy=max(abs(ppval(f0,gridA) - ppval(f,gridA)));
iter = iter + 1;
end
if iter == 19999;
warning('failure to converge in storage');
end
pstar =((1-d)*lamda/(1+r))*(ppval(f,h)*probh');
A_star=dem(pstar,rho);
plot(gridA,ppval(f,gridA),gridA,invdem(gridA,rho).*(invdem(gridA,rho)>0))
randn('seed',80);
hh0=min(max(randn(MM,1),-5),5);
hh1=(sigma1.*hh0)+mu1;
hh=exp(hh1);
AA=zeros(MM,1);
qq=zeros(MM,1);
ss=zeros(MM,1);
pp=zeros(MM,1);
AA(1,1)=A_star;
pp(1,1)=ppval(f,AA(1,1));
t=0;
for i=2:MM
t = t+1;
```

```
if t == 1000
clc; disp(strcat('Generating prices...', num2str(i)));
t = 0;
end
qq(i-1,1)=dem(pp(i-1),rho);
ss(i-1,1)=max(0,AA(i-1,1)-qq(i-1,1));
AA(i,1)=hh(i,1)+((1-d)/(lamda.^(-1/rho))).*ss(i-1,1);
pp(i,1)=max(0,ppval(f,AA(i,1)));
end
save(strcat('simuT=',num2str(MM),'r=',num2str(100*r),'lamda=',num2str(lamda),'.mat'))
```

onestep.m

```
function [lamda,gama,pstar,exitflag,perstk,maxp,sigma1,sigma2,sigma3] = ...
    onestept(pdat,param0)
T=length(pdat);
P=pdat(2:T)./pdat(1:T-1);
options = optimset(optimset('fminsearch'),...
'Display', 'off',...
'TolFun', 1e-06);
[param0,¬,exitflag]= fminsearch(@(x) ...
    sum((P-x(1).*min(x(2).*((x(3).^(1:T-1)')./pdat(1:T-1)),1)).^2),param0,options);
options = optimset(optimset('fmincon'), 'Display', 'off',...
'LargeScale', 'off', 'Algorithm', 'active-set','TolFun',1e-6, 'MaxFunEval', 5000);
[param] = fmincon(@(x) ...
    sum((P-x(1).*min(x(2).*((x(3).^(1:T-1)')./pdat(1:T-1)),1)).^2),param0,[],[],[],[],param0 ...
    - abs(param0.*0.5),param0 + abs(param0.*0.5),[],options);
gama=param(1);
pstar=param(2);
lamda=param(3);
maxp=max(pdat(1:T-1));
minp=min(pdat(1:T-1));
et1g=P-gama.*min(pstar.*((lamda.^(1:T-1)')./pdat(1:T-1)),1);
lamdas=zeros(T-1,1);
for i=1:T-1
lamdas(i)=lamda^(i);
end
ptg=pdat(1:T-1)./lamdas;
et1ptg=et1g./ptg;
et1pt=zeros(length(et1ptg),1);
for i=1:length(et1ptg)
if ptg(i)>pstar
et1pt(i)=et1ptg(i)*1;
else
et1pt(i)=et1ptg(i)*0;
end
end
invpt=zeros(length(ptg),1);
for i=1:length(ptg)
if ptg(i)>pstar
invpt(i)=(1/ptg(i))*1;
else
invpt(i)=(1/ptg(i))*0;
end
end
et1=zeros(length(et1g),1);
for i=1:length(et1g)
if ptg(i)≤pstar
et1(i)=et1g(i)*1;
else
et1(i)=et1g(i)*0;
end
end
A=sum((et1pt).^2)/(T-1);
B=sum(et1.^2)/(T-1);
C=sum((invpt).^2)/(T-1);
D=sum((min((pstar./ptg),1)).^2)/(T-1);
LAMDA=zeros(3,3);
```

```
SIGMA=zeros(3,3);
LAMDA(1,1)=(4*A*(pstar^2)*gama^2)/(3*lamda^2);
LAMDA(1,2)=(2*A*pstar*gama^2)/lamda;
LAMDA(1,3)=(2*A*(pstar^2)*gama)/lamda;
LAMDA(2,1)=(2*A*pstar*gama^2)/lamda;
LAMDA(2,2)=4*A*gama^2;
LAMDA(2,3)=4*A*pstar*gama;
LAMDA(3,1)=(2*A*(pstar^2)*gama)/lamda;
LAMDA(3,2)=4*A*pstar*gama;
LAMDA(3,3)=4*(B+A*pstar^2);
SIGMA(1,1)=(2*C*(pstar^2)*(gama^2))/(3*lamda^2);
SIGMA(1,2)=(C*pstar*gama^2)/lamda;
SIGMA(1,3)=(C*(pstar^2)*gama)/lamda;
SIGMA(2,1)=(C*pstar*gama^2)/lamda;
SIGMA(2,2)=2*C*gama^2;
SIGMA(2,3)=2*C*pstar*gama;
SIGMA(3,1)=(C*(pstar^2)*gama)/lamda;
SIGMA(3,2)=2*C*pstar*gama;
SIGMA(3,3)=2*D;
OMEGA=SIGMA\LAMDA/SIGMA;
omega1=OMEGA(1,1);
omega2=OMEGA(2,2);
omega3=OMEGA(3,3);
sigma1=sqrt(omega1/T^3);
sigma2=sqrt(omega2/T);
sigma3=sqrt(omega3/T);
trt=zeros(T,1);
for i=1:T
trt(i,1)=lamda^(i);
end
lamdaP=trt.*pstar;
perstk=(length(find(pdat(1:T-1)>lamdaP(1:T-1)))/T);
end
```