

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

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

invdem.m

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

Estimatemodelprices.m

```
1 function []=eststoragetables(MM,lamda,r,a,b,maxA)
2 d=0;
3 h = [-1.755 -1.045 -0.677 -0.386 -0.126 0.126 0.386 0.677 1.045 1.755];
4 probh= 0.1.*ones(1,10);
5 h=10*h+100;
6 tol=10^-10;
7 minA=min(h);
8 pointsA=3000;
9 gridA = linspace(minA,maxA,pointsA);
10 newA=zeros(length(h),pointsA);
11 f0=spline(gridA,invdem(gridA,a,b));
12 discrepancy = 1;
13 iter = 1;
14 while (iter < 20000) && (discrepancy > tol)
15 f = f0;
16 for j = 1:length(h)
17 newA(j,:) = h(j) + max(0, ((1-d)/lamda).*(gridA - dem(ppval(f,gridA),a,b)));
18 end
19 p = max(invdem(gridA,a,b), (lamda*(1-d)/(1+r)).*(probh*ppval(f,newA)));
20 f0 = spline(gridA,p);
21 discrepancy=max(abs(ppval(f0,gridA) - ppval(f,gridA)));
22 iter = iter + 1;
23 end
24 if iter == 19999;
25 warning('failure to converge in storage');
26 end
27 pstar =((1-d)*lamda/(1+r)).*(ppval(f,h)*probh');
28 A_star=dem(pstar,a,b);
29 randn('seed',80);
30 hh=min(max(randn(MM,1),-5),5);
31 hh=hh*10+100;
32 AA=zeros(MM,1);
33 qq=zeros(MM,1);
34 ss=zeros(MM,1);
35 pp=zeros(MM,1);
```

```

36 AA(1,1)=A_star;
37 pp(1,1)=ppval(f,AA(1,1));
38 t=0;
39 for i=2:MM
40 t = t+1;
41 if t == 1000
42 clc; disp(strcat('Generating prices...', num2str(i)));
43 t = 0;
44 end
45 qq(i-1,1)=dem(pp(i-1),a,b);
46 ss(i-1,1)=max(0,AA(i-1,1)-qq(i-1,1));
47 AA(i,1)=hh(i,1)+((1-d)/lamda).*ss(i-1,1);
48 pp(i,1)=max(0,ppval(f,AA(i,1)));
49 end
50 save(strcat('simuT=',num2str(MM),'r=',num2str(100*r),'%lamda=',num2str(lamda),'.mat'))

```

### onestep.m

```

1 function [lamda,gama,pstar,exitflag,perstk,maxp,sigma1,sigma2,sigma3] = ...
   onestep(pdat,param0)
2 T=length(pdat);
3 P=pdat(2:T)./pdat(1:T-1);
4 options = optimset(optimset('fminsearch'),...
5 'Display','off',...
6 'TolFun', 1e-06);
7 [param0,~,exitflag]= fminsearch(@(x) ...
   sum((P-x(1).*(min(x(2).*(x(3).^(1:T-1))./pdat(1:T-1)),1)).^2),param0,options);
8 options = optimset(optimset('fmincon'),'Display','off',...
9 'LargeScale','off','Algorithm','active-set','TolFun',1e-6,'MaxFunEval', 5000);
10 [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);
11 gama=param(1);
12 pstar=param(2);
13 lamda=param(3);
14 maxp=max(pdat(1:T-1));
15 minp=min(pdat(1:T-1));
16 etlg=P-gama.*min(pstar.*((lamda.^(1:T-1))./pdat(1:T-1)),1);
17 lamdas=zeros(T-1,1);
18 for i=1:T-1
19 lamdas(i)=lamda^i;
20 end
21 ptg=pdat(1:T-1)./lamdas;
22 etlptg=etlg./ptg;
23 etlpt=zeros(length(etlptg),1);
24 for i=1:length(etlptg)
25 if ptg(i)>pstar
26 etlpt(i)=etlptg(i)*1;
27 else
28 etlpt(i)=etlptg(i)*0;
29 end
30 end
31 invpt=zeros(length(ptg),1);
32 for i=1:length(ptg)
33 if ptg(i)>pstar
34 invpt(i)=(1/ptg(i))*1;
35 else
36 invpt(i)=(1/ptg(i))*0;
37 end
38 end
39 etl=zeros(length(etlg),1);
40 for i=1:length(etlg)
41 if ptg(i)<=pstar
42 etl(i)=etlg(i)*1;
43 else
44 etl(i)=etlg(i)*0;
45 end
46 end

```

```

47 A=sum((et1pt).^2)/(T-1);
48 B=sum(et1.^2)/(T-1);
49 C=sum((invpt).^2)/(T-1);
50 D=sum((min((pstar./ptg),1)).^2)/(T-1);
51 LAMDA=zeros(3,3);
52 SIGMA=zeros(3,3);
53 LAMDA(1,1)=(4*A*(pstar^2)*gama^2)/(3*lamda^2);
54 LAMDA(1,2)=(2*A*pstar*gama^2)/lamda;
55 LAMDA(1,3)=(2*A*(pstar^2)*gama)/lamda;
56 LAMDA(2,1)=(2*A*pstar*gama^2)/lamda;
57 LAMDA(2,2)=4*A*gama^2;
58 LAMDA(2,3)=4*A*pstar*gama;
59 LAMDA(3,1)=(2*A*(pstar^2)*gama)/lamda;
60 LAMDA(3,2)=4*A*pstar*gama;
61 LAMDA(3,3)=4*(B+A*pstar^2);
62 SIGMA(1,1)=(2*C*(pstar^2)*(gama^2))/(3*lamda^2);
63 SIGMA(1,2)=(C*pstar*gama^2)/lamda;
64 SIGMA(1,3)=(C*(pstar^2)*gama)/lamda;
65 SIGMA(2,1)=(C*pstar*gama^2)/lamda;
66 SIGMA(2,2)=2*C*gama^2;
67 SIGMA(2,3)=2*C*pstar*gama;
68 SIGMA(3,1)=(C*(pstar^2)*gama)/lamda;
69 SIGMA(3,2)=2*C*pstar*gama;
70 SIGMA(3,3)=2*D;
71 OMEGA=SIGMA\LAMDA/SIGMA;
72 omegal=OMEGA(1,1);
73 omega2=OMEGA(2,2);
74 omega3=OMEGA(3,3);
75 sigma1=sqrt(omegal/T^3);
76 sigma2=sqrt(omega2/T);
77 sigma3=sqrt(omega3/T);
78 trt=zeros(T,1);
79 for i=1:T
80 trt(i,1)=lamda^(i);
81 end
82 lamdaP=trt.*pstar;
83 perstk=(length(find(pdat(1:T-1)>lamdaP(1:T-1)))/T);
84 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

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

invdem.m

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

Estimatemodelprices.m

```
1 function []=Estimatemodelprices(MM, lamda, r, rho, maxA)
2 d=0.05;
3 h0 = [-1.755 -1.045 -0.677 -0.386 -0.126 0.126 0.386 0.677 1.045 1.755];
4 probh= 0.1.*ones(1,10);
5 mul=0;
6 sigma1=0.5;
7 h1=(sigma1.*h0)+mul;
8 h=exp(h1);
9 tol=10^-10;
10 minA=min(h);
11 pointsA=500;
12 gridA = linspace(minA,maxA,pointsA);
13 newA=zeros(length(h),pointsA);
14 f0=spline(gridA, invdem(gridA, rho));
15 discrepancy = 1;
16 iter = 1;
17 while (iter < 20000) && ( discrepancy > tol)
18 f = f0;
19 for j = 1:length(h)
20 newA(j,:) = h(j) + max(0, ((1-d)/(lamda^(-1/rho)))*(gridA - dem(ppval(f,gridA),rho)));
21 end
22 p = max(invdem(gridA, rho), (lamda*(1-d)/(1+r)).* (probh*ppval(f,newA)));
23 f0 = spline(gridA,p);
24 discrepancy=max(abs(ppval(f0,gridA) - ppval(f,gridA)));
25 iter = iter + 1;
26 end
27 if iter == 19999;
28 warning('failure to converge in storage');
29 end
30 pstar =((1-d)*lamda/(1+r))* (ppval(f,h)*probh');
31 A.star=dem(pstar, rho);
32 plot(gridA, ppval(f, gridA), gridA, invdem(gridA, rho).*(invdem(gridA, rho)>0))
33 randn('seed', 80);
34 hh0=min(max(randn(MM,1),-5),5);
35 hhl=(sigma1.*hh0)+mul;
36 hh=exp(hhl);
37 AA=zeros(MM,1);
38 qq=zeros(MM,1);
39 ss=zeros(MM,1);
40 pp=zeros(MM,1);
41 AA(1,1)=A.star;
42 pp(1,1)=ppval(f,AA(1,1));
43 t=0;
44 for i=2:MM
45 t = t+1;
```

```

46 if t == 1000
47 clc; disp(strcat('Generating prices...', num2str(i)));
48 t = 0;
49 end
50 qq(i-1,1)=dem(pp(i-1),rho);
51 ss(i-1,1)=max(0,AA(i-1,1)-qq(i-1,1));
52 AA(i,1)=hh(i,1)+((1-d)/(lamda.^(-1/rho))).*ss(i-1,1);
53 pp(i,1)=max(0,ppval(f,AA(i,1)));
54 end
55 save(strcat('simuT=',num2str(MM),'r=',num2str(100*r),'lamda=',num2str(lamda),'.mat'))

```

### onestep.m

```

1 function [lamda,gama,pstar,exitflag,perstk,maxp,sigma1,sigma2,sigma3] = ...
   onestep(pdat,param0)
2 T=length(pdat);
3 P=pdat(2:T)./pdat(1:T-1);
4 options = optimset(optimset('fminsearch'),...
5 'Display','off',...
6 'TolFun', 1e-06);
7 [param0,~,exitflag]= fminsearch(@(x) ...
   sum((P-x(1).*(min(x(2).*(x(3).^(1:T-1))./pdat(1:T-1)),1)).^2),param0,options);
8 options = optimset(optimset('fmincon'),'Display','off',...
9 'LargeScale','off','Algorithm','active-set','TolFun',1e-6,'MaxFunEval', 5000);
10 [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);
11 gama=param(1);
12 pstar=param(2);
13 lamda=param(3);
14 maxp=max(pdat(1:T-1));
15 minp=min(pdat(1:T-1));
16 etlg=P-gama.*min(pstar.*((lamda.^(1:T-1))./pdat(1:T-1)),1);
17 lamdas=zeros(T-1,1);
18 for i=1:T-1
19 lamdas(i)=lamda^i;
20 end
21 ptg=pdat(1:T-1)./lamdas;
22 etlptg=etlg./ptg;
23 etlpt=zeros(length(etlptg),1);
24 for i=1:length(etlptg)
25 if ptg(i)>pstar
26 etlpt(i)=etlptg(i)*1;
27 else
28 etlpt(i)=etlptg(i)*0;
29 end
30 end
31 invpt=zeros(length(ptg),1);
32 for i=1:length(ptg)
33 if ptg(i)>pstar
34 invpt(i)=(1/ptg(i))*1;
35 else
36 invpt(i)=(1/ptg(i))*0;
37 end
38 end
39 etl=zeros(length(etlg),1);
40 for i=1:length(etlg)
41 if ptg(i)<=pstar
42 etl(i)=etlg(i)*1;
43 else
44 etl(i)=etlg(i)*0;
45 end
46 end
47 A=sum((etlpt).^2)/(T-1);
48 B=sum(etl.^2)/(T-1);
49 C=sum((invpt).^2)/(T-1);
50 D=sum((min((pstar./ptg),1)).^2)/(T-1);
51 LAMDA=zeros(3,3);

```

```

52 SIGMA=zeros(3,3);
53 LAMDA(1,1)=(4*A*(pstar^2)*gama^2)/(3*lamda^2);
54 LAMDA(1,2)=(2*A*pstar*gama^2)/lamda;
55 LAMDA(1,3)=(2*A*(pstar^2)*gama)/lamda;
56 LAMDA(2,1)=(2*A*pstar*gama^2)/lamda;
57 LAMDA(2,2)=4*A*gama^2;
58 LAMDA(2,3)=4*A*pstar*gama;
59 LAMDA(3,1)=(2*A*(pstar^2)*gama)/lamda;
60 LAMDA(3,2)=4*A*pstar*gama;
61 LAMDA(3,3)=4*(B+A*pstar^2);
62 SIGMA(1,1)=(2*C*(pstar^2)*(gama^2))/(3*lamda^2);
63 SIGMA(1,2)=(C*pstar*gama^2)/lamda;
64 SIGMA(1,3)=(C*(pstar^2)*gama)/lamda;
65 SIGMA(2,1)=(C*pstar*gama^2)/lamda;
66 SIGMA(2,2)=2*C*gama^2;
67 SIGMA(2,3)=2*C*pstar*gama;
68 SIGMA(3,1)=(C*(pstar^2)*gama)/lamda;
69 SIGMA(3,2)=2*C*pstar*gama;
70 SIGMA(3,3)=2*D;
71 OMEGA=SIGMA\LAMDA/SIGMA;
72 omega1=OMEGA(1,1);
73 omega2=OMEGA(2,2);
74 omega3=OMEGA(3,3);
75 sigma1=sqrt(omega1/T^3);
76 sigma2=sqrt(omega2/T);
77 sigma3=sqrt(omega3/T);
78 trt=zeros(T,1);
79 for i=1:T
80 trt(i,1)=lamda^(i);
81 end
82 lamdaP=trt.*pstar;
83 perstk=(length(find(pdat(1:T-1)>lamdaP(1:T-1)))/T);
84 end

```