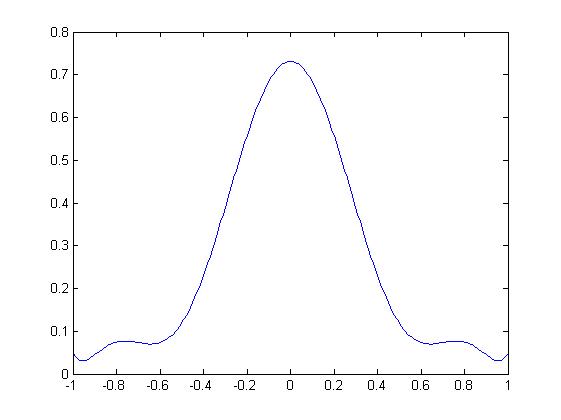
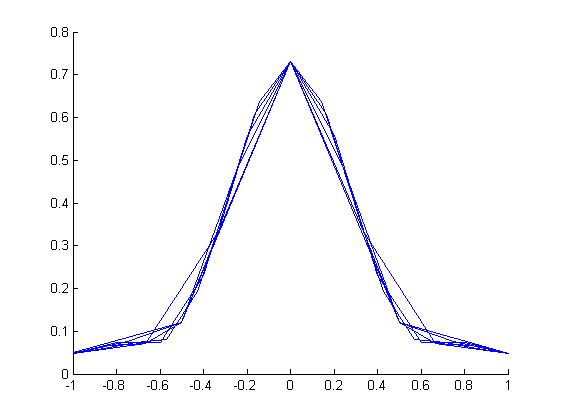
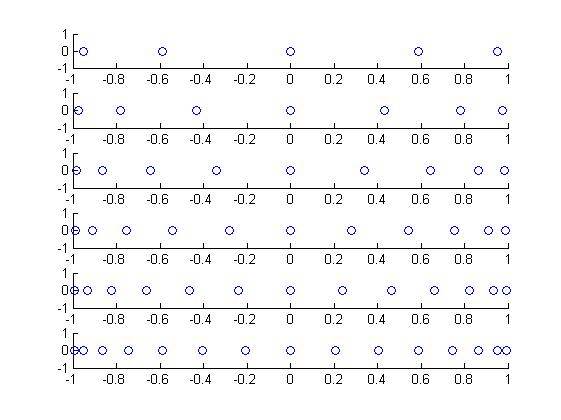
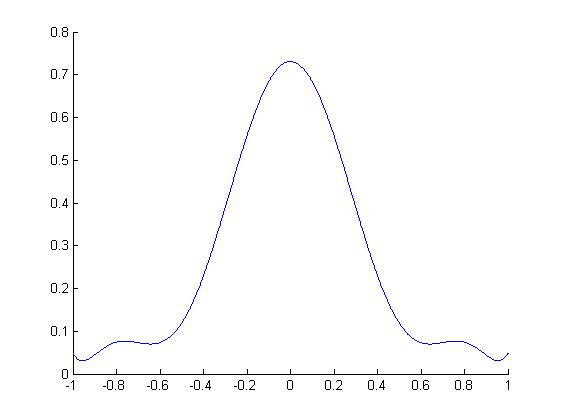
Computational Economics Problem Set 4

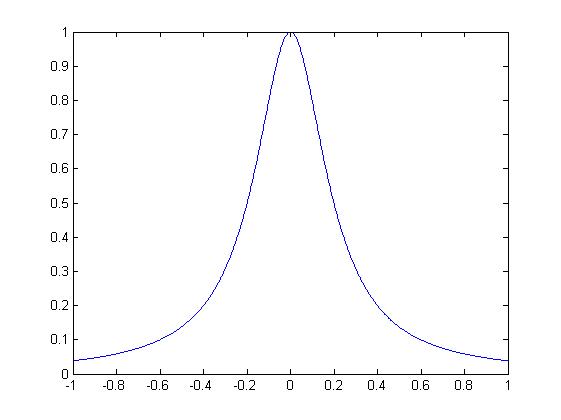
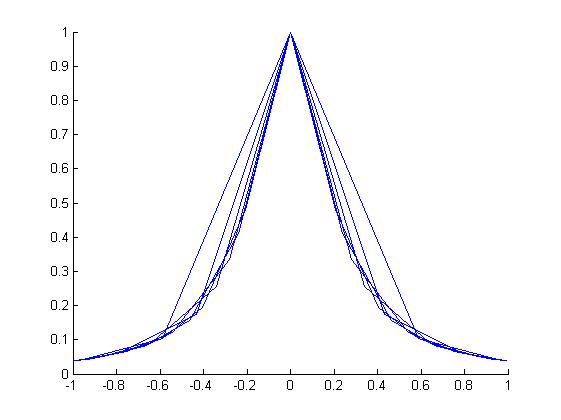
Names: Di Lu(5706079),), Zexi Sun(5917910), Penghui Yin(5702358), Shiwei Yu(5913037)

**Problem 1**

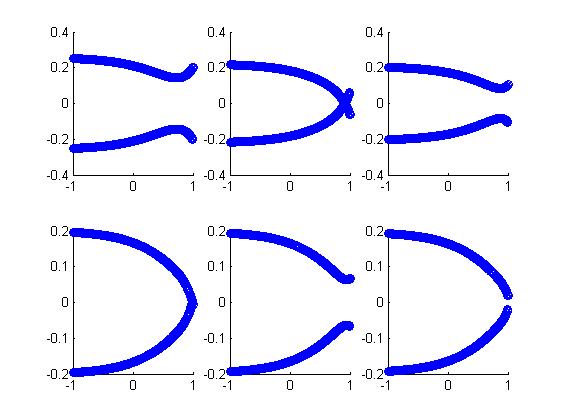
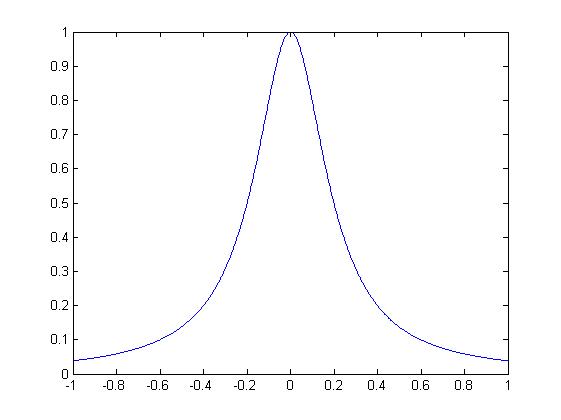


We can see from the left picture that with the increase number of equally distributed nodes from 5 to 15, we approximate the function better. In the right picture, with a large number 100, we have approximated the function the same as the number 1000. And they don’t have nice property on both sides of the tail.



When we repeat the same process for Chebychev nodes, we see from the up left picture that the function has been already approximated as well as the large number of evenly distributed nodes, no matter how the number of nodes changes from 5 to 15. In the case of 100 nodes, the Chebychev nodes has already approximated curve very well in the down left picture. However, in the down right picture, we use the number of nodes instead of the whole length of x-axis to approximate the function, and it looks like the even-distributed nodes, or even does a worse job. For a test, we also use the whole length of x-axis with the normal order polynomial from 5 to 15 estimated from the first practice and the result is totally a disaster as the following left picture shows. Besides that, we have an interesting finding, we did all these exercises by setting the path in the toolbox of M&F. But after we add this toolbox as the subpath of Matlab’s default setting. The following right picture also shows up in the large number of equivalent distributed nodes and small number of Chebychev nodes. The possibility is that matlab warns there is conflict between the eps function in the toolbox and in the default function, and we set the default as the original default one. So we suppose that as the accuracy increases, the problem of bad tails disappear.

See code appendix PS4q1

**Problem 2**

2.1

If , then the first derivative with respect to Ct is

F.O.C wrt C0 in Plug in

we get .

Rearrange, we get

Certainty equivalence means that uncertainty does not affect the choices an individual takes. As we can see here, only the expectation matters.

2.2

C0 star remains the same if variance increases, it does not make economic sense if people are risk averse. However in our utility setting, people are risk neutral.

2.3

If , then the first derivative with respect to Ct is

F.O.C wrt C0 in Plug in

we get

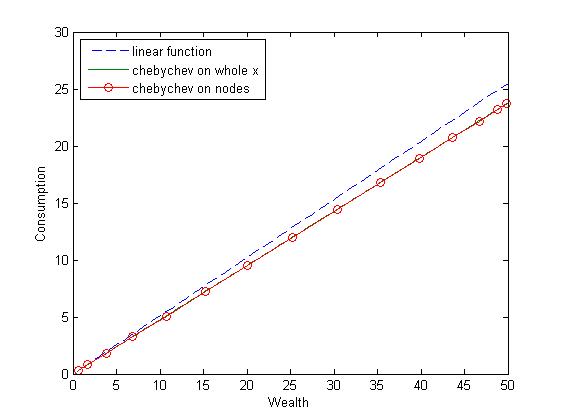
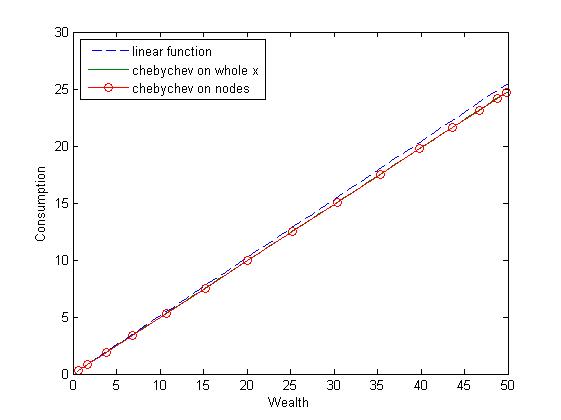
Rearrange, we get

If the variance of r increases, utility for future consumption decreases as there is more uncertainty, so the agent will consume more in period 0.

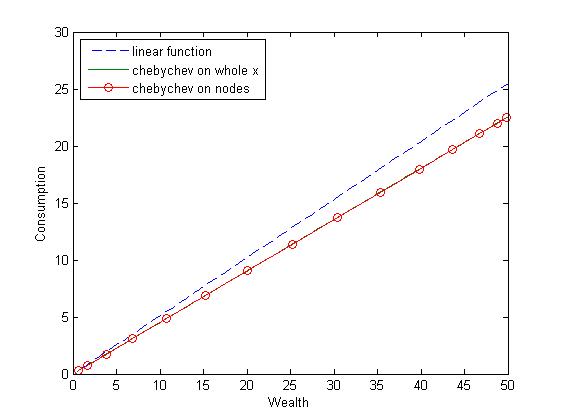
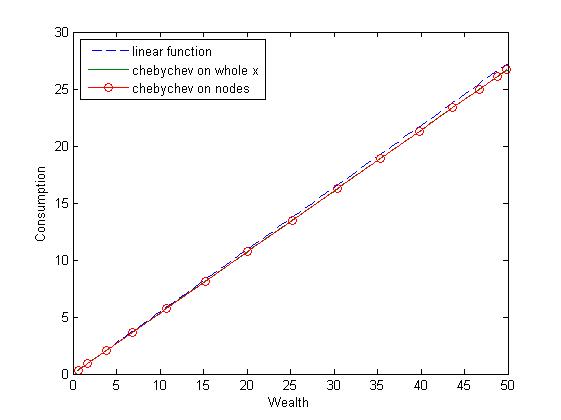
2.4&2.5

We use the toolbox by M&F and the method of collaction of M&F, the results are following.

Code see code appendix ps4q2 and resid



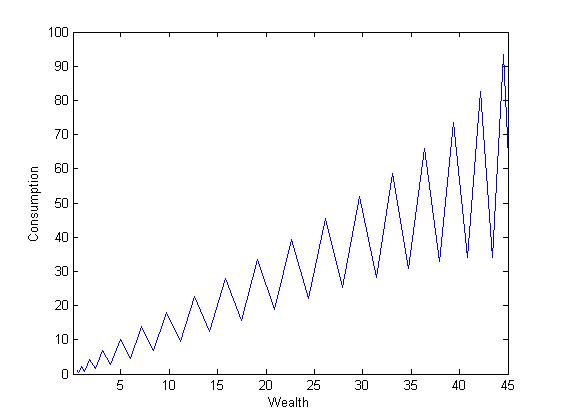
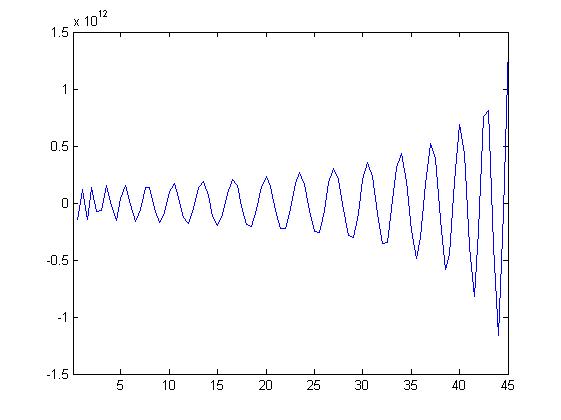
rmin=-0.08;rhigh=0.12;gemma=2;p=0.5; gemma=10



p=0.15(chance for small prize) rmin=-0.2,rhigh=0.24

We can see in every picture the solution for the linear function against the Chebychev approximation. There are both Chebychev on the whole x axis and the nodes, but the results coincide as always. The results is that in the CRRA utility assumption as the Chebychev line, richer agents save more than the poor, so there consumption deviates from the straight linear line when they have more asset in the beginning. We change different parameters as the other three pictures show us and the basic result remain the same. We increase the gemma from 2 to 10 in picture 2 and the consumption line for CRRA deviates from the original line earlier in terms of original asset. As gemma itself is called the risk averse degree, the more agents unlike the risk, they save more and thus consume less. The third picture decreases the opportunity getting a bad investment result. The result also makes sense as people expect they are more likely to win a good prize, they consume more. We increase the gap or variance of interest rate in the last picture, the result is similar to the one we increase the gemma. It is of no surprise that here we increase the real risk instead of increasing agent’s degree of risk averse, which should have the same effect.

However, there are three things need mentioned. First is the number of nodes, there is a critical change if we increase the number to 44, and then we get weird pictures as follow. The left one is the residual and the right one is the consumption against wealth. They both have the feature of volatility and so to say, a worse result. So it is not right that the more nodes, the better result we will get. Second, in terms of the residual, when number of nodes is within the normal range, the residual has a linear relationship with the wealth, the slope is around 0.5. So there will be a larger error if we increase the wealth. And actually, it is not tolerable even with asset equal to 0.05, when the minimum absolute error is equal to 0.025 and the relative error remains 1 through the range of wealth. The third thing is also about residual. In fact, we find our result is very sensitive to the way we define the residual. If we change the form of having c to the power of minus gemma on the left side in the function residual, the methodology total breaks out. And that’s we why we create such a file with only two lines separately for easier future different try.



Code Appendix

|  |
| --- |
| Ps4q1  clc; close all; clear all;  %% Problem Set 4, Question 1  % USE fspace = fundefn(bastype,n,a,b,order) in M&F  % Setup Function  syms x y  y=1/(1+25\*x^2);  f=matlabFunction(y);  % Setup Parameters  bastype='cheb';n=100;%nth-degree Chebychev approximants;  a=-1;b=1;node=(5:2:15);l=length(node);  fspace = fundefn(bastype,n,a,b);%fspace is a structured MATLAB variable containing  %numerous fields of information necessary for forming approximations in the chosen  %function space  c = funfitf(fspace,f);%compute the coefficient vector for the approximant  %that interpolates the function at the standard Chebychev nodes  x\_v = nodeunif(1001,-1,1);%simulate real function  y\_ch = funeval(c,fspace,x\_v);%simlated function  plot(x\_v,y\_ch-f(x\_v));%compare difference  figure;plot(x\_v,y\_ch)  x\_ch = funnode(fspace);  figure;o=zeros(length(x\_ch),1);scatter(x\_ch,o)    %% To simulate on fix nodes  x\_eq=(-1:0.01:1)';%define a vector of equidistant nodes x  B=funbas(fspace,x\_eq);%returns the matrix containing the values of the basis functions evaluated at the points x.  y\_eq=B\*c;%you next calculate the function values at x  c\_B=B\y\_eq;%get the polynomial coefficients,think as OLS!!  figure;plot(x\_eq,y\_eq);    figure;  hold on  for i=1:1:l  x\_eq\_i=(-1:2/(-1+node(i)):1)';  B\_i=funbas(fspace,x\_eq\_i);  y\_eq\_i=B\_i\*c;  plot(x\_eq\_i,y\_eq\_i);  end  hold off  %Repeat the exercise using Chebychev nodes.  figure;hold on  for i=1:1:l  fspace\_i = fundefn(bastype,node(i),a,b);  x\_ch\_i = funnode(fspace\_i);  c\_i = funfitf(fspace\_i,f);  y\_ch\_i = funeval(c,fspace,x\_v);  plot(x\_v,y\_ch\_i);  end  hold off    figure;hold on  for i=1:1:l  fspace\_i = fundefn(bastype,node(i),a,b);  x\_ch\_i = funnode(fspace\_i);  c\_i = funfitf(fspace\_i,f);  y\_ch\_i = funeval(c,fspace,x\_ch\_i);  plot(x\_ch\_i,y\_ch\_i);  end  hold off    figure  for i=1:1:l  fspace\_i = fundefn(bastype,node(i),a,b);  x\_ch\_i = funnode(fspace\_i);  o\_i=zeros(length(x\_ch\_i),1);  subplot(l,1,i);  scatter(x\_ch\_i,o\_i)  end    x\_vand=vander(x\_v);  figure;  for i=1:1:l  fspace\_i = fundefn(bastype,node(i),a,b);  c\_i = funfitf(fspace\_i,f);  x\_eq\_i=(-1:2/(-1+node(i)):1)';  B\_i=funbas(fspace\_i,x\_eq\_i);  y\_eq\_i=B\_i\*c\_i;  c\_B\_i=B\_i\y\_eq\_i;  y\_eqv=x\_vand(1:node(i),:)'\*c\_B\_i;  subplot(2,3,i);  scatter(x\_v,y\_eqv);  end |

Ps4q2

|  |
| --- |
| clc; close all; clear all;  %Part two with different parameters  %1.increas gemma, was 2;2.decrease p, was 0.5;3.enlarge the gap of rmin&rhigh;was  %-0.08&0.12  w=(0.5:0.5:50)';rmin=-0.08;rhigh=0.12;gemma=10;p=0.5;%for rmin  %linear solution  c\_l=0.5\*w\*(1+p\*rmin+(1-p)\*rhigh);  %Chebychev intepolation based on M&F collation method  n=15;a=w(1);b=w(length(w));%critical change when n=44  fspace = fundefn('cheb',n,a,b);  w\_ch = funnode(fspace);  coeff = 0.1\*(1:1:length(w\_ch))';  %coeff = ones(length(w\_ch),1);  coeff = broyden('resid',coeff,w\_ch,fspace,gemma,p,rmin,rhigh);  splot = funeval(coeff,fspace,w);  plot(w,splot);%plot the risidual function  c1 = funeval(coeff,fspace,w\_ch);  c2 = funeval(coeff,fspace,w);  figure  plot(w,c\_l,'--',w,c2,w\_ch,c1,'-o')  legend('linear function','chebychev on whole x','chebychev on nodes','Location','NorthWest')  xlabel('Wealth')  ylabel('Consumption') |

Resid

|  |
| --- |
| function r = resid(coeff,w\_ch,fspace,gemma,p,rmin,rhigh)  c = funeval(coeff,fspace,w\_ch);  r = -c+(p.\*(w\_ch.\*(1+rmin)-c).^(-gemma)+(1-p).\*(w\_ch.\*(1+rhigh)-c).^(-gemma)).^(-1/gemma);  %very sensitive to the form of the residual, for example c or c^-gemma on the LHS |