Homework #4

Table of Contents

Ian Blackstone
Math 365, Fall 2016

Problems

```
function hmwk4()

hmwk_problem(@prob2,'prob2');
hmwk_problem(@prob3,'prob3');
hmwk_problem(@prob4,'prob4');
hmwk_problem(@prob5,'prob5');
hmwk_problem(@prob6,'prob6');


end

function hmwk_problem(prob,msg)

try
    prob()
    fprintf('%s : Success!\n',msg);
catch me
    fprintf('%s : Something went wrong.\n',msg);
    fprintf('%s\n',me.message);
end
fprintf('\n');
end
```

Problem #1 : Polynomial Interpolation

```
function prob1()

% part a, given values of (-1,6),(0,1) and (1,2) determine unique 2nd
% degree polynomial p2(x) to the data.
p2 = @(x) 3.*x.*(x-1)-(x+1).*(x-1)+2.*(x+1)/2.*x;
x = -1:1:1;

% part b) we can approach using Van Der Monde matricies where we have
% three equations and three unknowns to solve for our coefficients
```

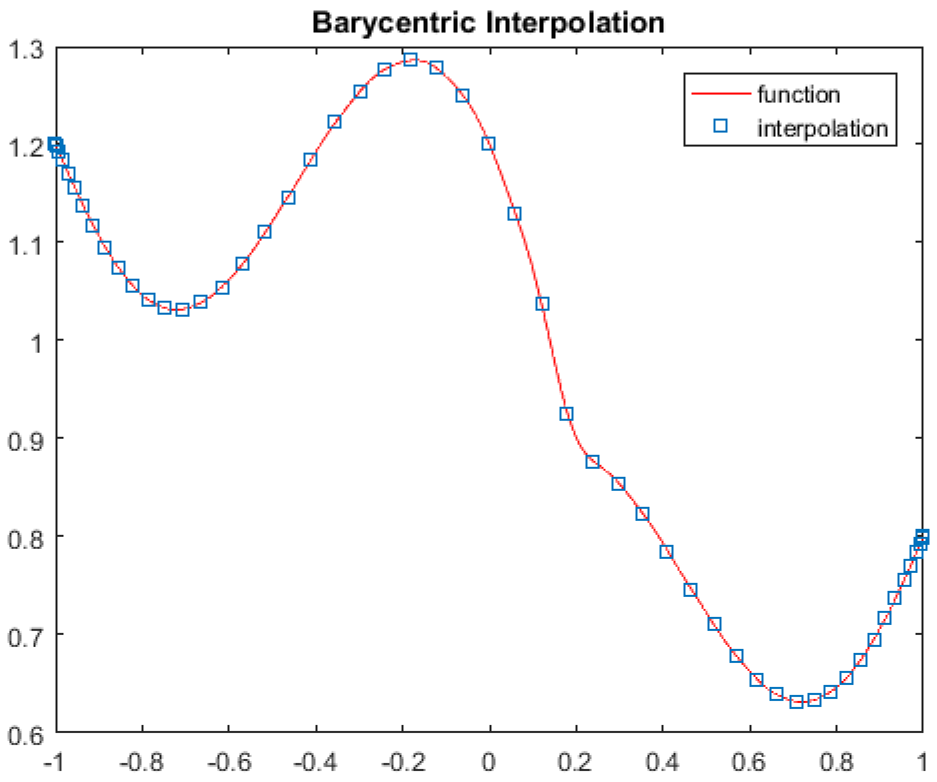
```
A = vander(x);
B = [6;1;2];

coeff = A\B;
a0 = coeff(3)
a1 = coeff(2)
a2 = coeff(1)

p2new = @(x) a0+a1.*x+ a2.*x.^2

% part c: check that a and b give same result:
X = linspace(-1,1,20);

err = p2(X) - p2new(X);
fprintf('The maximum error is %.3f \n',max(err))

end
```

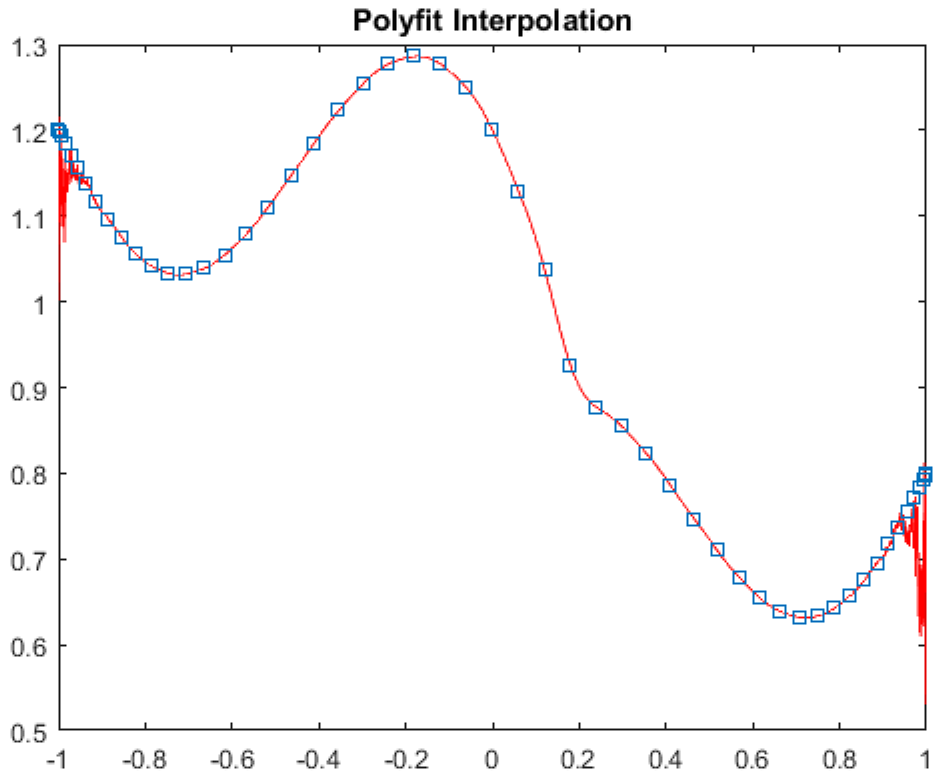
Problem #2 : Barycentric Interpolation

```
function prob2()

% part a
% create 52 points to evaluate the function at
n = 52;
xj = -cos((0:n)*pi/n);

% Delcare the function and create y values at the desired points
f = @(x) abs(x-0.2)+(1-x.^2).^3;
fj = f(xj);

% Create 1000 points to evaluate the interpolation at.
m = 1000;
u = -cos(pi*(0:m)/m);

% Create the barycentered interpolation of the function.
qn = baryinterp(xj,fj,u);

% Plot the function and the interpolation of the function
figure
plot(u,qn,'r-',xj,fj,'s')
legend('function','interpolation')
title('Barycentric Interpolation')

% part b
% create an interpolation of the function using the polyfit function
 and
% evaluate the polynomial at the points created earlier using polyval
cj = polyfit(xj,fj,n);
pn = polyval(cj,u);

% plot the function and the interpolation
figure
```

```
plot(u,pn,'r-',xj,fj,'s')
title('Polyfit Interpolation')
end
```

```
Warning: Polynomial is badly conditioned. Add points with distinct X
 values,
reduce the degree of the polynomial, or try centering and scaling as
 described
in HELP POLYFIT.
prob2 : Success!
```



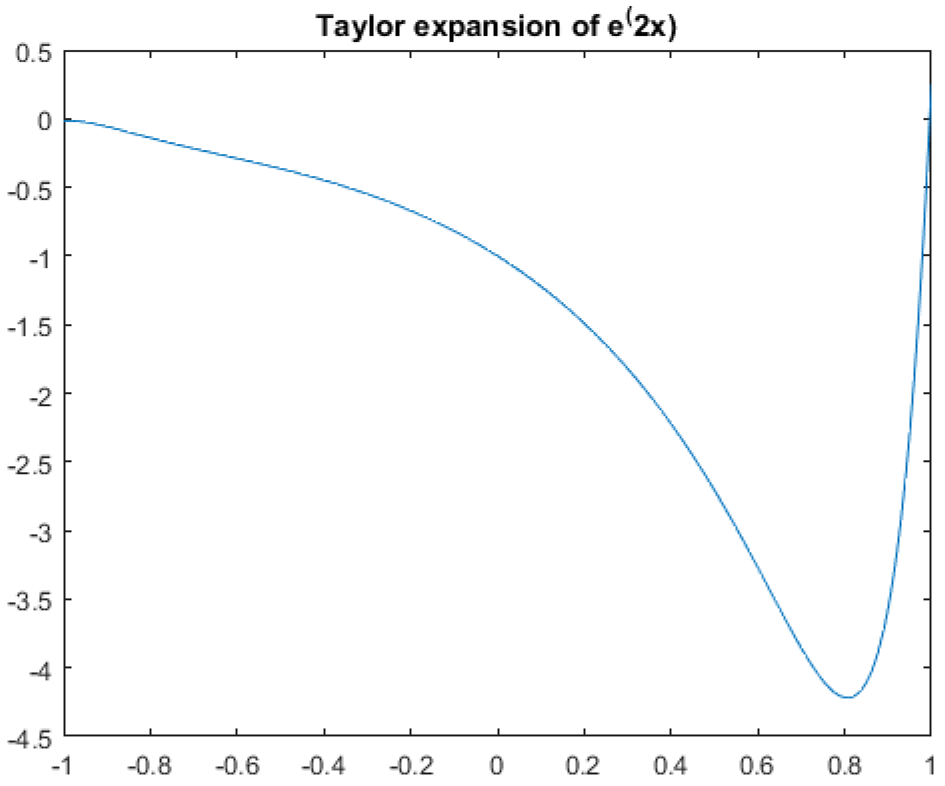


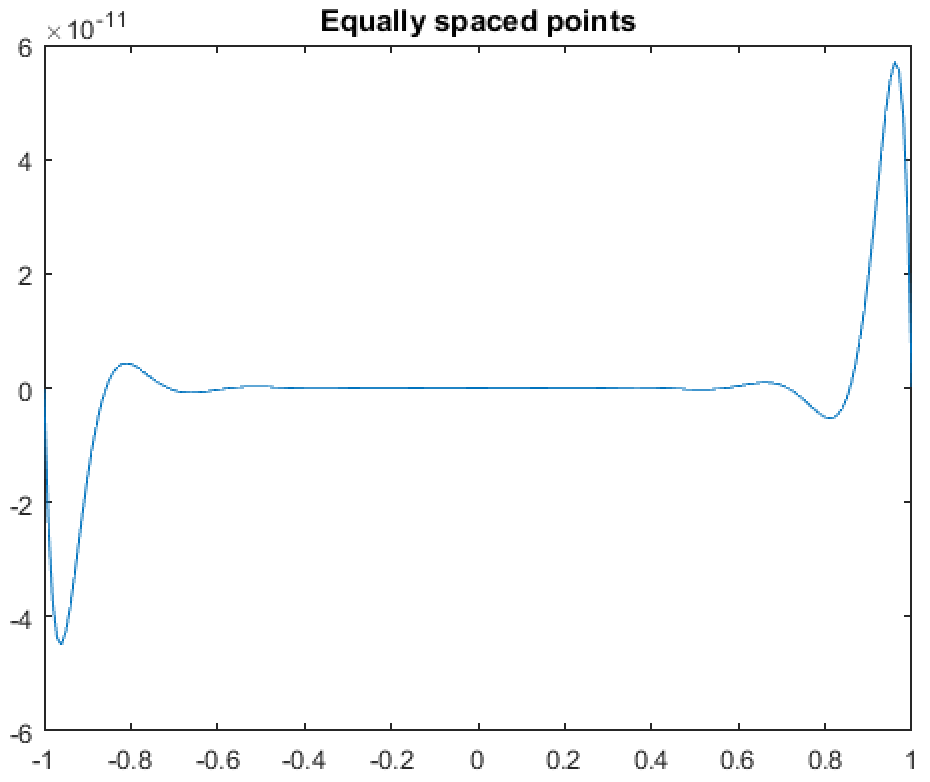
Problem #3 : Approximation Theory

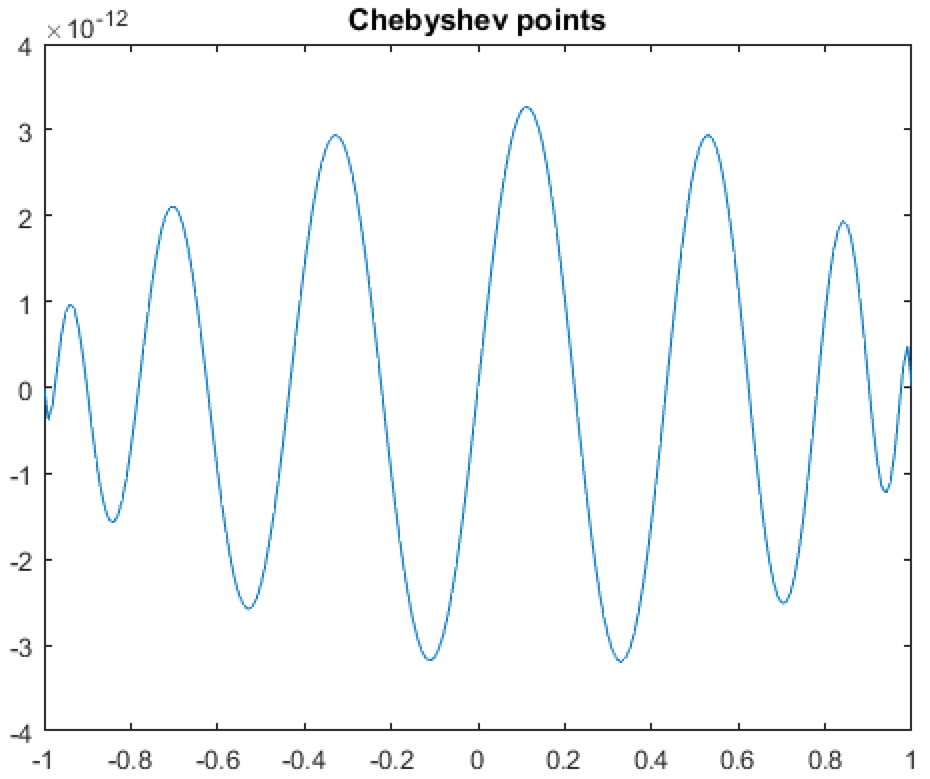
```
function prob3()

% List the coeficients of the 14th degree taylor expansion of e^(2x)
T =
 [1,2,2,4/3,2/3,4/15,4/45,88/315,2/315,4/2835,4/14175,4/467775,8/6081075,8/4256752

% Create a polynomial using the coefficients given and define the
% exact function.
x = linspace(-1,1,201);
p = polyval(T,x);
y = exp(2*x);

% calculate the difference between the polynomial and the exact
 function
% and plot the error.
err = p - y;
n1 = norm(err,1);
n2 = norm(err,2);
ninf = norm(err,inf);
figure
table(n1,n2,ninf)
plot(x,err)
title('Taylor expansion of e^(2x)')
```

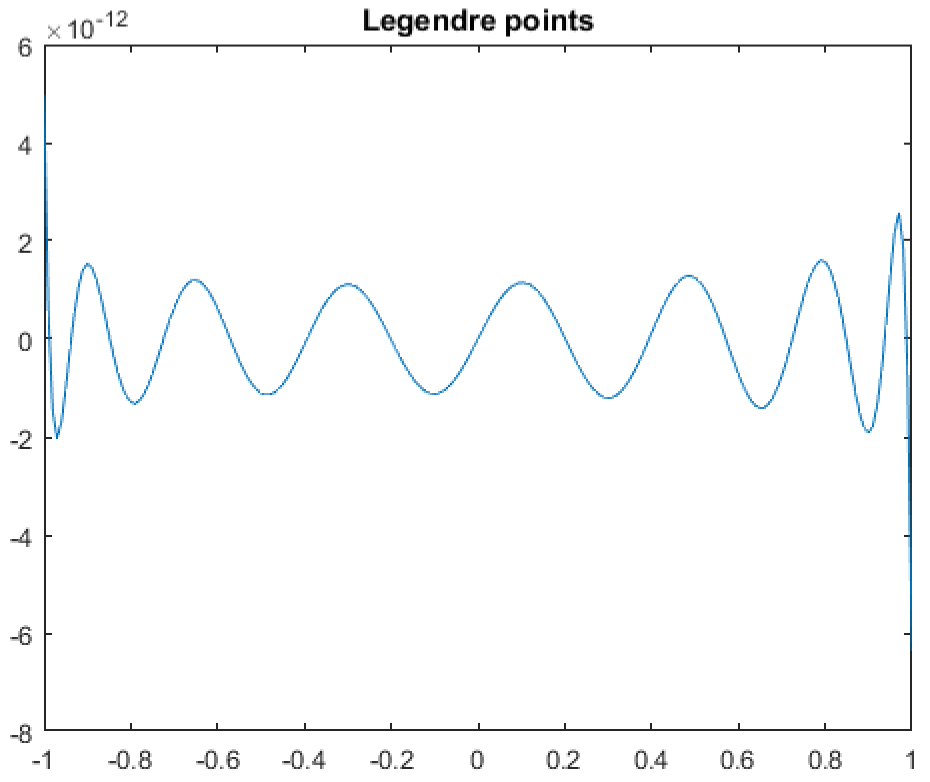
```
% part b

% Define equally spaced points, Chebyshev points, and legendre points
% within the range of -1 to 1.
i = 0:14;
xeq = -1 + 2*i./14;
xch = -cos(i*pi()./14);
xle = [-0.987992518020485 -0.394151347077563 0.570972172608539
 -0.937273392400706 -0.201194093997435 0.724417731360170
 -0.848206583410427 0 0.848206583410427 -0.724417731360170
 0.201194093997435 0.937273392400706 -0.570972172608539
 0.394151347077563 0.987992518020485];

% calculate the y values for each set of x and create polynomials to
% interpolate the points.
Teq = polyfit(xeq,exp(2*xeq),14);
Tch = polyfit(xch,exp(2*xch),14);
Tle = polyfit(xle,exp(2*xle),14);

% evaluate the created polynomials at each set of points
peq = polyval(Teq,x);
pch = polyval(Tch,x);
ple = polyval(Tle,x);

% plot the error for each method and report the norms in a table.
figure
erreq = peq - y;
n1eq = norm(erreq,1);
n2eq = norm(erreq,2);
ninfeq = norm(erreq,inf);
table(n1eq,n2eq,ninfeq)
plot(x,erreq)
title('Equally spaced points')

figure
errch = pch - y;
n1ch = norm(errch,1);
n2ch = norm(errch,2);
ninfch = norm(errch,inf);
table(n1ch,n2ch,ninfch)
plot(x,errch)
title('Chebyshev points')

figure
errle = ple - y;
n1le = norm(errle,1);
n2le = norm(errle,2);
ninfle = norm(errle,inf);
table(n1le,n2le,ninfle)
plot(x,errle)
title('Legendre points')

% part c
% The chebyshev and legendre points produce very small errors,
 legendre
% has a smaller norm1 and 2 but larger infinite norm.  I would select
% the chebyshev points to use however, as they were able to be built
% using a simple formula rather than typed manually.

% part d
% call the chebfun function to define the function to be interpolated
% and the remez function to find the best interpolation with 14
 degrees.
f = chebfun('exp(2*x)');
[r,errr] = remez(f,14);

% Plot the error of both the Chebyshev point interpolation and the
% chebfun interpolation.
figure
plot(x,errch,'b',x,errr*x,'r')
title('error in Chebyshev points versus chebfun interpolation')
legend('Chebyshev','Chebfun')
% This plot has some text overlap, I couldn't find a good correlary to
% python's tight_layout()


end
```

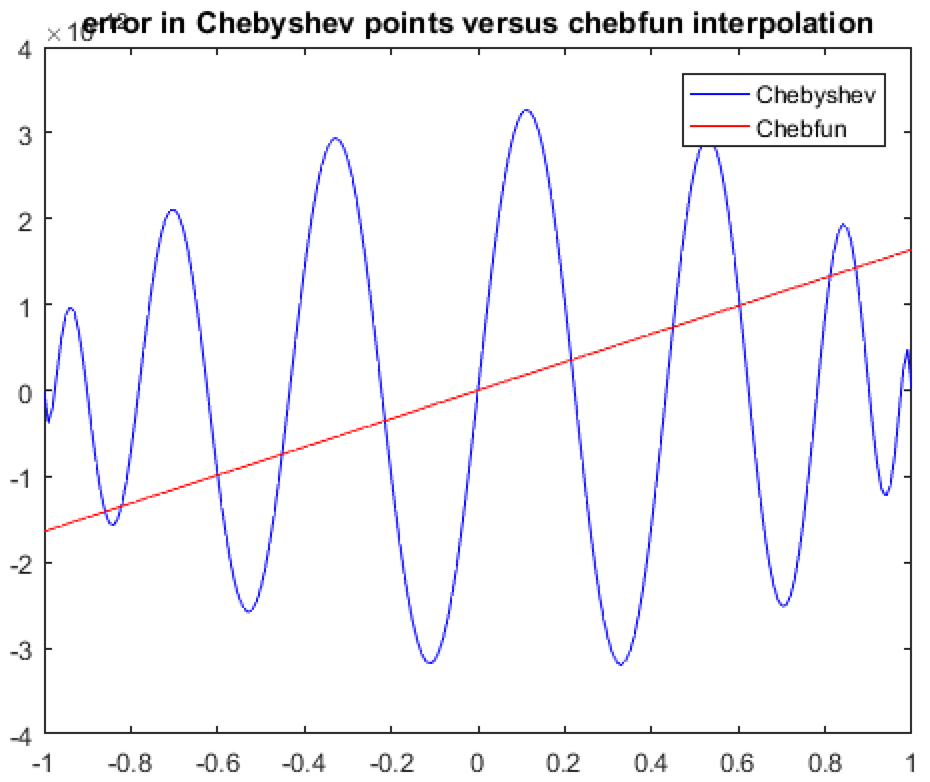
```
ans =

      n1        n2        ninf
    ______    ______    ______

    294.04    28.014    4.2182


ans =

       n1eq           n2eq           ninfeq
    __________     __________     __________

    8.9668e-10     1.7686e-10     5.6999e-11


ans =

      n1ch           n2ch          ninfch
    _________     __________     __________

    3.255e-10     2.6605e-11     3.2683e-12


ans =

       n1le           n2le           ninfle
    __________     __________     __________
    1.8024e-10    1.5836e-11    6.3682e-12

prob3 : Success!
```




Equally spaced points
×10^-11
6
4
2
0
-2
-4
-6
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


Chebyshev points
×10^-12
4
3
2
1
0
-1
-2
-3
-4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

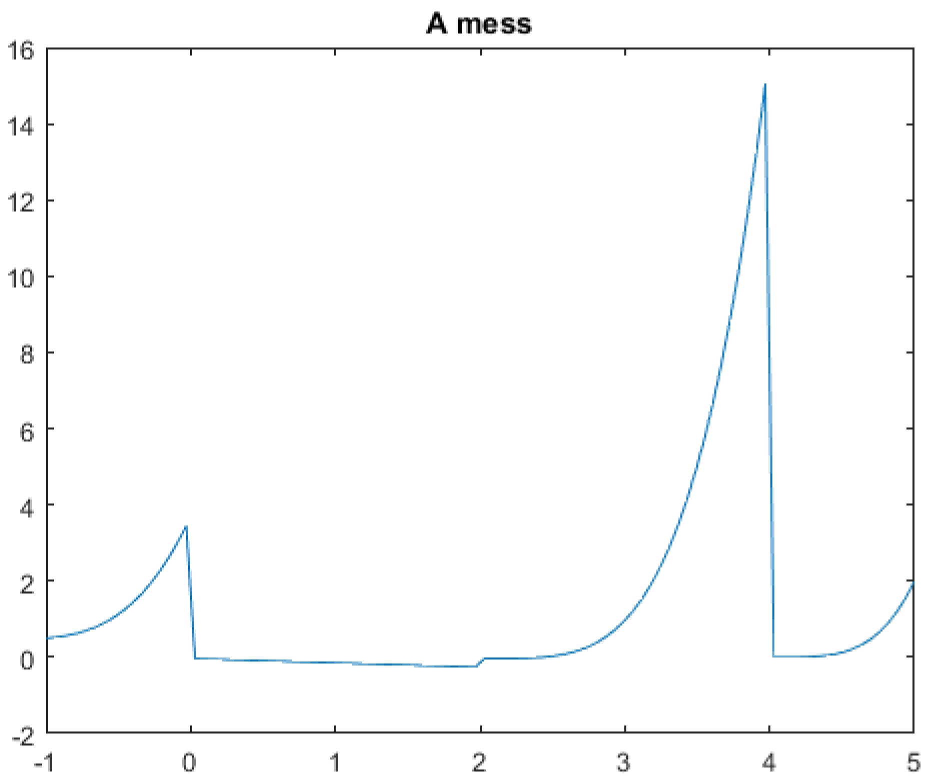

Legendre points
×10^-12
6
4
2
0
-2
-4
-6
-8
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1


error in Chebyshev points versus chebfun interpolation
×10^-12
Chebyshev
Chebfun
4
3
2
1
0
-1
-2
-3
-4
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1

Problem #4 : Piecewise Cubic Hermite Interpolation

```
function prob4()

% Declare x and y points
y = [0,0,1,2,2];
x = [-1,0,2,4,5];

% declare the values of d, c, and b for each interval
d = [0, 0 , 0, 19/22 , 0];
c = [0 , 0 , (1.5-19/22)/2 , (1.5-19/11)/2 , 0];
b = [0 , 0.5 , (-1+19/22)/4 , (19/22-1)/4 , 0];

% create a matrix of each value
g = [y(1:5)' d' c' b'];

% create a piecewise function
P = mkpp(x,g);

% Evaluate the piecewise function from -1 to 5 and plot it
x2 = linspace(-1,5);
y2 = ppval(x2,P);
plot(x2,y2)
title('A mess')

% These results are not correct, I do not understand how the mkpp
% function is working to create these results.

end
```

prob4 : Success!

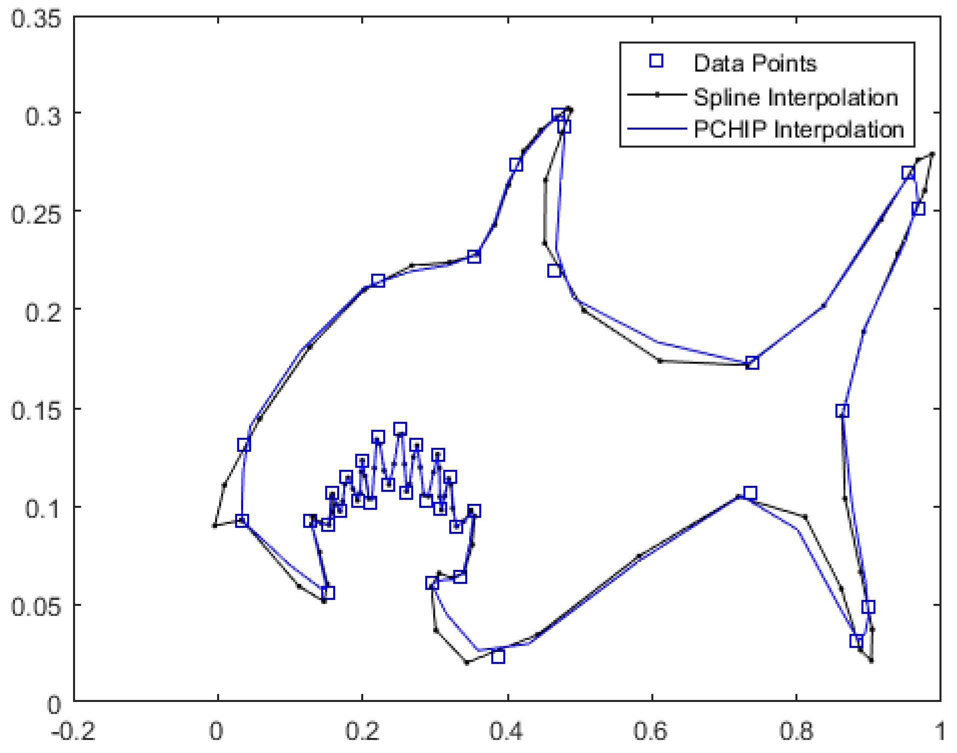


Problem #5 : Closed Curve Interpolation

```
function prob5()

% part a: we load our given data from the file
load shark;
plot(x,y,'k.-','MarkerSize',10);
t=1:length(x);
tt=linspace(t(1),t(end),101); %to parametrize our data

% part b

% Using Spline: create smooth parametric line
% We use spline on each data set x and y
xx=spline(t,x,tt);
yy=spline(t,y,tt);
px=pchip(t,x,tt);
py=pchip(t,y,tt);

% To plot our spline and pchip data against the data points:
plot(x,y,'bs',xx,yy,'k.-',px,py,'b-')
legend('Data Points','Spline Interpolation','PCHIP Interpolation')
% In part b we see that the plot produced has a slight kink near the
% ending and beginning points
```

```
%part c
x1=[x(end-1);x;x(2)]; % here we create new data points by correcting
 and adding the points after the end and the points before.
y1=[y(end-1);y;y(2)];
t1=1:length(x1); % again to parameterize
tt1=linspace(t1(1),t1(end),N+1);
% Now we run our new data through to see if it produces a better
 looking interpolation:
xx1=spline(t1,x1,tt1);
yy1=spline(t1,y1,tt1);
px1=pchip(t1,x1,tt1);
py1=pchip(t1,y1,tt1);
plot(x1,y1,'bs',px1,py1,'r-',xx1,yy1,'g-')
legend('Data Points','Corrected PCHIP Interpolation','Corrected Spline
 Interpolation')

% part d
% The representation I like best is that of the corrected PCHIP
% Interpolation. The shark's curves are much smoother and there is
 less of
% an overshoot on the nose.

% part e


end
```

```
prob5 : Something went wrong.
Undefined function or variable 'N'.
```

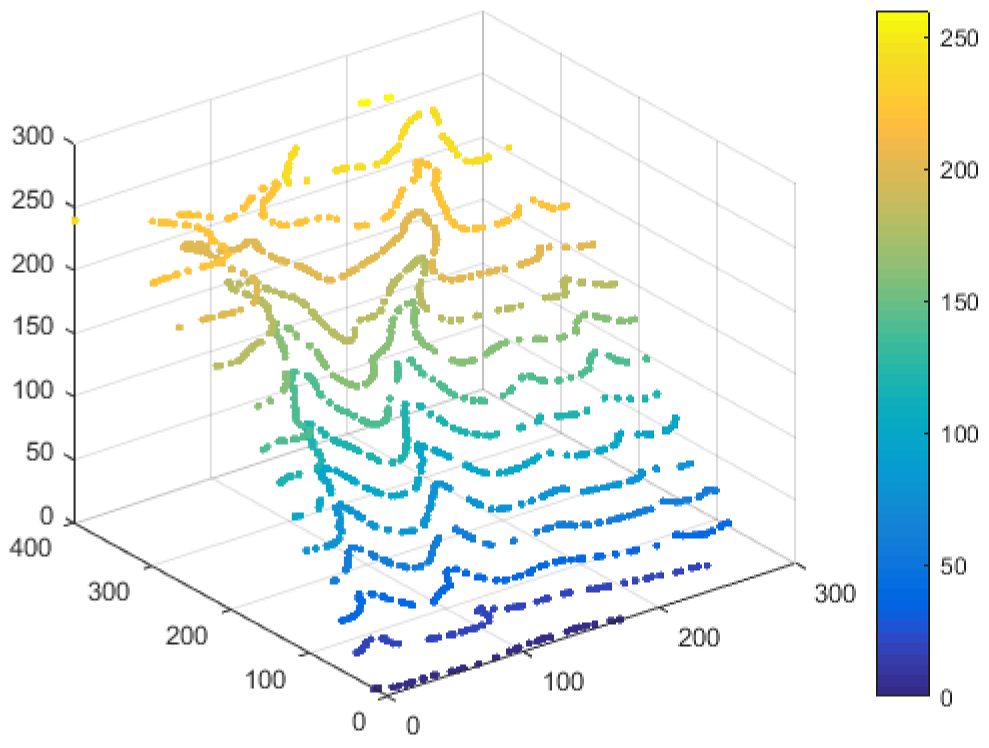


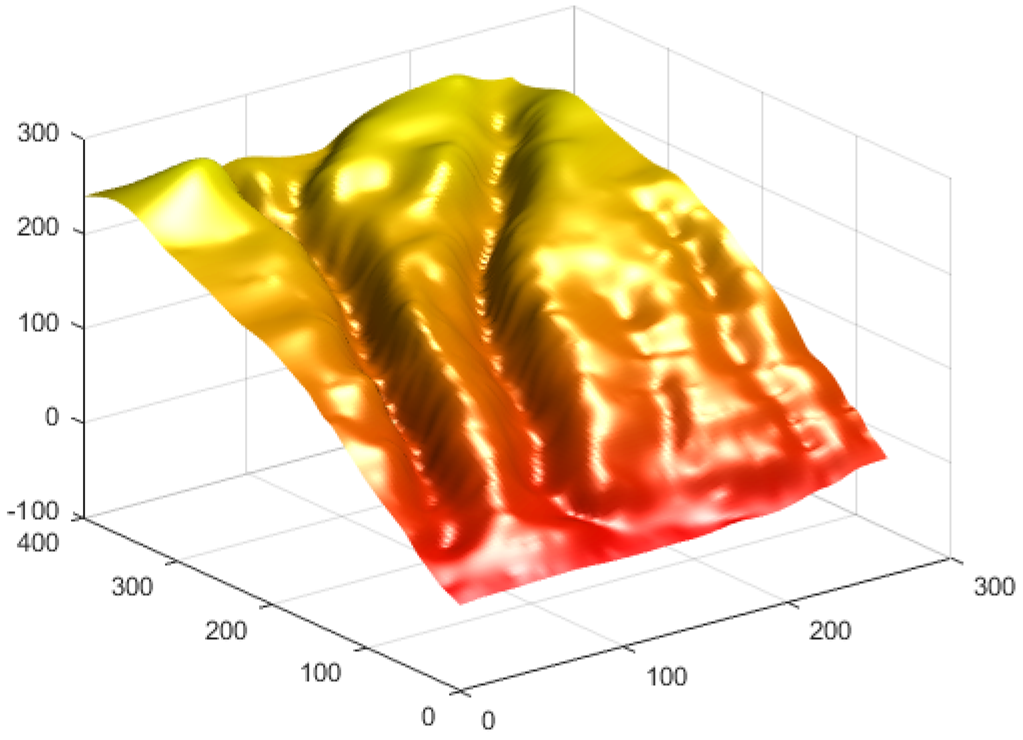
Problem #6 : Scattered Data Interpolation

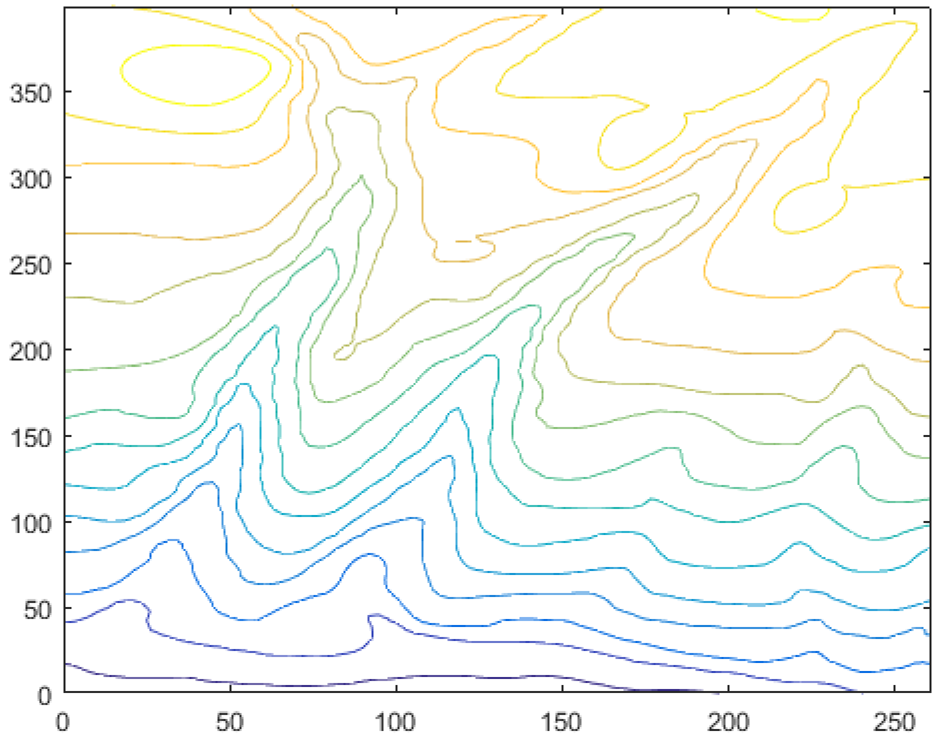
```
function prob6()

% Load data
load topodata;

% First figure, color coded scatter plot fo the data
figure
scatter3(x,y,z,40,z,'.');
colorbar

% call rbffit function
r = rbffit(x,y,z);

% generate a list of X and Y points
x1 = linspace(0,261,100);
y1 = linspace(0,399,153);

% Create a grid from the generated points
[X,Y] = meshgrid(x1,y1);

% Call rbval function to generate the matching Z points for the x and
 y points generated earlier.
Z = rbfval(r,x,y,X,Y);
```

```
% Second figure, a 3D surface map
figure
surf(X,Y,Z)
shading interp;
colormap(autumn);
lighting phong;
camlight right;

% define the contour lines to use.
v = 0:20:260;

% third figure, a contour plot.
figure
contour(X,Y,Z,v)

end
```

prob6 : Success!




300
200
100
0
-100
400
300
200
100
0
0
100
200
300


350
300
250
200
150
100
50
0
0
50
100
150
200
250