Stephanie Klumpe
Math 6680
HW 2

5.1.Modify Program 9 to compute and plot the maximum error over $[-1, 1]$ for equispaced and Chebyshev interpolation on a log scale as a function of N. What asymptotic divergence and convergence constants do you observe for these two cases? (Confine your attention to small enough values of N that rounding errors are not dominant.) Now, based on the potential theory of this chapter, determine exactly what these geometric constants should be. How closely do your numerical experiments match the theoretical answers? [1]

Here, we are choosing N to be multiples of 4 between 4 and 28.

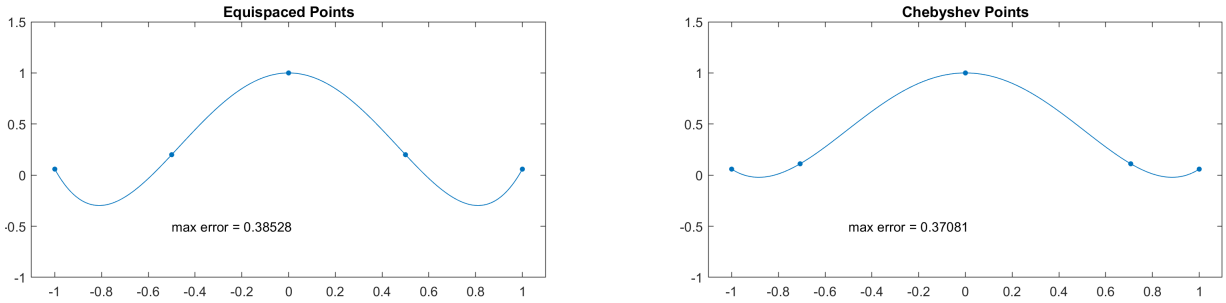


Figure 1: $N = 4$

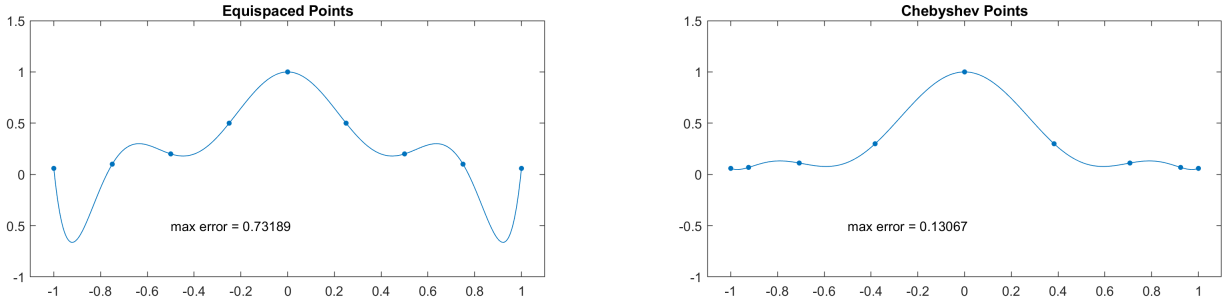


Figure 2: $N = 8$

[1] I accidentally did this problem and didn't want to delete all of the work, so I kept it in.

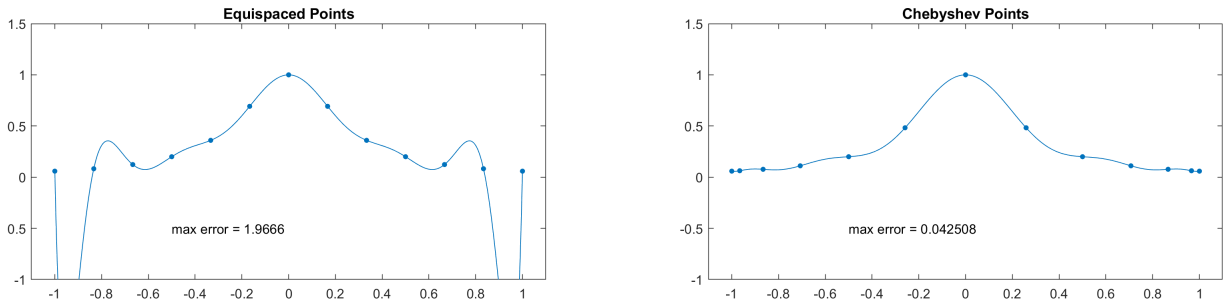


Figure 3: $N = 12$

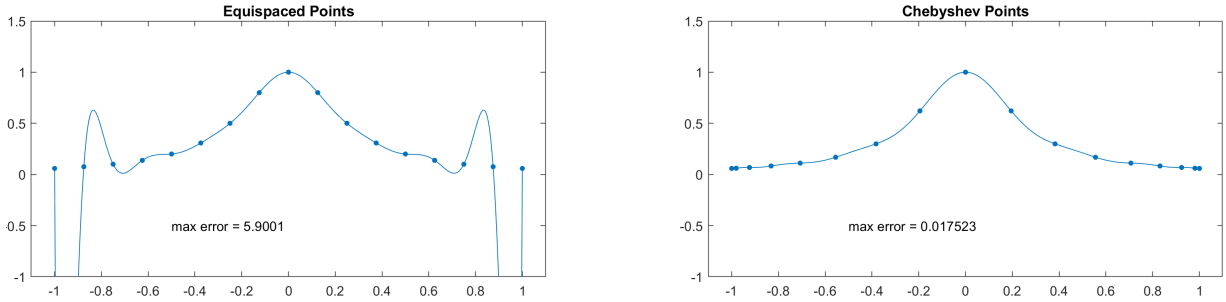


Figure 4: $N = 16$

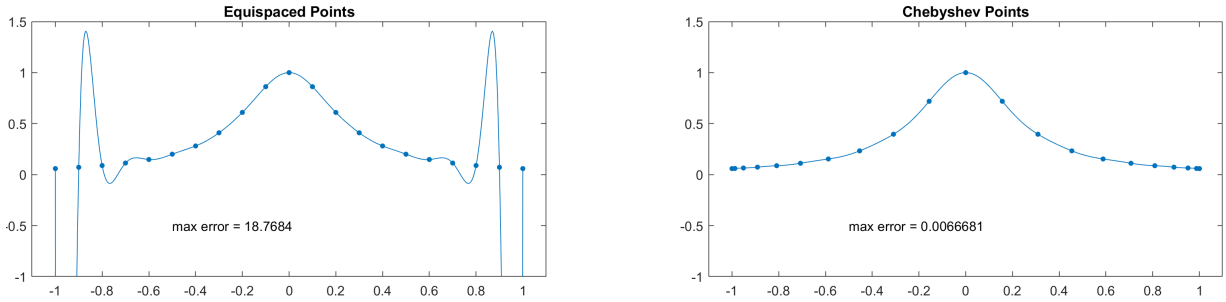


Figure 5: $N = 20$

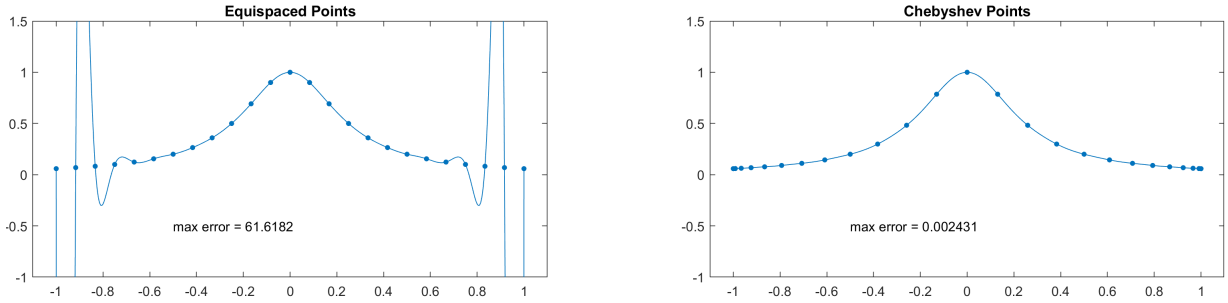


Figure 6: $N = 24$

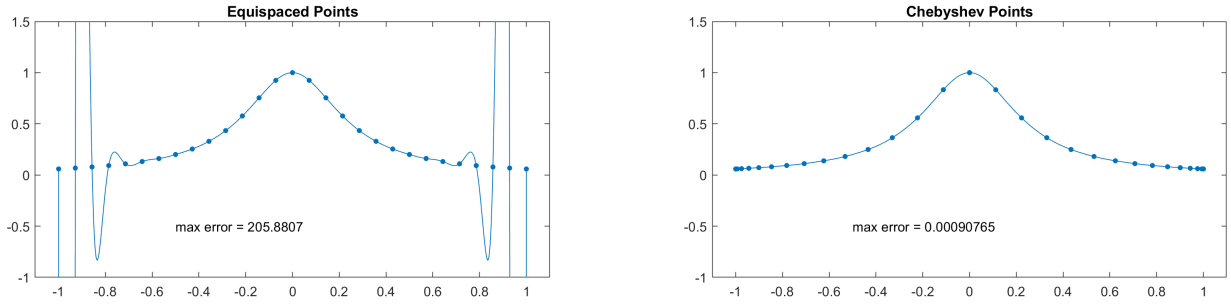
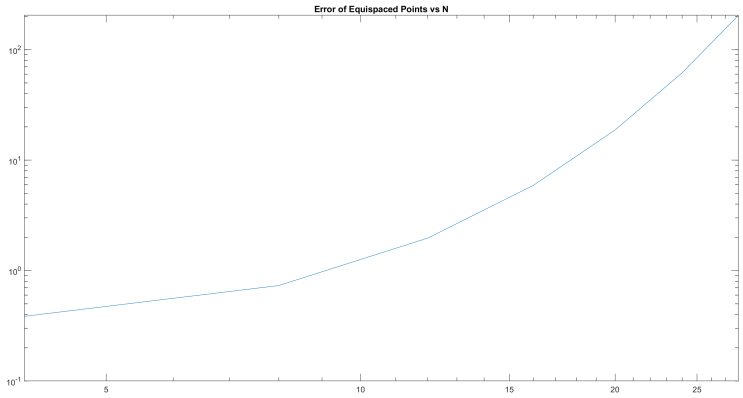
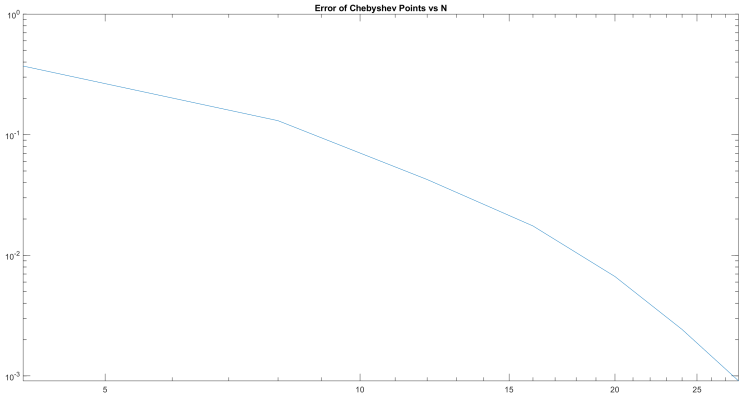


Figure 7: $N = 28$

Now, looking at the two plots of the error vs N, we see that for the equidistal points, there is no convergence. Rather, there is divergence towards infinity and this divergence happens exponentialy. Similarly, for the Chebyshev nodes, there is an exponential plotting, however, this is exponentially convergent.





```
% Stephanie Klumpe
% Problem 5.3 edited code

clear
close all
clc

error = zeros(2,7);       %% Set up an error matrix to hold the max errors
counter = 1;                    %% Counter used for indexing
xx = -1:.005:1; clf             %% x space
v=1; %% Used to split the above error matrix into the two different errors
Nval=[4 8 12 16 20 24 28];      %% N-value vector

 for N = Nval

  for i = 1:2

    if i==1, s = 'Equispaced Points'; x = -1 + 2*(0:N)/N; end
    if i==2, s = 'Chebyshev Points';  x = cos(pi*(0:N)/N); end
    subplot(2,2,i)
    u = 1./(1+16*x.^2);
    uu = 1./(1+16*xx.^2);
    p = polyfit(x,u,N);              % interpolation
    pp = polyval(p,xx);              % evaluation of interpolant
    plot(x,u,'.','markersize',13)
    hold off
    line(xx,pp)
    axis([-1.1 1.1 -1 1.5]), title(s)
    error(i,counter) = norm(uu-pp,inf);
    text(-.5,-.5,['max error = ' num2str(error(i,counter))])

  end

  err1=error(1:v,:);             %% Max error vector of equispaced
  err2=error(v+1:end,:);         %% Max error vector for the Cheb nodes
  figure          %% Opens new figure for each of the different N-values
  counter=counter+1;

 end

loglog(Nval,err1)
title('Error of Equispaced Points vs N')
figure
loglog(Nval,err2)
title('Error of Chebyshev Points vs N')
```

5.3. Let $E_N = \inf_p ||p(x) - e^x||_\infty$, where $||f||_\infty = \sup_{x\in[-1,1]} |f(x)|$, denote the error in degree N minimax polynomial approximation to e^x on $[-1, 1]$. (a) One candidate approximation $p(x)$ would be the Taylor series truncated at degree N. From this approximation, derive the bound $E_N < ((N + 2)/(N + 1))/(N + 1)!$ for $N \geq 0$. (b) In fact, the truncated Taylor series falls short of optimal by a factor of about 2^N, for it is known (see equation (6.75) of [Mei67]) that as $N \to \infty$, $E_N\ 2^{-N}/(N + 1)!$. Modify Program 9 to produce a plot showing this asymptotic formula, the upper bound of (a), the error when $p(x)$ is obtained from interpolation in Chebyshev points, and the same for equispaced points, all as a function of N for $N = 1, 2, 3, \ldots, 12$. Comment on the results.

We know that the Taylor series expansion of e^x about $x = 0$ is given by $\sum \frac{x^n}{n!}$. So, the truncated series is $1 + x + \frac{x^2}{2} + \ldots + \frac{x^N}{N!}$. Hence, $|p(x) - e^x| = |R_N(x)|$ where $R_N(x)$ is the remainder of the exact Taylor expansion of e^x. We also know that such an expansion exists as e^x is an analytic function. So,

$$E_N = \sup_{x\in[-1,1]} |\frac{1}{n!} \int_0^x e^t (x - t)^n dt|$$

using the integral form of the remainder. Now,

$$E_N = \sup_{x\in[-1,1]} \frac{1}{n!} |\int_0^x e^t (x - t)^n dt| \leq \sup_{x\in[-1,1]} \frac{1}{n!} \int_0^x |e^t (x - t)^n| dt$$

$$\leq \sup_{x\in[-1,1]} \frac{1}{n!} \int_0^x e|(x - t)^n| dt = \sup_{x\in[-1,1]} \frac{e}{n!} |\frac{x^n}{(n + 1)}| = \frac{e}{(n + 1)!}$$

```
% Stephanie Klumpe
% Problem 5.3 edited code

clear
close all
clc

N=16;
  xx = -1:.005:1; clf
  for i = 1:2
    if i==1, s = 'equispaced points'; x = -1 + 2*(0:N)/N; end
    if i==2, s = 'Chebyshev points';  x = cos(pi*(0:N)/N); end
    subplot(2,2,i)
    u = 1./(1+16*x.^2);
    uu = 1./(1+16*xx.^2);
    p = polyfit(x,u,N);              % interpolation
    pp = polyval(p,xx);              % evaluation of interpolant
    plot(x,u,'.','markersize',13)
    hold off
    line(xx,pp)
    axis([-1.1 1.1 -1 1.5]), title(s)
    error(N,i) = norm(uu-pp,inf);
    text(-.5,-.5,['max error = ' num2str(error(N,i))])
  end
```

6.1. If $x_0, x_1, \ldots, x_N \in \mathbb{R}$ are distinct, then the cardinal function $p_j(x)$ defined by

$$p_j(x) = \frac{1}{a_j} \prod_{k=0,\neq j}^{N} (x - x_k), \quad a_j \prod_{k=0,\neq j}^{N} (x_j - x_k)$$

is the unique polynomial interpolant of degree N to the values 1 at x_j and 0 at x_k, $k \neq j$. Take the logarithm and differentiate to obtain

$$p_j'(x) = p_j(x) \sum_{k=0,\neq j}^{N} (x - x_k)^{-1},$$

and derive the formulas

$$D_{ij} = \frac{1}{a_j} \prod_{k=0,\neq j}^{N} (x_i - x_k) = \frac{a_i}{a_j(x_i - x_j)} \quad (i \neq j)$$

and

$$D_{jj} = \sum_{k=0,\neq j}^{N} (x_j - x_k)^{-1}$$

Letting $p_j(x)$ and a_j be as given above, we take the natural logarithm of both sides. This gives us

$$\ln(p_j(x)) = \ln(\prod_{k=0,\neq j}^{N} \frac{(x - x_k)}{(x_j - x_k)}) = \sum_{k=0,\neq j}^{N} \ln(\frac{x - x_k}{x_j - x_k})$$

$$= \sum_{k=0,\neq j}^{N} \ln(x - x_k) - \sum_{k=0,\neq j}^{N} \ln(x_j - x_k)$$

Now differentiating both sides yields

$$\frac{p_j'(x)}{p_j(x)} = \sum_{k=0,\neq j}^{N} \frac{1}{(x - x_k)} - 0$$

which gives us the desired equation

$$p_j'(x) = p_j(x) \sum_{k=0,\neq j}^{N} (x - x_k)^{-1}$$

6.8. Let D_N be the usual Chebyshev differentiation matrix. Show that the power $(D_N)^{N+1}$ is identically equal to zero. Now try it on the computer for $N = 5$ and 20 and report the computed 2-norms $||(D_5)^6||_2$ and $||(D_{20})^{21}||_2$. Discuss.

Well,

7.1. Modify Program 13 so that instead of polyval and polyfit, it uses the more stable formula of barycentric interpolation

$$p(x) = \sum_{j=0}^{N} \frac{a_j^{-1} u_j}{x - x_j} \Big/ \sum_{j=0}^{N} \frac{a_j^{-1}}{x - x_j}$$

where $\{a_j\}$ are defined by (6.7). Experiment with various interpolation problems (such as that of Exercise 5.1) and find evidence of the enhanced stability of this method.

Well,

```
% Stephanie Klumpe
% Problem 7.1 edited code

clear
close all
clc

  N = 16;
  [D,x] = cheb(N);
  D2 = D^2;
  D2 = D2(2:N,2:N);                    % boundary conditions
  f = exp(4*x(2:N));
  u = D2\f;                            % Poisson eq. solved here
  u = [0;u;0];
  clf, subplot('position',[.1 .4 .8 .5])
  plot(x,u,'.','markersize',16)
  xx = -1:.01:1;
  uu = polyval(polyfit(x,u,N),xx);     % interpolate grid data
  line(xx,uu)
  grid on
  exact = ( exp(4*xx) - sinh(4)*xx - cosh(4) )/16;
  title(['max err = ' num2str(norm(uu-exact,inf))],'fontsize',12)

  %inva = 1./prod(x_j-x_k)
  %uu=

% M=length(data);     N=length(x);
% Compute the barycentric weights
% X=repmat(data(:,1),1,M);
% matrix of weights
% W=repmat(1./prod(X-X.'+eye(M),1),N,1);
% Get distances between nodes and interpolation points
% xdist=repmat(x,1,M)-repmat(data(:,1).',N,1);
% Find all of the elements where the interpolation point is on a node
% [fixi,fixj]=find(xdist==0);
% Use NaNs as a place-holder
% xdist(fixi,fixj)=NaN;
% H=W./xdist;
% Compute the interpolated polynomial
% p=(H*data(:,2))./sum(H,2);
% Replace NaNs with the given exact values.
% p(fixi)=data(fixj,2);
```

7.2. Solve the boundary value problem $u_{xx} + 4u_x + e^x u = \sin(8x)$ numerically on $[-1, 1]$ with boundary conditions $u(\pm 1) = 0$. To ten digits of accuracy, what is $u(0)$?

Well,

```
% Stephanie Klumpe
% Problem 7.2

clear
close all
clc

xx = -1:.01:1;
u=zeros(size(xx));
u(-1)=0;
u(1)=0;
```

8.3. Modify Program 12 (p. 57) to make use of chebfft instead of cheb. The results should be the same as in Output 12, except for rounding errors. Are the effects of rounding errors smaller or larger than before?

Below are the graph outputs of program 12, the first of which is using cheb and the second using chebfft:

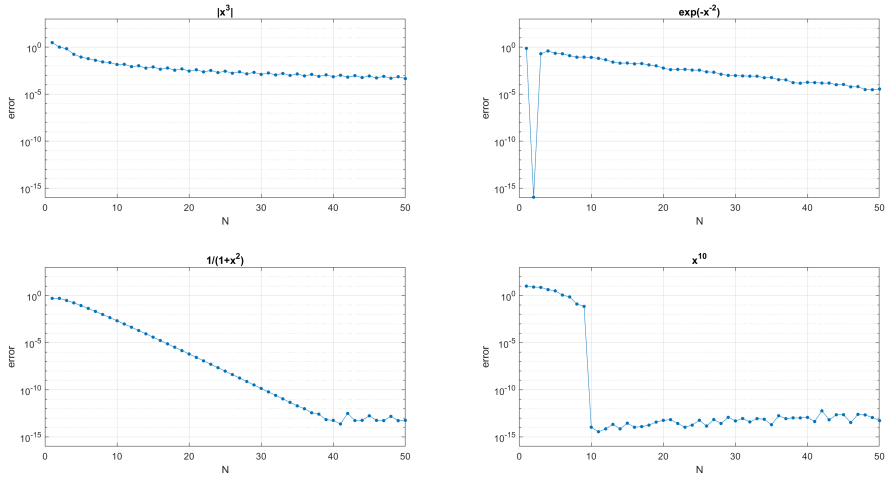


Figure 8: cheb

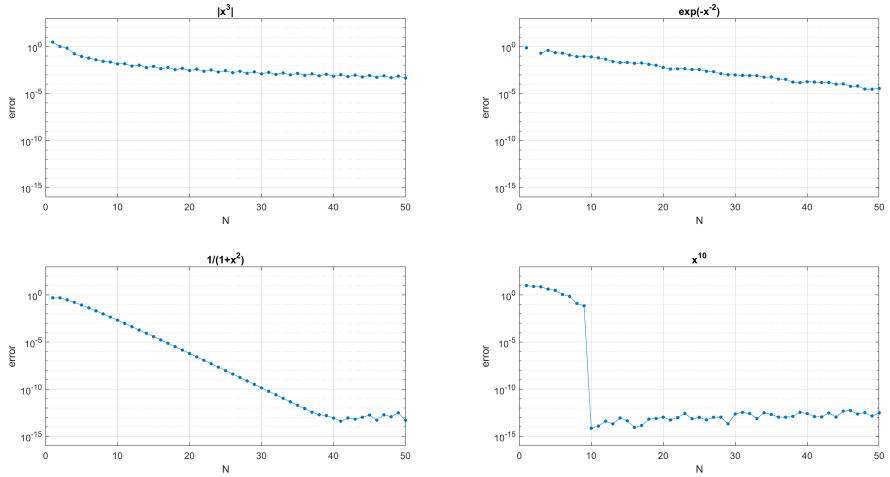


Figure 9: chebfft

We see that the round off error is the only difference between the two graphs. It can also be seen that the rounding error for the chebfft function is slightly less than that of the regular cheb function.

```
% Stephanie Klumpe
% Problem 8.3 edited code

clear
close all
clc

% Compute derivatives for various values of N:
  Nmax = 50; E = zeros(3,Nmax);
  for N = 1:Nmax
    [D,x] = cheb(N);
    v = abs(x).^3; vprime = 3*x.*abs(x);      % 3rd deriv in BV
    E(1,N) = norm(chebfft(v)-vprime,inf);
    v = exp(-x.^(-2)); vprime = 2.*v./x.^3;  % C-infinity
    E(2,N) = norm(chebfft(v)-vprime,inf);
    v = 1./(1+x.^2); vprime = -2*x.*v.^2;     % analytic in [-1,1]
    E(3,N) = norm(chebfft(v)-vprime,inf);
    v = x.^10; vprime = 10*x.^9;              % polynomial
    E(4,N) = norm(chebfft(v)-vprime,inf);
  end                                  %% We replaced the D*v in the norm
                                        % computation with chebfft(v)

% Plot results:
  titles = {'|x^3|','exp(-x^{-2})','1/(1+x^2)','x^{10}'}; clf
  for iplot = 1:4
    subplot(2,2,iplot)
    semilogy(1:Nmax,E(iplot,:),'.','markersize',12)
    line(1:Nmax,E(iplot,:))
    axis([0 Nmax 1e-16 1e3]), grid on
    set(gca,'xtick',0:10:Nmax,'ytick',(10).^(-15:5:0))
    xlabel N, ylabel error, title(titles(iplot))
  end
```

8.5. Write a code chebfft2 for second-order differentiation by the FFT, and show by examples that it matches the results obtained by matrices, apart from rounding errors.

Here, we used the functions $\sin(x), e^x$, and e^{-x^2} as examples to demonstrate the chebfft2 function. In order to write this function, all we needed to do is repeat the FFT, inverse FFT, and deriving the algebraic polynomial steps just on the derivative vector. So, we get the following graphs.

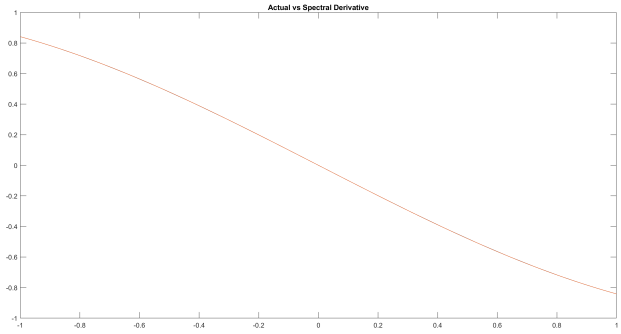


Figure 10: $\sin(x)$

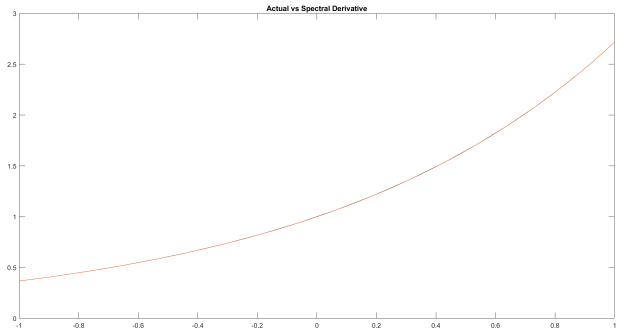


Figure 11: e^x

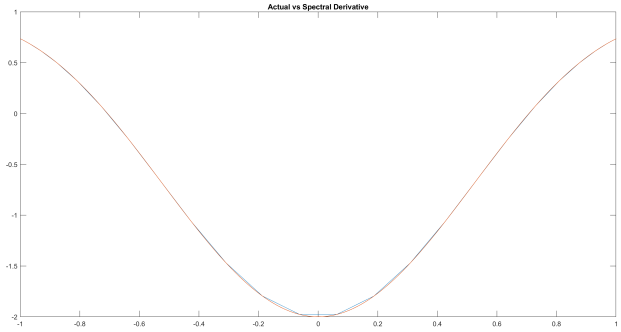


Figure 12: e^{-x^2}

And we see that the matrices also provide the same outputs baring the roundoff errors.

```
% Stephanie Klumpe
% Problem 8.5
% chebfft2

clear
close all
clc

t=-1:0.001:1;
[D,x]=cheb(25);                                % Compute Cheb points
v=sin(x);                                      % Choose a function
%v=exp(x);
%v=exp(-x.^2);
d2v=-sin(t);                                   % Define the second derivative
%d2v=exp(t);
%d2v=exp(-t.^2).*(4.*t.^2-2);
dw=chebfft2(v);                                % Call the new function
plot(x,dw)                                     % Plot the actual deriv and the
hold on                                        % spectral deriv to compare
plot(t,d2v)
title('Actual vs Spectral Derivative')



  function dw = chebfft2(v)
  N = length(v)-1; if N==0, w=0; return, end
  x = cos((0:N)'*pi/N);
  ii = 0:N-1;
  v = v(:); V = [v; flipud(v(2:N))];      % transform x -> theta
  U = real(fft(V));
  W = real(ifft(1i*[ii 0 1-N:-1]'.*U));
  w = zeros(N+1,1);
  w(2:N) = -W(2:N)./sqrt(1-x(2:N).^2);    % transform theta -> x
  w(1) = sum(ii'.^2.*U(ii+1))/N + .5*N*U(N+1);
  w(N+1) = sum((-1).^(ii+1)'.*ii'.^2.*U(ii+1))/N + ...
               .5*(-1)^(N+1)*N*U(N+1);
  dV=[w; flipud(w(2:N))];                  % Repeat the above process
  dU=real(fft(dV));                        % by defininig a vector to be
  dW = real(ifft(1i*[ii 0 1-N:-1]'.*dU)); % the derivative vector. Then
  dw = zeros(N+1,1);                       % compute the spectral deriv.
  dw(2:N) = -dW(2:N)./sqrt(1-x(2:N).^2);
  dw(1) = sum(ii'.^2.*dU(ii+1))/N + .5*N*U(N+1);
  dw(N+1) = sum((-1).^(ii+1)'.*ii'.^2.*dU(ii+1))/N + ...
               .5*(-1)^(N+1)*N*dU(N+1);
  end

% From this, we can say that if we repeate the same idea as above
% n times, we can get the nth derivative using Cheb spectral diff via FFT.
```