12.2 Write a MATLAB program to implement (6.8) and (6.9) and construct the differentiation matrix D_N associated with an arbitrary set of distinct points $x_0, \ldots, x_N$. Combine it with gauss to create a function that computes the matrix D_N associated with Legendre points in $(-1, 1)$. Print results for $N = 1, 2, 3, 4$.

Solution.
Implementing the formulas for (6.8) and (6.9) was not too difficult as it took the a^{-1} code from the Baycentric Interpolation code. After that, it took the same for and if loops to build the differentiation matrix. Running the code through $N = 1, 2, 3, 4$ gives the matrices

$$D_1 = 0$$

$$D_2 = \begin{pmatrix} -0.8660 & 0.8660 \\ -0.8660 & 0.8660 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} -0.6455 & 2.5820 & -0.6455 \\ -0.6455 & -1.2910 & 0.6455 \\ 0.6455 & -2.5820 & 1.2910 \end{pmatrix}$$

$$D_4 = \begin{pmatrix} -0.5806 & 4.8602 & -2.1088 & 0.5806 \\ -0.7576 & -0.8326 & 1.4707 & -0.3287 \\ 0.3287 & -1.4707 & -1.9188 & 0.7576 \\ -0.5806 & 2.1088 & -4.8602 & 1.9188 \end{pmatrix}$$

```
% Stephanie Klumpe
% Problem 12.2

clear
close all
clc

Nvals = [1 2 3 4];                  % Setting the desired N values to ...
    loop thru

for N = Nvals

[x,w] = gauss(N);
xx = -1:0.01:1;
u = exp(4*x);

ainv = ones(N,1);                   % Inititalize a^-1 vector
Dij = zeros(N,N);                   % Inititalize the diagonal and ...
    nondiagonal
Dii = zeros(N,N);                   % matrices

for i = 1:N
    for j = 1:N
        if j ≠ i
            ainv(i) = ainv(i)*(x(i) - x(j));  % Fill out the a^-1 ...
                vector
        end
    end
end

for i = 1:N
    for j = 1:N
        if j != i
           Dij(i,j) = ainv(i)/ainv(j)*(1/(x(i)-x(j)));
           Dii(i,i) = 1/(x(i)-x(j));  % Compute the entries of the ...
               matrix
        end
    end
end

Dn = Dij +Dii;                       % Combine the two matrices to get ...
    the actual
                                     % diff matrix
fprintf('D%d = \n',N);
disp(Dn)
fprintf('\n')
end
```

12.7 Use the FFT in N points to calculate the first 20 Taylor series coefficients of $f(z) = \log(1 + \frac{1}{2}z)$. What is the asymptotic convergence factor as $N \to \infty$? Can you explain this number?

Solution.
To get the first 20 coefficients of the Talor series expansion, we need to spectrally differentiate using the FFT. Now, we will consider the series centered about $x = 0$. By this, we know that the first coefficient is given by $a_0 = f(0) = 0$. As such, we only need the first nineteen derivatives. Using differential calculus, we have that the *nth* derivative of f is given by:

$$f^{(n)} = \frac{(-1)^{n+1}}{(x+2)^n}$$

Hence, the *nth* coefficient of the desired Taylor series is given by

$$a_n = \frac{1}{n!}\left(\frac{(-1)^{n+1}}{2^n}\right)$$

The code below gives us the following coefficients for $N = 30$ using the FFT function:

$$a_1 = 0.5000 \;\; a_2 = -0.1250 \;\; a_3 = 0.04167 \;\; a_4 = -0.01563 \;\; a_5 = 0.00625$$

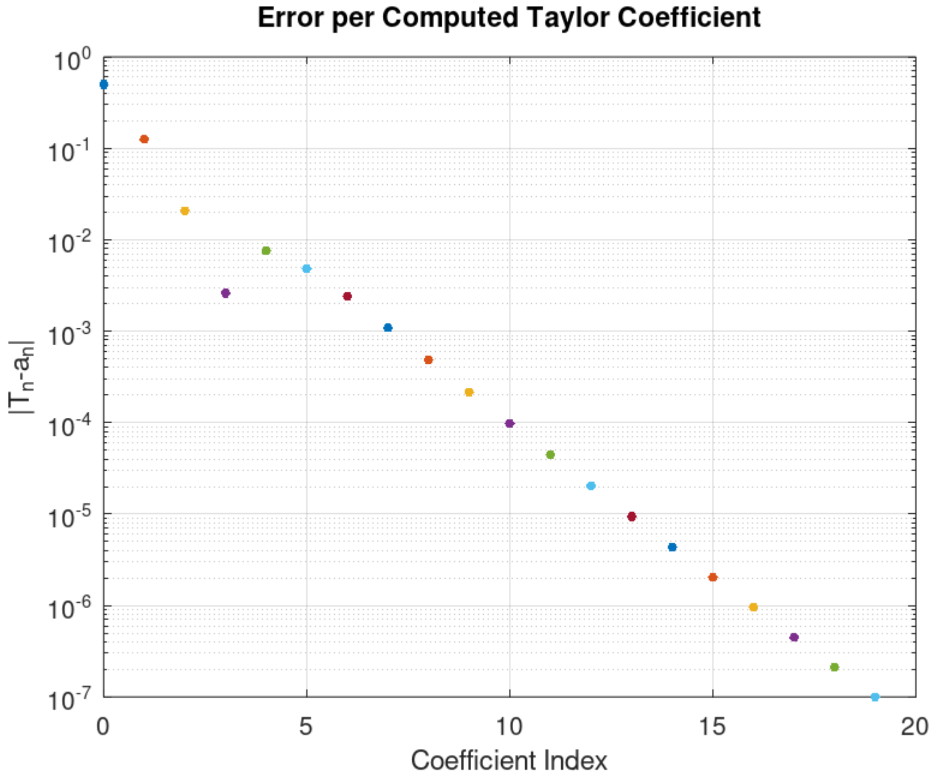
$$a_6 = -0.002604 \;\; a_7 = 0.001116 \;\; a_8 = -4.8828 \times 10^{-4} \;\; a_9 = 2.1701 \times 10^{-4} \;\; a_{10} = -9.7656 \times 10^{-5}$$

$$a_{11} = 4.4389 \times 10^{-5} \;\; a_{12} = -2.0345 \times 10^{-5} \;\; a_{13} = 9.3900 \times 10^{-6} \;\; a_{14} = -4.3597 \times 10^{-6}$$

$$a_{15} = 2.0345 \times 10^{-6} \;\; a_{16} = -9.5367 \times 10^{-7} \;\; a_{17} = 4.4879 \times 10^{-7} \;\; a_{18} = -2.1193 \times 10^{-7}$$

$$a_{19} = 1.0039 \times 10^{-7}$$

From here, we get the following plot of the error between the computed coefficients and the actual coefficients.



Since this is a semilog plot with respect to y, we see that the error is exponentially convergent as N tends towards infinity.

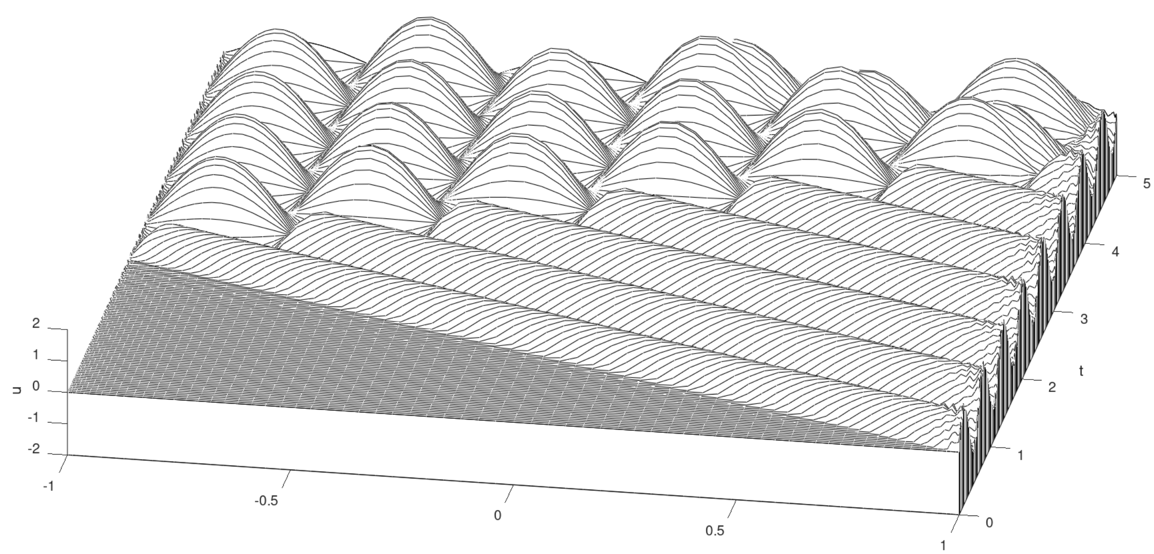
```
% Stephanie Klumpe
% Problem 12.7

clear
close all
clc

N = 30
deg = 20;      % Degree of the desired expansion
realco = zeros(length(deg));
error = zeros(length(deg));

z = @(theta) exp(1i*theta);    % Imaginary part
f = @(z) log(1+z./2);     % Desired function

t = 2*pi*(0:N-1)/N;
rho = z(t);

tayco = real(fft(f(rho),N)/length(rho));

xx=linspace(-6,6);

fxn = log(1+xx./2);   % Plot the original function

disp(tayco(2:deg))    % Display the 19 coefficients

fprintf('\n\n')
fprintf('T(x) = ')

for i = 1:deg
  fprintf('%ex^%d + ',tayco(i),i-1) % Display the found taylor series
end

for j = 1:deg
 realco(j) = (1/factorial(j))*((-1)^j+1/2^j);
  error(j)=abs(realco(j)-tayco(j));

  semilogy(j-1,error(j),'markersize',12)
  grid on
  hold on
end

title('Error per Computed Taylor Coefficient');
xlabel('Coefficient Index')
ylabel('|T_n-a_n|')

print('-dpng', 'problem12_7.png')
```

13.3 Modify Program 19 (p. 82) to solve $u_{tt} = u_{xx}$ as before but with initial and boundary conditions

$$u(x,0) = 0, \;\; u_x(-1,t) = 0, \;\; u(1,t) = \sin(10t)$$

Produce an 'attractive' plot of the solution for $0 \leq t \leq 5$.

Solution.
In the modification of program 19, we switch over to using the second order Chebyshev differentiation matrix instead of chebfft. This allowed for easier setup of the Neumann boundary condition at $x = -1$. After that, we also needed to define our time steps so that we could evaluate $\sin(10t)$ at the other boundary $x = 1$. The majority of the rest of the code remained the same with some minor numeric changes. As such, we get the following plot, which if I do say so my self, looks attractive.



```
% Stephanie Klumpe
% Problem 13.3 edited code

clear
close all
clc

  N = 80;
  [D,cx] = cheb(N);            % Initialize eveerything
  D2 = D^2;

  D2(N+1,:) = D(N+1,:);        % Set up Neumann BC at x=-1

  x = cx';
  dt = 8/N^2;                  % More initializing
  tmax = 5;
  tplot = .025;

  v = zeros(size(x));          % Define the intitial condition v(0,x)=0
  vold = zeros(size(x-dt));

  plotgap = round(tplot/dt);
  dt = tplot/plotgap;
  nplots = round(tmax/tplot);
  plotdata = [v; zeros(nplots,N+1)];
  tdata = 0;
  clf, drawnow,

  for i = 1:nplots
    t = dt*i*plotgap;       % Define the time steps used for the ...
        left BC

    for n = 1:plotgap
      w = (D2*v')';         % Use cheb matrix not chebfft for easier BC
      w(1) = 0;
      w(N+1) = 0;

      vnew = 2*v - vold + dt^2*w;
      vold = v;
      v = vnew;

      v(1) = sin(10*t);         % Dirichlet BC at x=1

    end
    plotdata(i+1,:) = v;
    tdata = [tdata; dt*i*plotgap];

  end

% Plot results:
  clf, drawnow, waterfall(x,tdata,plotdata)
  axis([-1 1 0 tmax -2 2]), view(10,70), grid off
  colormap(1e-6*[1 1 1]); ylabel t, zlabel u,
```

13.4 The time step in Program 37 is specified by $\Delta t = 5/(N_x + N_y^2)$. Study this discretization theoretically and, if you like, numerically, and decide: Is this the right choice? Can you derive a more precise stability limit on Δt?

Solution.
Taking a look at program 37, the implemented time step is not necessarily bad. Numerically speaking, it works well in terms of computation time as well as stability. Now, we know that

$$\Delta t \leq \frac{6}{N_x}$$

due to the length of the x spacial dimension and the discretization using Fourier nodes, and

$$\Delta t \leq \frac{8}{N_y^2}$$

due to the discretization using CHebyshev nodes in y. So, theoretically we only need $\Delta t \leq \min\{\frac{6}{N_x}, \frac{8}{N_y^2}\}$. Implementing this value in the p37 script, we still have stability. So, clearly the current Δt value is sufficient. However, for clearity and conciseness, we can let $\Delta t = \frac{1}{(N_x+N_y)^2}$ as this value is less than the theoretical minimum.

14.1 Determine the first five eigenvalues of the problem

$$u_{xxxx} + u_{xxx} = \lambda u_{xx}, \quad u(\pm 2) = u_x(\pm 2) = 0, \quad -2 < x < 2$$

and plot the corresponding eigenvectors.

Solution.
Using the code developed below, we have the following five eigenvalues:

$$e_1 = -3.5268$$

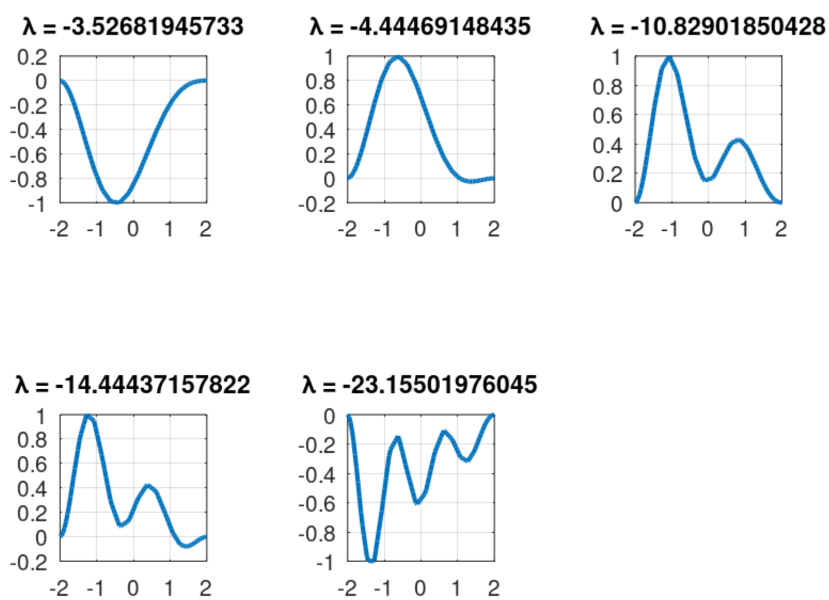
$$e_2 = -4.4447$$

$$e_3 = -10.8290$$

$$e_4 = -14.4444$$

$$e_5 = -23.1561$$

As such, we have the following plot of the eigenvectors:



```
% Stephanie Klumpe
% Problem 14.1

clear
close all
clc

N = 25;                    % Set up the nodes
[D,x] = cheb(N);

S = diag([0; 1 ./(1-x(2:N).^2); 0]);

D2=D^2;
D2 = D2(2:N,2:N);
                        % Define and reshape our differential matrices
D3 = (diag(1-x.^2)*D^3 - 6*diag(x)*D^2 - 6*D)*S;
D4 = (diag(1-x.^2)*D^4 - 8*diag(x)*D^3 - 12*D^2)*S;

D3 = D3(2:N,2:N);
D4 = D4(2:N,2:N);

LHS = (1/16)*D4 + (1/8)*D3;
RHS = (1/4)*D2;     % Get the left and right sides of the ODE

[vect, lam] = eig(LHS, RHS);
[evals, ii] = sort(diag(lam), 'descend');
ii = ii(1:5);          % Get the evals/evects and order them
evects = real(vect(:, ii));

figure
disp(evals(1:5))                 % Display the evals
xx = linspace(-2, 2);

%Plot the evects
for i = 1:5
    subplot(2, 3, i)
    plot(xx, interp1(2*x, [0; evects(:, i); 0], xx), 'linewidth', 2)
    grid on
    axis square
    title(['\lambda = ' num2str(evals(i), '%15.11f')]);
end

print('-dpng', 'problem14_1_evects.png')
```