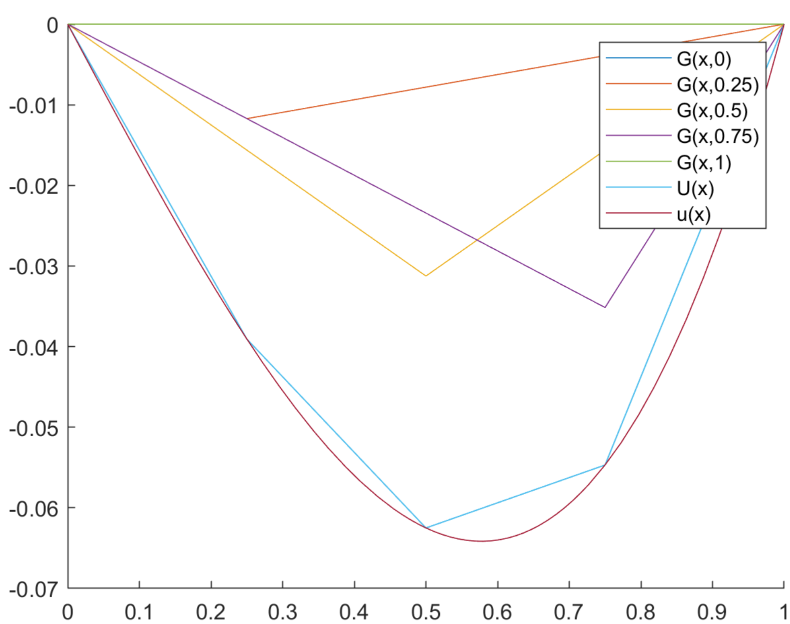
2.1 a. Write out the 5×5 matrix A from (2.43) for the boundary value problem $u''(x) = f(x)$ with $u(0) = u(1) = 0$ for $h = 0.25$.
b. Write out the 5×5 inverse matrix A^{-1} explicitly for this problem.
c. If $f(x) = x$, determine the discrete approximation to the solution of the boundary value problem on this grid and sketch this solution and the five Green's functions whose sum gives this solution.

Solution.
We have that

$$A = 16 \begin{pmatrix} 0.0625 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0.0625 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & -32 & 16 & 0 & 0 \\ 0 & 16 & -32 & 16 & 0 \\ 0 & 0 & 16 & -32 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

By MATLAB, we have that

$$A^{-1} = \frac{1}{16} \begin{pmatrix} 16 & 0 & 0 & 0 & 0 \\ 12 & -0.75 & -0.5 & -0.25 & 4 \\ 8 & -0.5 & -1 & -0.5 & 8 \\ 4 & -0.25 & -0.5 & -0.75 & 12 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{3}{64} & -\frac{1}{32} & -\frac{1}{64} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{32} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{64} & -\frac{1}{32} & -\frac{3}{64} & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



2.2 a. Find the function $G(x, \bar{x})$ solving $u''(x) = \delta(x - \bar{x}), u'(0) = 0, u(1) = 0$ and the functions $G_0(x)$ solving $u''(x) = 0, u'(0) = 1, u(1) = 0$ and $G_1(x)$ solving $u''(x) = 0, u'(0) = 0, u(1) = 1$.
b. Using this as guidance, find the general formulas for the elements of the inverse of the matrix in equation (2.54). Write out the 5×5 matrices A and A^{-1} for the case $h = 0.25$.

Solution.
Since we want $G(x, \bar{x})$ to be the solution to $u''(x) = \delta(x - \bar{x})$ with $u'(0) = 0, u(1) = 0$ we integrate $u''(x)$ to get

$$G'(x, \bar{x}) = \begin{cases} 0 & 0 \leq x < \bar{x} \\ 1 & \bar{x} < x <\leq 1 \end{cases}$$

Since $0 \leq \bar{x} \leq 1$, we have that $u'(0) = 0$. So, integrating again yields

$$G(x, \bar{x}) = \begin{cases} \bar{x} - 1 & 0 \leq x < \bar{x} \\ x - 1 & \bar{x} < x \leq 1 \end{cases}$$

Here we see that $G(1, \bar{x}) = u(1) = 0$. Thus, we have found the solution to the boundry value problem. Now, to find $G_0(x)$, we will integrate $u''(x) = 0$. This gives us $G_0'(x) = 1$. Clearly, $G_0'(0) = u'(0) = 1$. Integrating again gives us $G_0(x) = x + C$. Hence, $G_0(1) = 1 + C = 0$. So, $G_0(x) = x - 1$. So, we have found the solution to the boundry value problem $u''(x) = 0, u'(0) = 1, u(1) = 0$. Finally, to solve $u''(x) = 0, u'(0) = 0, u(1) = 1$, we will integrate $u''(x) = 0$. To satisfy $u'(0) = 0$, we get $G_1'(x) = 0$. Integrating again yields $G_1(x) = C$. To satisfy $u(1) = 1$, we have $G_1(x) = 1$. So, we have found the solution to the final boundry problem.

Now, the general formula for the elements of A^{-1} is given by the Green's functions we found above. The formulas are

$$B_{i0} = G_0(x_i) = x_i - 1, B_{im+1} = G_1(x_i) = 1$$

and

$$B_{ij} = hG(x_i, x_j) = \begin{cases} hx_j - h & 1 \leq i \leq j \\ hx_i - h & j \leq i \leq m \end{cases}$$

where i and j are the rows and columns of the inverse matrix respctively, $m + 1$ is the dimension of the matrix, and x_i, x_j are the grid or stencil points used. Using this and $h = 0.25$, we have that the first column of the 5×5 matrix is

$$\begin{pmatrix} -1 \\ -\frac{3}{4} \\ -\frac{1}{2} \\ -\frac{1}{4} \\ 0 \end{pmatrix}$$

Also, the last column is given by

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Now, the middle thre columns are given by

$$\begin{pmatrix} \frac{1}{4}\cdot\frac{1}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{1}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{1}{2}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot 1-\frac{1}{4} \end{pmatrix} = \begin{pmatrix} -\frac{3}{16} \\ -\frac{3}{16} \\ -\frac{1}{8} \\ -\frac{1}{16} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{4}\cdot\frac{1}{2}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{1}{2}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{1}{2}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot 1-\frac{1}{4} \end{pmatrix} = \begin{pmatrix} -\frac{1}{8} \\ -\frac{1}{8} \\ -\frac{1}{8} \\ -\frac{1}{16} \\ 0 \end{pmatrix}, \begin{pmatrix} \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot\frac{3}{4}-\frac{1}{4} \\ \frac{1}{4}\cdot 1-\frac{1}{4} \end{pmatrix} = \begin{pmatrix} -\frac{1}{16} \\ -\frac{1}{16} \\ -\frac{1}{16} \\ -\frac{1}{16} \\ 0 \end{pmatrix}$$

Hence, our inverse matrix is

$$A^{-1} = \begin{pmatrix} -1 & -\frac{3}{16} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{3}{4} & -\frac{3}{16} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{1}{2} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{1}{4} & -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

By the book, we have that

$$A = 16\begin{pmatrix} -0.25 & 0.25 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0.0625 \end{pmatrix} = \begin{pmatrix} -4 & 4 & 0 & 0 & 0 \\ 16 & -32 & 16 & 0 & 0 \\ 0 & 16 & -32 & 16 & 0 \\ 0 & 0 & 16 & -32 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and MATLAB gives us

$$A^{-1} = \frac{1}{16}\begin{pmatrix} -16 & -3 & -2 & -1 & 16 \\ -12 & -3 & -2 & -2 & 16 \\ -8 & -2 & -2 & -1 & 16 \\ -4 & -1 & -1 & -1 & 16 \\ 0 & 0 & 0 & 0 & 16 \end{pmatrix} = \begin{pmatrix} -1 & -\frac{3}{16} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{3}{4} & -\frac{3}{16} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{1}{2} & -\frac{1}{8} & -\frac{1}{8} & -\frac{1}{16} & 1 \\ -\frac{1}{4} & -\frac{1}{16} & -\frac{1}{16} & -\frac{1}{16} & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

which verifies our above work.

2.3 2.3. Determine the null space of the matrix A^T , where A is given in equation (2.58), and verify that the condition (2.62) must hold for the linear system to have solutions.

Solution.
Transposing the matrix A of (2.58) we have that

$$A^T = \begin{pmatrix} -h & 1 & 0 & \cdots & 0 \\ h & -2 & \ddots & \ddots & \vdots \\ 0 & 1 & \ddots & 1 & 0 \\ \vdots & \ddots & \ddots & -2 & h \\ 0 & \cdots & 0 & 1 & -h \end{pmatrix}$$

So we need to solve $A^T x = 0$. Hence,

$$\begin{aligned} &-hx_0 + x_1 && = 0 \\ &hx_0 - 2x_1 + x_2 && = 0 \\ &x_1 - 2x_2 + x_3 = 0 \\ &\vdots \\ &x_{m-1} - 2x_m + hx_{m+1} = 0 \\ &x_m - hx_{m+1} = 0 \end{aligned}$$

So, we see that $x_1 = hx_0$ and therefore, $x_2 = x_1$. Continuing, we have that $x_i = x_1$ for $3 \le i \le m$. Also, $x_0 = x_{m+1}$ since $x_m = x_1$. So, letting $x_0 = c$, we have that $c \begin{pmatrix} 1 \\ h \\ \vdots \\ h \\ 1 \end{pmatrix}$ or simply $\begin{pmatrix} 1 \\ h \\ \vdots \\ h \\ 1 \end{pmatrix}$ is the basis for the null space of A^T. Also, using MATLAB's null space function with an analogous 5×5 matrix function, we have that $\begin{pmatrix} 1 \\ h \\ \vdots \\ h \\ 1 \end{pmatrix}$ is the null space of A^T. Now, we have that

$$F = \begin{pmatrix} \sigma_0 + \frac{h}{2}f(x_0) \\ f(x_1) \\ \vdots \\ f(x_m) \\ -\sigma_1 + \frac{h}{2}f(x_{m+1}) \end{pmatrix}$$

Therefore, since $\begin{pmatrix} 1 \\ h \\ \vdots \\ h \\ 1 \end{pmatrix}$ is the basis for the null space of A^T,

$$Null(A^T)F = \sigma_0 + \frac{h}{2}f(x_0) + hf(x_1) + \cdots + hf(x_m) - \sigma_1 + \frac{h}{2}f(x_{m+1})$$

Since the system in question only has solutions if $Null(A^T) \cdot F = 0$, assume as such. Whence

$$\sigma_0 + \frac{h}{2}f(x_0) + hf(x_1) + \cdots + hf(x_m) - \sigma_1 + \frac{h}{2}f(x_{m+1}) =$$

$$\sigma_0 + \frac{h}{2}f(x_0) + h\sum_{i=1}^{m} f(x_i) - \sigma_1 + \frac{h}{2}f(x_{m+1}) = 0$$

Thus,

$$\frac{h}{2}f(x_0) + h\sum_{i=1}^{m} f(x_i) + \frac{h}{2}f(x_{m+1}) = -\sigma_0 + \sigma_1$$

must hold for the system from (2.58) to have solutions, verifying condition (2.62).

2.4 a. Modify the m-file bvp2.m so that it implements a Dirichlet boundary condition at $x = a$ and a Neumann condition at $x = b$ and test the modified program.
b. Make the same modification to the m-file bvp4.m, which implements a fourth order accurate method. Again test the modified program.

```
% bvp_2.m (edited)
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% Using 3-pt differences on an arbitrary nonuniform grid.
% Should be 2nd order accurate if grid points vary smoothly, but may
% degenerate to "first order" on random or nonsmooth grids.
%
% Different BCs can be specified by changing the first and/or last ...
    rows of
% A and F.
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/  (2007)
clear
close all
clc
ax = 0;
bx = 3;
sigma = -5;   % Neumann boundary condition at bx
beta = 3;     % Dirichlet boundary condtion at ax

f = @(x) exp(x);  % right hand side function
utrue = @(x) exp(x) + (sigma-exp(bx))*(x - ax) + beta - exp(ax);  ...
    % true soln

disp(f);
disp(utrue);

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test ...
    convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                 % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for ...
      convergence tests

  % set grid points:
```

```
    gridchoice = 'uniform';          % see xgrid.m for other choices
    x = xgrid(ax,bx,m,gridchoice);

    % set up matrix A (using sparse matrix storage):
    A = spalloc(m2,m2,3*m2);   % initialize to zero matrix

    % first row for Dirichlet BC at ax:
    A(1,1:3) = fdcoeffF(0, x(1), x(1:3));

    % interior rows:
      for i=2:m1
           A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1)));
      end

    % last row for Neuamann BC at bx:
    A(m2,m:m2) = fdcoeffF(1,x(m2),x(m:m2));

    % Right hand side:
    F = f(x);
    F(1) = beta;
    F(m2) = sigma;

    % solve linear system:
    U = A\F;


    % compute error at grid points:
    uhat = utrue(x);
    err = U - uhat;
    E(jtest) = max(abs(err));
    disp(' ')
    disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

    clf
    plot(x,U,'o')  % plot computed solution
    title(sprintf('Computed solution with %i grid points',m2));
    hold on
    plot(xfine,ufine)  % plot true solution
    hold off

    % pause to see this plot:
    drawnow
    input('Hit <return> to continue ');

end

error_table(hvals, E);   % print tables of errors and ratios
error_loglog(hvals, E);  % produce log-log plot of errors and ...
     least squares fit
```

```
% bvp4.m (edited)
% second order finite difference method for the bvp
%   u''(x) = f(x),   u'(ax)=sigma,   u(bx)=beta
% fourth order finite difference method for the bvp
%   u'' = f,   u'(ax)=sigma,   u(bx)=beta
% Using 5-pt differences on an arbitrary grid.
% Should be 4th order accurate if grid points vary smoothly.
%
% Different BCs can be specified by changing the first and/or last ...
    rows of
% A and F.
%
% From  http://www.amath.washington.edu/¬rjl/fdmbook/chapter2  (2007)
clear
close all
clc

ax = 0;
bx = 3;
sigma = -5;   % Neumann boundary condition at bx
beta = 3;     % Dirichlet boundary condtion at ax

f = @(x) exp(x);  % right hand side function
utrue = @(x) exp(x) + (sigma-exp(bx))*(x - ax) + beta - exp(ax);  ...
    % true soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,101);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test ...
    convergence:
m1vals = [10 20 40 80];
ntest = length(m1vals);
hvals = zeros(ntest,1);  % to hold h values
E = zeros(ntest,1);      % to hold errors

for jtest=1:ntest
  m1 = m1vals(jtest);
  m2 = m1 + 1;
  m = m1 - 1;                  % number of interior grid points
  hvals(jtest) = (bx-ax)/m1;  % average grid spacing, for ...
      convergence tests

  % set grid points:
  gridchoice = 'uniform';
  x = xgrid(ax,bx,m,gridchoice);

  % set up matrix A (using sparse matrix storage):
  A = spalloc(m2,m2,5*m2);   % initialize to zero matrix

  % first row for Dirichlet BC on u'(x(1))
  A(1,1:5) = fdcoeffF(0, x(1), x(1:5));
```

```
    % second row for u''(x(2))
    A(2,1:6) = fdcoeffF(2, x(2), x(1:6));

    % interior rows:
    for i=3:m
        A(i,i-2:i+2) = fdcoeffF(2, x(i), x((i-2):(i+2)));
    end

    % next to last row for u''(x(m+1))
    A(m1,m-3:m2) = fdcoeffF(2,x(m1),x(m-3:m2));
    % last row for Neumann BC on u(x(m+2))
    A(m2,m-2:m2) = fdcoeffF(1,x(m2),x(m-2:m2));

    % Right hand side:
    F = f(x);
    F(1) = beta;
    F(m2) = sigma;

    % solve linear system:
    U = A\F;


    % compute error at grid points:
    uhat = utrue(x);
    err = U - uhat;
    E(jtest) = max(abs(err));
    disp(' ')
    disp(sprintf('Error with %i points is %9.5e',m2,E(jtest)))

    clf
    plot(x,U,'o')  % plot computed solution
    title(sprintf('Computed solution with %i grid points',m2));
    hold on
    plot(xfine,ufine)  % plot true solution
    hold off

    % pause to see this plot:
    drawnow
    input('Hit <return> for next plot ');

end

error_table(hvals, E);   % print tables of errors and ratios
error_loglog(hvals, E);  % produce log-log plot of errors and ... least squares fit
```

```
%% Dallas Klumpe
%% Sci Comp
%% Homework 2
%% 2.1 Part a
clear;
close all;
clc;
fprintf('2.1\n');
fprintf('Part a\n\n');
A=(16)*[0.0625 0 0 0 0; 1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 ...
    0 0.0625];
disp(A)
%% Part b
fprintf('Part b\n\n');
Ainv=inv(A);
disp(Ainv)
%% Part c
fprintf('Part c\n\n');
syms x
G_0=(0.25*0)*piecewise((0≤x)&(x≤0),(0-1)*x,(0≤x)&(x≤1),0*x-0);
G_1=(0.25*0.25)*piecewise((0≤x)&(x≤0.25),(0.25-1)*x,(0.25≤x)&(x≤1),0.25*x-0.25);
G_2=(0.25*0.5)*piecewise((0≤x)&(x≤0.5),(0.5-1)*x,(0.5≤x)&(x≤1),0.5*x-0.5);
G_3=(0.25*0.75)*piecewise((0≤x)&(x≤0.75),(0.75-1)*x,(0.75≤x)&(x≤1),0.75*x-0.75);
G_4=(0.25*1)*piecewise((0≤x)&(x≤1),(1-1)*x,(1≤x)&(x≤1),1*x-1);
U=G_0+G_1+G_2+G_3+G_4;
u=(1/6)*x^3-(1/6)*x;
hold on
fplot(G_0);
fplot(G_1);
fplot(G_2);
fplot(G_3);
fplot(G_4);
fplot(U);
fplot(u,[0,1]);
legend({'G(x,0)','G(x,0.25)','G(x,0.5)','G(x,0.75)','G(x,1)','U(x)','u(x)'},'Locati
%% 2.2
fprintf('2.2\n\n');
B=16*[-0.25 0.25 0 0 0; 1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 ...
    0 0.0625];
disp(B);
Binv=inv(B);
disp(inv(B));
%% 2.3
fprintf('2.3\n\n');
syms h
A2=[-h h 0 0 0; 1 -2 1 0 0; 0 1 -2 1 0; 0 0 1 -2 1; 0 0 0 h -h];
AT=A2';
disp(AT);
RA=rref(A);
disp(RA);
n=null(AT);
disp(n)
nulity=size(n,2);
```

```
disp(nulity);
```