MATH 232 Homework 2

1. **Problem 1**

SOLUTION:

(a) From (2.43), we have:

$$A = \frac{1}{h^2}\begin{bmatrix} h^2 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & h^2 \end{bmatrix}, \quad U = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \quad F = \begin{bmatrix} \alpha \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ \beta \end{bmatrix}$$

Given $u(0) = u(1) = \alpha = \beta = 0$ and $h = 0.25$, our A and F become:

$$A = 16\begin{bmatrix} \frac{1}{16} & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix}, \quad F = \begin{bmatrix} 0 \\ f(x_1) \\ f(x_2) \\ f(x_3) \\ 0 \end{bmatrix}$$

U remains the same.

(b) To analytically find the inverse of A, we can use the discrete Green's function analog that was covered in class and (2.46) from LeVeque:

$$B_{i,0} = 1 - x_i$$

$$B_{i,j} = hG(x_i; x_j) \begin{cases} h(x_j - 1)x_i, & i = 1, 2, ..., j \\ h(x_i - 1)x_j, & i = j, j+1, ..., m \end{cases}$$

$$B_{i,m+1} = x_i$$

By substituting our values for h and x_i, x_j, the inverse matrix B is:

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{3}{64} & -\frac{1}{32} & -\frac{1}{64} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{32} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{64} & -\frac{1}{32} & -\frac{3}{64} & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We can check that this is in fact the inverse of A by computing the matrix product of AB by hand or in `MATLAB` to find:

$$AB = 16\begin{bmatrix} \frac{1}{16} & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{3}{64} & -\frac{1}{32} & -\frac{1}{64} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{32} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{64} & -\frac{1}{32} & -\frac{3}{64} & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} = I$$

(c) Recall that our system of equations started as $AU = F$. With B being the inverse of A, we can find our approximate solution by computing:

$$U = BF = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{3}{4} & -\frac{3}{64} & -\frac{1}{32} & -\frac{1}{64} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{32} & -\frac{1}{16} & -\frac{1}{32} & \frac{1}{2} \\ \frac{1}{4} & -\frac{1}{64} & -\frac{1}{32} & -\frac{3}{64} & \frac{3}{4} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{1}{2} \\ \frac{3}{4} \\ 0 \end{bmatrix} = \frac{1}{128} \begin{bmatrix} 0 \\ -5 \\ -8 \\ -7 \\ 0 \end{bmatrix}$$

If we plot the approximate solution to $u''(x) = f(x)$ with $u(0) = u(1) = 0$ for $h = 0.25$ with the five Green's functions we see the following:

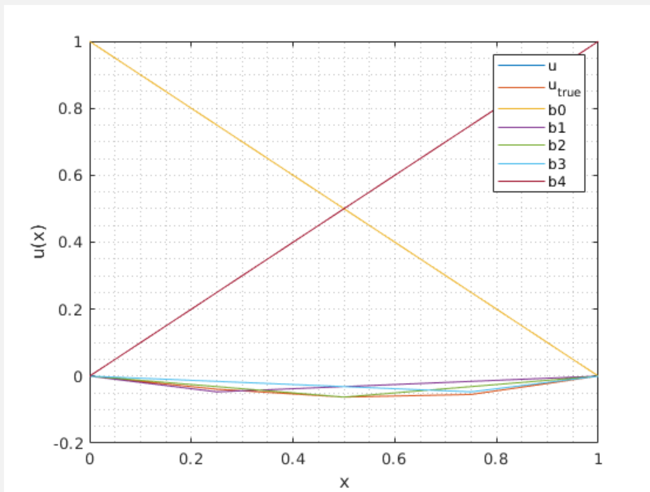

Figure 1: The approximate solution $u(x)$ with the five Green's functions.

However, we note that given our particular choice of $\alpha = \beta = 0$, we don't include the Green's functions b_0 and b_4. If we re-plot our solution without these, we can see a more accurate representation of $u(x)$ and the five Green's functions.

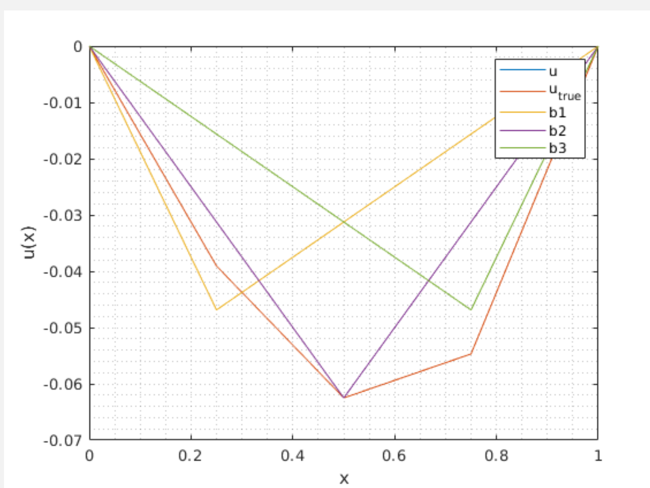

Figure 2: The approximate solution $u(x)$ with the three contributing Green's functions.

2. **Problem 2**

SOLUTION:

(a) The modifications made to `bvp_2.m` that need to be made are changing the parameters in lines 15-18 to match the BVP given, namely, `ax` $= 0$, `bx` $= 1$, $\alpha = 2$, $\beta = 3$. Similarly, we need to change lines 20 and 21 to be `f` $= 0$, and `utrue` $= 2cos(x) + \frac{3-2cos(1)}{sin(1)} sin(x)$, where `utrue` is calculated by solving the BVP $u''(x) + u(x) = 0$, with the given initial conditions:

$$u''(x) + u(x) = 0$$
$$r^2 + 1 = 0 \longrightarrow r = \pm i$$
$$u(x) = c_1 cos(x) + c_2 sin(x)$$

Applying initial conditions:

$$u(0) = c_1 + 0 = 2 \longrightarrow c_1 = 2$$
$$u(1) = 2cos(1) + c_2 sin(1) = 3 \longrightarrow c_2 = \frac{3-2cos(1)}{sin(1)}$$

Also because we are solving $u''(x) + u(x) = 0$, we must modify the script since it is intended for the Neumann problem $u''(x) = f(x)$. To do this, change line 47 to only compute the value of `A(1,1)` and have it be the 0^{th} derivative since we have Dirichlet boundary conditions. Similarly, when forming the bulk of `A` in the for loop at line 50, we must incorporate that this is $u''(x) + u(x)$ by adding to the current code `fdcoeffF(0, x(i), x((i-1):(i+1)))`, making all the interior rows computed by `A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1))) + fdcoeffF(0, x(i), x((i-1):(i+1)));`. The rest of the code is left unmodified. Running it as is, we obtain the following error table from the script:

h	error	ratio	observed order
0.05000	8.10848e-05	NaN	NaN
0.02500	2.02705e-05	4.00014	2.00005
0.01250	5.06999e-06	3.99812	1.99932
0.00625	1.26748e-06	4.00007	2.00002
0.00313	3.16866e-07	4.00004	2.00002

It has observed order 2, so it checks out with the suspected $O(h^2)$ error as computed in the lecture of 27/01/2022. See `q2abvp_2.m` in the appendix for the exact script.

(b) See next page

(b) Solving the BVP for the initial conditions $u(0) = \alpha$ and $u(1) = \beta$, we get:

$$u''(x) + u(x) = 0$$
$$r^2 + 1 = 0 \longrightarrow r = \pm i$$
$$u(x) = c_1 cos(x) + c_2 sin(x)$$

Applying initial conditions:

$$u(0) = c_1 + 0 = \alpha \longrightarrow c_1 = \alpha$$
$$u(\pi) = (\pi) = \beta \longrightarrow \alpha = -\beta$$

Hence, we see that c_2 is a free variable since there is no constraints on it, and for a solution to exist to this problem, we must have that $\alpha = -\beta$. Since this is the case, we can plot a family of solutions, using the script `family.m` found in the appendix, with $c_2 = 0$, and produced a family of cosine curves for varying initial conditions that obey $\alpha = -\beta$:

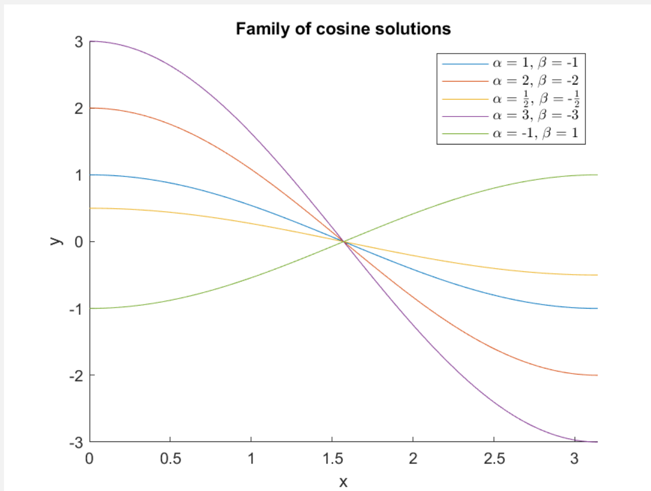

Figure 3: Family of cosine curves to the solution of $u''(x) + u(x) = 0$ with various initial conditions and $c_2 = 0$.

(c) Changing the conditions to have $a = 0$, $b = \pi$, $\alpha = 1$, $\beta = -1$, we see that by part (b), our true solution changes to $u(x) = cos(x) + c_2 sin(x)$, where c_2 is a free parameter. Here we see that $\alpha = -\beta$, so we suspect this to be a well-posed problem. Taking c_2 = 0, we see that as $h \longrightarrow 0$, the solution approaches to $f(x) = cos(x)$. We see that using 41 grid points in the figure below that our approximate solution approximates $f(x) = cos(x)$ extremely well (approximate in dotted blue, true in red, on next page).

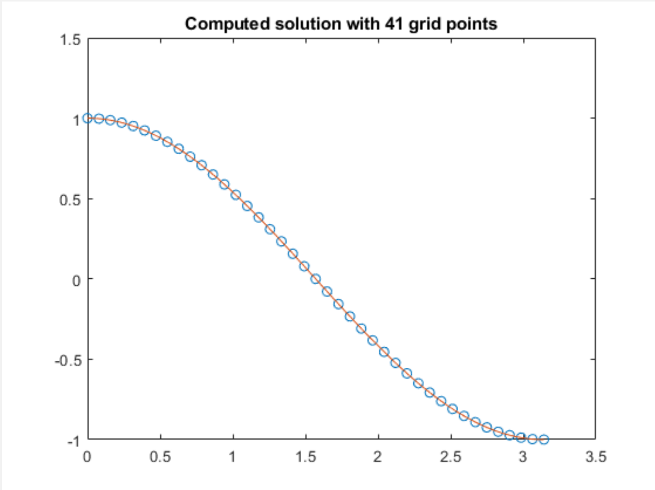

Figure 4: The approximate solution $u(x)$ to $u''(x) + u(x) = 0$, with boundary conditions $a = 0$, $b = \pi$, $\alpha = 1$, $\beta = -1$

Modifying $\beta = 1$ breaks the condition we solved for in part (b), hence the problem is not well posed, we see that as $h \longrightarrow 0$, the solution behaves completely erratically and does not approximate the true solution at all.

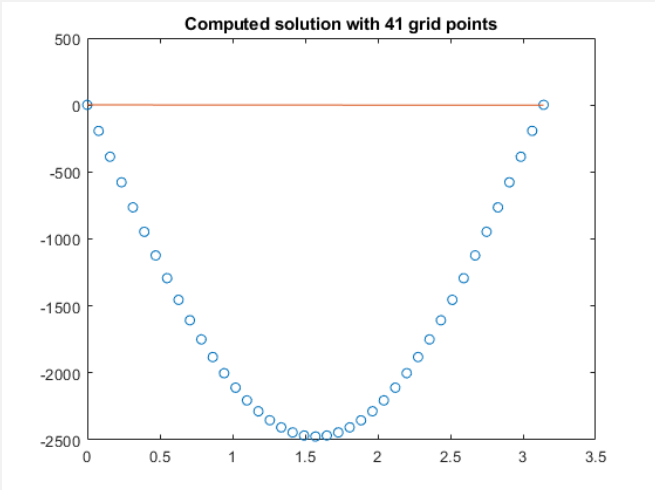

Figure 5: The approximate solution $u(x)$ to $u''(x) + u(x) = 0$, with boundary conditions $a = 0$, $b = \pi$, $\alpha = 1$, $\beta = 1$

Here we see that due to the problem being ill-posed, our numerical method does not approximate the true solution accurately whatsoever. In the table below (produced from `bvp_2.m`), we see that as $h \longrightarrow 0$, the error blows up.

h	error	ratio	observed order
0.15708	6.17509e+02	NaN	NaN
0.07854	2.47520e+03	0.24948	-2.00301
0.03927	9.90595e+03	0.24987	-2.00075
0.01963	3.96290e+04	0.24997	-2.00019
0.00982	1.58521e+05	0.24999	-2.00005

(d) If we compute the eigenvalues of the matrix A from part (c) using `MATLAB`, we see the following trend as $h \to 0$ using the script q2d.m found in the appendix:

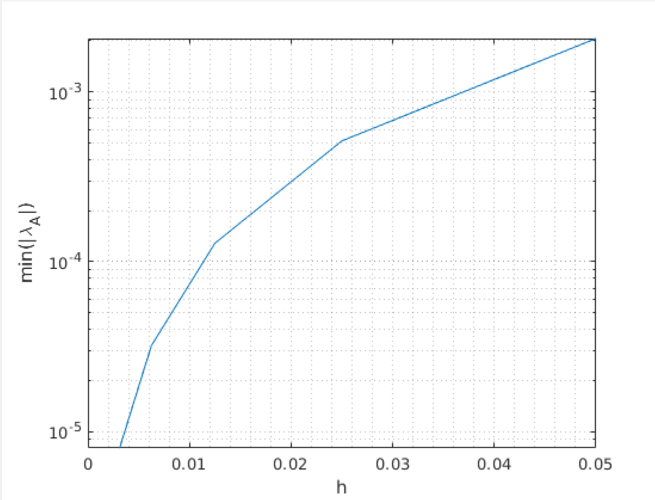

Figure 6: The minimum, absolute value of the eigenvalues of A as $h \to 0$.

We can see from this plot that the minimum, absolute value eigenvalue of A approaches 0 as $h \to 0$. If we compute $||A^{-1}||_2$ as $h \to 0$, we also see that this result blows up.

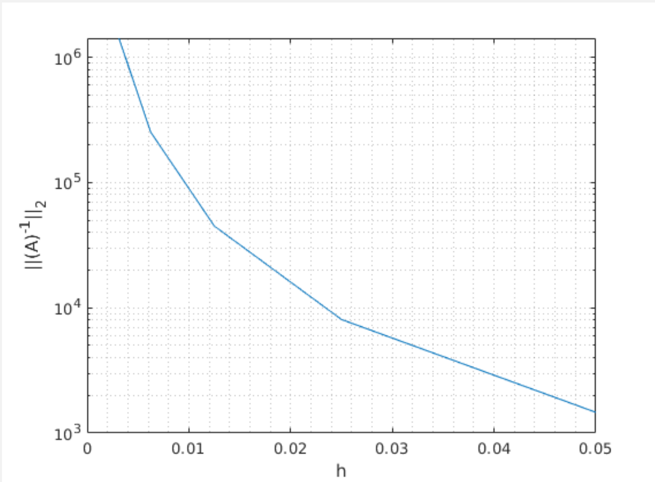

Figure 7: The result of $||A^{-1}||_2$ as $h \to 0$.

From these two plots, we can see how the discretization error yield a matrix that is not singular. However, the problem is ill-posed as the inverse matrix blows up as $h \to 0$.

Contributions:

Each group member completed the assignment individually, then the team we met together to discuss the answers complete the write up in Latex.

APPENDIX

```
q2abvp_2.m
clear all
%
% bvp_2.m
% 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)

ax = 0;
bx = 1;
sigma = 2;  % Neumann boundary condition at ax
beta = 3;    % Dirichlet boundary condtion at bx

f = @(x) 0; % right hand side function
utrue = @(x) 2*cos(x) + (3-2*cos(1))/(sin(1)) * sin(x); % true soln

% true solution on fine grid for plotting:
xfine = linspace(ax,bx,201);
ufine = utrue(xfine);

% Solve the problem for ntest different grid sizes to test convergence:
m1vals = [20 40 80 160 320];
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 Neumann BC at ax:
  A(1,1) = fdcoeffF(0, x(1), x(1:1));

  % interior rows:
  for i=2:m1
     A(i,i-1:i+1) = fdcoeffF(2, x(i), x((i-1):(i+1))) + fdcoeffF(0, x(i),
         x((i-1):(i+1)));
  end

  % last row for Dirichlet BC at bx:
  A(m2,m:m2) = fdcoeffF(0,x(m2),x(m:m2));
```

```
  % Right hand side:
  F = f(x);
  F(1) = sigma;
  F(m2) = beta;
  F = transpose(F);
  % solve linear system:
  U = A\F;

  if jtest==1
     U_save1=U;
     uhat1 = utrue(x);
  elseif jtest==2
     U_save2=U;
     V = (1/3)*(4*U_save2(1:2:end) - U_save1);
     uhat2 = utrue(x);
     err2 = V - uhat1;
  elseif jtest==3
     U_save3=U;
     V = (1/3)*(4*U_save3(1:2:end) - U_save2);
     uhat3 = utrue(x);
     err2 = V - uhat2;
  elseif jtest==4
     U_save4=U;
     V = (1/3)*(4*U_save4(1:2:end) - U_save3);
     uhat4 = utrue(x);
     err2 = V - uhat3;
     elseif jtest==5
     U_save5=U;
     V = (1/3)*(4*U_save5(1:2:end) - U_save4);
     uhat5 = utrue(x);
     err2 = V - uhat4;
  end

  % compute error at grid points:
  uhat = utrue(x);
  err = U - uhat;

  E(jtest) = max(abs(err));
  if jtest >=2
        E2(jtest-1) = max(abs(err2));
  end
  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
```

```
family.m
f = @(x) cos(x);
g = @(x) 2*cos(x);
h = @(x) 1/2 *cos(x);
a = @(x) 3 * cos(x);
b = @(x) -cos(x);

xs = linspace(0,pi);
hold on
plot(xs, f(xs))
plot(xs, g(xs))
plot(xs, h(xs))
plot(xs, a(xs))
plot(xs, b(xs))
hold off
legend('$\alpha$ = 1, $\beta$ = -1', '$\alpha$ = 2, $\beta$ = -2','$\alpha$ =
    $\frac{1}{2}$, $\beta$ = -$\frac{1}{2}$','$\alpha$ = 3, $\beta$ =
    -3','$\alpha$ = -1, $\beta$ = 1','Interpreter', 'latex')
xlabel('x')
ylabel('y')
xlim([0,pi])
title('Family of cosine solutions')
saveas(gcf,'family.png')



q2d.m
close all
% run bvp_2_p2d fist to generate the A1 - A5 matrices.
eig_A1 = eig(full(A1));
eig_A2 = eig(full(A2));
eig_A3 = eig(full(A3));
eig_A4 = eig(full(A4));
eig_A5 = eig(full(A5));

% compute min eig(As)
eA_vec = [min(abs(eig_A1)),min(abs(eig_A2)),min(abs(eig_A3))
,min(abs(eig_A4)),min(abs(eig_A5))];

figure
semilogy(1./m1vals,eA_vec)
xlabel 'h'
ylabel 'min(|\lambda_A|)'
grid minor

% compute norm(As)
nA_vec = [norm(inv(full(A1))),norm(inv(full(A2))),norm(inv(full(A3))),
norm(inv(full(A4))),norm(inv(full(A5)))];

figure
semilogy(1./m1vals,nA_vec)
xlabel 'h'
ylabel '||(A)^{-1}||_2'
grid minor
```