Problem 1. *a)* *Use the method of undetermined coefficients to set up the 5×5 Vandermonde system that would determine a fourth-order accurate finite difference approximation to $u''(x)$ based on 5 equally spaced points,*

$$u''(x) = c_{-2}u(x-2h) + c_{-1}u(x-h) + c_0u(x) + c_1u(x+h) + c_2u(x+2h) + O(h^4). \tag{1}$$

b) *Compute the coefficients using the MATLAB code* `fdstencil.m` *available from the Randy LeVeque's website. The codes will be posted also in MyCourses under Contents/Codes (download both* `fdstencil.m` *and* `fdcoeffV.m`*). Verify that the coefficients satisfy the system you determined in part a).*

Solution to a). In order to use the method of undetermined coefficients we first need to Taylor-expand the expressions on Eq. (1):

$$u(x+h) = u(x) + u'(x)h + u''(x)\frac{h^2}{2} + u'''(x)\frac{h^3}{6} + u^{(4)}(x)\frac{h^4}{24} + O(h^5) \tag{2a}$$

$$u(x-h) = u(x) - u'(x)h + u''(x)\frac{h^2}{2} - u'''(x)\frac{h^3}{6} + u^{(4)}(x)\frac{h^4}{24} + O(h^5) \tag{2b}$$

$$u(x+2h) = u(x) + 2u'(x)h + 2u''(x)h^2 + u'''(x)\frac{4h^3}{3} + u^{(4)}(x)\frac{2h^4}{3} + O(h^5) \tag{2c}$$

$$u(x-2h) = u(x) - 2u'(x)h + 2u''(x)h^2 - u'''(x)\frac{4h^3}{3} + u^{(4)}(x)\frac{2h^4}{3} + O(h^5). \tag{2d}$$

Now, matching these expressions with the coefficients c_i on (1), we get

$$\begin{aligned}
u''(x) = {} & (c_{-2} + c_{-1} + c_0 + c_1 + c_2)u(x) \\
& + (-2c_{-2} - c_{-1} + c_1 + 2c_2)\,hu'(x) \\
& + \frac{1}{2}\left(4c_{-2} + c_{-1} + c_1 + 4c_2\right)\, h^2u''(x) \\
& + \frac{1}{6}\left(-8c_{-2} - c_{-1} + c_1 + 8c_2\right)\, h^3u'''(x) \\
& + \frac{1}{24}\left(16c_{-2} + c_{-1} + c_1 + 16c_2\right)\, h^4u^{(4)}(x).
\end{aligned}$$

In order for this expression to be compatible with Eq. (1), the following linear system must be satisfied:

$$\begin{aligned}
c_{-2} + c_{-1} + c_0 + c_1 + c_2 &= 0 \\
-2c_{-2} - c_{-1} + c_1 + 2c_2 &= 0 \\
4c_{-2} + c_{-1} + c_1 + 4c_2 &= \frac{2}{h^2} \\
-8c_{-2} - c_{-1} + c_1 + 8c_2 &= 0 \\
16c_{-2} + c_{-1} + c_1 + 16c_2 &= 0,
\end{aligned}$$

or, in matrix form,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{bmatrix} \begin{bmatrix} c_{-2} \\ c_{-1} \\ c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{h^2} \\ 0 \\ 0 \end{bmatrix}. \tag{3}$$

The solution to this system is

$$\begin{bmatrix} c_{-2} \\ c_{-1} \\ c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} -\frac{1}{12h^2} \\ \frac{4}{3h^2} \\ -\frac{5}{2h^2} \\ \frac{4}{3h^2} \\ -\frac{1}{12h^2} \end{bmatrix}.$$ □

Solution to b). Indeed, using the provided code in the `fdcoeffV.m` *file, we get $h^2 * c_i$, where c_i are the coefficients that we got on part a):*

```
>> j = [-2:2];
>> fdcoeffV(2,0,j)

ans =
   -0.0833    1.3333   -2.5000    1.3333   -0.0833
```

□

Problem 2. Consider the following finite difference approximation for $u'(\bar{x})$

$$Du(\bar{x}) = \frac{1}{2h}\left[3u(\bar{x}) - 4u(\bar{x}-h) + u(\bar{x}-2h)\right]. \tag{4}$$

a) *Approximate the derivative $u'(\bar{x})$ for one of the choices below*

   i. $u(x) = e^{-3x}\sin(2x), \bar{x} = 1.23$

   ii. $u(x) = \arctan(4x), \bar{x} = -0.45$

   using the formula above for $h = 10^{-1}$, $h = 5\times10^{-2}$, $h = 10^{-2}$, $h = 5\times10^{-3}$, $h = 10^{-3}$, and $h = 5\times10^{-4}$. Make a table containing approximate derivatives and the corresponding errors (error = actual − approximate) for all h values.

b) *Plot h values vs. absolute errors in log-log scale (i.e., plot $\log(h_i)$ vs. $\log|E(h_i)|$). You can use ~~MATLAB's `loglog`~~ (Mathematica's* `ListLogLogPlot`*!) to produce the plot.*

c) *Assume that the error behaves according to $|E(h)| \approx Ch^p$ and determine the (numerical) order of convergence p using your results from part b). Find the constant C in the error expression.*

Solution to a). Start with the first function, i.. We rewrite its derivative approximation (4) as a function of h:

$$Du(h)|_{\bar{x}=1.23} = \frac{1}{2h}\left[3e^{-3\cdot1.23}\sin(2\cdot1.23) - 4e^{-3\cdot(1.23-h)}\sin[2\cdot(1.23-h)] + e^{-3\cdot(1.23-2h)}\sin[2\cdot(1.23-2h)]\right].$$

The values for this function for the given values of h are seen on the output on the following `Mathematica` *snippet:*

```
In[1]:= D1[h_] :=
  1/(2*h)* (
     3*E^(-3*1.23)*Sin[2*1.23] -
     4*E^(-3*(1.23 - h))*Sin[2*(1.23 - h)] +
     E^(-3*(1.23 - 2 h))*Sin[2*(1.23 - 2 h)]
     );
Table[D1[h], {h, {10^-1, 5*10^-2, 10^-2, 5*10^-3, 10^-3, 5*10^-4 }}]

Out[2]= {-0.0834738, -0.0853506, -0.0859593, -0.0859781, -0.0859842, -0.0859843}
```

Similarly, for function ii.,

$$Du(h)|_{\bar{x}=-0.45} = \frac{1}{2h}\left[3\arctan\left[4\cdot(-0.45)\right] - 4\arctan\left[4\cdot(-0.45-h)\right] + \arctan\left[4\cdot(-0.45-2h)\right]\right],$$

we get

```
In[3]:= D2[h_] :=
  1/(2*h)*(
     3*ArcTan[4*(-0.45)]
     - 4*ArcTan[4*(-0.45 - h)]
     + ArcTan[4*(-0.45 - 2 h)]
     );
Table[D2[h], {h, {10^-1, 5*10^-2, 10^-2, 5*10^-3, 10^-3, 5*10^-4 }}]

Out[4]= {0.909797, 0.933326, 0.942927, 0.943277, 0.943391, 0.943395}
```

The actual values for the derivatives of these functions evaluated at the given respective $\bar{x}$ values are

```
In[1]:= D[E^(-3x)*Sin[2x], x] /. x -> 1.23
In[2]:= D[ArcTan[4x], x] /. x -> -0.45

Out[1]= -0.0859844
Out[2]= 0.943396
```

Let us summarize our results in the following table: [1]

h	$^{(i.)}Du$	$^{(i.)}E$	$^{(ii.)}Du$	$^{(ii.)}E$
10^{-1}	-0.0834738	0.00251063	0.909797	0.0335994
5×10^{-2}	-0.0853506	0.000633807	0.933326	0.0100705
10^{-2}	-0.0859593	0.0000251206	0.942927	0.000469375
5×10^{-3}	-0.0859781	6.26751×10^{-6}	0.943277	0.000119657
10^{-3}	-0.0859842	2.50265×10^{-7}	0.943391	4.86187×10^{-6}
5×10^{-4}	-0.0859843	6.25523×10^{-8}	0.943395	1.21785×10^{-6}

□

Solution to b). The following plots were generated using Mathematica's `ListLogLogPlot`*:*

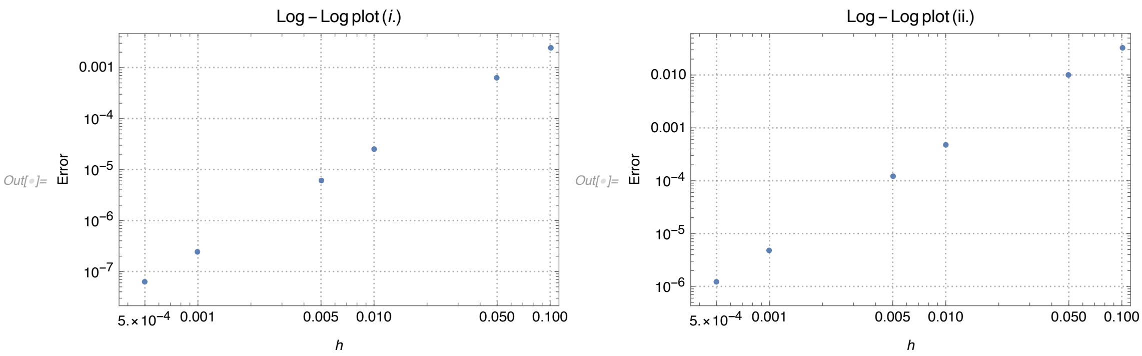

Figure 1: Log-Log plots of the absolute errors (y-axis) vs. the h values (x-axis) for the derivative approximations to the function in i. (left) and the function in ii. (right). □

---

[1] *The left-superscripts denote the function (i. or ii.) to which the derivative approximations (Du) and the errors (E) belong. Also, the errors are written as absolute-value errors.*

Solution to c). If the error behaves like $|E(h)| \approx Ch^p$, then we must have

$$\log|E(h)| \approx \log|C| + p \log h.$$

The Mathematica snippet below shows that $p \approx 2$, which signals that the approximation (4) has quadratic order of convergence. We also see that $\log {}^{(i.)}C = -1.37376$ and $\log {}^{(ii.)}C = 1.17351$, so that ${}^{(i.)}C \approx 0.25$ and ${}^{(ii.)}C \approx 3.23$.

```
In[30]:=
logdata1 = Table[{Log[h[[i]]], Log[error1[[i]]]}, {i, 1, 6}];
logdata2 = Table[{Log[h[[i]]], Log[error2[[i]]]}, {i, 1, 6}];
Fit[logdata1, {1, x}, x]
Fit[logdata2, {1, x}, x]

Out[31]= -1.37376 + 2.0016 x
Out[32]= 1.17351 + 1.93804 x
```

□

Problem 3. Consider the BVP

$$u_{xx} = 1 - |x|, \; x \in (-1, 1) \tag{5a}$$

$$u(-1) = 5, \; u(1) = 7. \tag{5b}$$

a) Solve the BVP analytically.

b) Discretize the BVP using grid points $x_i = -1 + ih$, $i = 0, 1, \ldots, n+1$ where $h = 2/(n+1)$ by using the centered finite difference scheme. Solve the resulting linear system with an $n \times n$ coefficient matrix and plot the numerical solution for $n = 49$ along with the exact solution you computed in part (a).

c) Record the L^1, L^2, and L^∞ norm errors for $n = 24, 49, 99, 199$.

d) Find the slope of the line in a log-log plot of the error $\|\mathbf{u}_n - \mathbf{u}_{exact}\|_p$ $(p = 1, 2, \infty)$ as a function of n. Is this what would you expect? Explain.

e) Solve the linear system for $n \in \{9, 49, 99, 999, 4999, 9999\}$ and document the CPU times.

Solution to a). Integrating twice, we get

$$u(x) = \begin{cases} \iint (1+x)\, dx^2 & x \in (-1, 0]; \\ \iint (1-x)\, dx^2 & x \in [0, 1). \end{cases} \tag{6a}$$

Since the function u is assumed to be continuous, the value of the piecewise at $x = 0$ must be consistent. In fact, since we are evaluating a second-order ODE, the function u must be (at least) C^2, which means that not just u, but also u' and u'' must have a consistent value at $x = 0$; we shall use this in what follows to determine some of the coefficients of integration.

Expanding (6a), we have

$$u(x) = \begin{cases} \frac{x^3}{6} + \frac{x^2}{2} + {}^{-}C_1 x + {}^{-}C_2 & x \in (-1, 0]; \\ -\frac{x^3}{6} + \frac{x^2}{2} + {}^{+}C_1 x + {}^{+}C_2 & x \in [0, 1). \end{cases} \tag{6b}$$

Hence we have four yet-to-be-determined constants ${}^{-}C_1$, ${}^{-}C_2$, ${}^{+}C_1$, and ${}^{+}C_2$. At $x = 0$, the two expressions must be equal; thus,

$$u(0) = {}^{-}C_2 = {}^{+}C_2.$$

So we may drop the superscript and just call this constant C_2. Now, as we alluded to earlier, $u'(0)$ must also have a consistent value from the expressions in the piecewise; so

$$u'(0) = {}^{-}C_1 = {}^{+}C_1.$$

Thus, again, we drop the superscript and call this constant C_1. Hence we are left with two coefficients C_1 and C_2, and we have two boundary conditions, so the problem can be fully determined:

$$
\begin{aligned}
5 = u(-1) &= -\frac{1}{6} + \frac{1}{2} - C_1 + C_2; \\
7 = u(1) &= -\frac{1}{6} + \frac{1}{2} + C_1 + C_2.
\end{aligned}
$$

So we have the system

$$
\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} \frac{14}{3} \\ \frac{20}{3} \end{bmatrix},
$$

with solution

$$
\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{17}{3} \end{bmatrix}
$$

Therefore, we conclude that our function u is given by

$$
u(x) = \begin{cases} \frac{x^3}{6} + \frac{x^2}{2} + x + \frac{17}{3} & x \in (-1, 0]; \\ -\frac{x^3}{6} + \frac{x^2}{2} + x + \frac{17}{3} & x \in [0, 1). \end{cases}
$$

Or, simply,

$$
\boxed{u(x) = -\frac{|x|^3}{6} + \frac{x^2}{2} + x + \frac{17}{3}} \qquad \square \qquad (7)
$$

Solution to b). The centered discretization of the BVP (5) is of the form

$$
\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = 1 - |x_i|, \qquad i = 1, \ldots, n
$$

or, in matrix form,

$$
\underbrace{\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{bmatrix}}_{\mathbf{u}} = \underbrace{\begin{bmatrix} 1 - |x_1| - 5/h^2 \\ 1 - |x_2| \\ \vdots \\ \vdots \\ 1 - |x_{n-1}| \\ 1 - |x_n| - 7/h^2 \end{bmatrix}}_{\mathbf{f}}.
$$

The following MATLAB code solves this linear system for $\mathbf{u}$ and plots both the numerical solution and the closed-form solution we found in part a):

```
%Set constants:
a = -1;
b = 1;
alph = 5;
bet = 7;
n = 49;
h = (b-a)/(n+1);

%Generate the matrix A (size nxn):
A = zeros(n);  %initialize nxn matrix
for i = 1:n
    for j = 1:n
        if i == j
            A(i,i) = -2/h^2;
        elseif (j == i+1) || (i == j+1)
            A(i,j) = 1/h^2;
        end
    end
end
```

```
%define the x grid in either of the two ways:

%Method 1:
% for i = 0:n+1
%     x(i+1) = -1+i*h;
% end

%or Method 2:
x = linspace(a,b,n+2);  %size n+2 (n interior pts + 2 BCs)

%define function vector f:
f = zeros(n,1);     %initialize nx1 vector
for i = 1:n
    if i==1
        f(i) = 1 - abs(x(i+1)) - alph/h^2;
    elseif i == n
        f(i) = 1 - abs(x(i+1)) - bet/h^2;
    else
        f(i) = 1 - abs(x(i+1));
    end
end

%Use linear solver to solve Au=f for u:
u  = linsolve(A,f);
usol = [alpha; u; beta]; %extend solution to include BCs

%Plot results:
plot(x,usol, "r--x")
hold on
funct = @(t) -(abs(t)^3)/6 + t^2/2 + t + 17/3; %closed-form solution
fplot(funct, [-1,1], "g--o")
ylabel("u(x)")
xlabel("x")
legend("Numerical Solution", "Exact Solution", 'Location','northwest')
exportgraphics(gcf,'BVP_1.pdf')
```

The code generates the following plot:

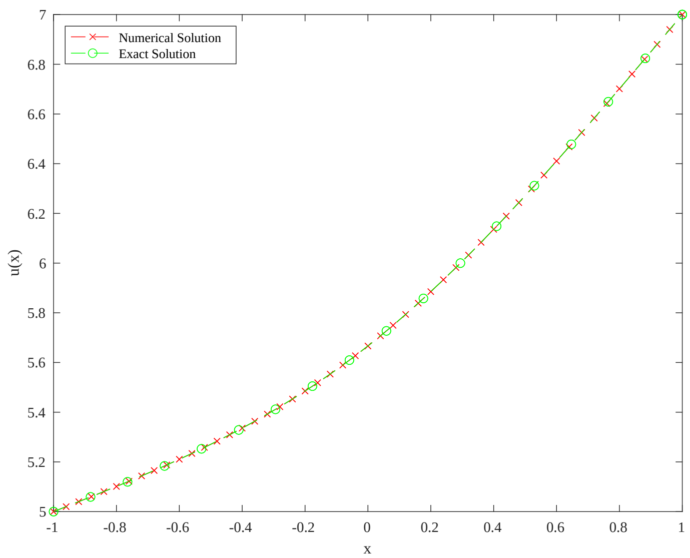

Figure 2: Closed-form and numerical solutions (for $n = 49$ grid points) of the BVP (5).

They match quite nicely! □

Solution to c). We modify the code to record all three norms for the different n*-values:*

```
%Set constants:
a = -1;
b = 1;
alph = 5;
bet = 7;
N = [24,49,99,199];

%initialize vectors of norms to be used later:
norml1_vec =  zeros(size(N,2),1);
norml2_vec =  zeros(size(N,2),1);
normlinf_vec =  zeros(size(N,2),1);

for n = N

    h = (b-a)/(n+1);

    %Generate the matrix A (size nxn):
    A = zeros(n);  %initialize nxn matrix
    for i = 1:n
        for j = 1:n
            if i == j
                A(i,i) = -2/h^2;
            elseif (j == i+1) || (i == j+1)
                A(i,j) = 1/h^2;
            end
        end
    end

    %define the x grid:
    x = linspace(a,b,n+2);  %size n+2 (n interior pts + 2 BCs)

    %define function vector f:
    f = zeros(n,1);     %initialize nx1 vector
    for i = 1:n
        if i==1
            f(i) = 1 - abs(x(i+1)) - alph/h^2;
        elseif i == n
            f(i) = 1 - abs(x(i+1)) - bet/h^2;
        else
            f(i) = 1 - abs(x(i+1));
        end
    end

    %Use linear solver to solve Au=f for u:
    u  = linsolve(A,f);
    usol = [alph; u; bet]; %extend solution to include BCs

    %closed-form solution
    funct = @(t) -(abs(t)^3)/6 + t^2/2 + t + 17/3;


    %generate vector of errors
    error_vec =  zeros(n,1);
    for i = 1:n
        error_vec(i) = usol(i) - funct(x(i));
    end

    l1 = norm(error_vec,1);
    l2 = norm(error_vec,2);
    l_inf = norm(error_vec,Inf);

    %fill in vectors of norms (we'll use these in the next part):
    it =  find(N==n);    %get the n-index of the tuple N
    norml1_vec(it) = l1;
    norml2_vec(it) = l2;
    normlinf_vec(it) = l_inf;

    %displays the norms, for each n
    norm_display_1 = ['The L1 norm for n= ',num2str(n), ' is ', num2str(l1)];
    norm_display_2 = ['The L2 norm for n= ',num2str(n), ' is ', num2str(l2)];
    norm_display_inf = ['The L^inf norm for n= ',num2str(n), ' is ', num2str(l_inf)];
    disp(norm_display_1)
    disp(norm_display_2)
    disp(norm_display_inf)
end
```

The resulting norms are

```
The L1 norm for n= 24 is 0.0033067
The L2 norm for n= 24 is 0.00076889
The L^inf norm for n= 24 is 0.000256
The L1 norm for n= 49 is 0.006656
The L2 norm for n= 49 is 0.001089
The L^inf norm for n= 49 is 0.00026667
The L1 norm for n= 99 is 0.003332
The L2 norm for n= 99 is 0.00038494
The L^inf norm for n= 99 is 6.6667e-05
The L1 norm for n= 199 is 0.0016665
The L2 norm for n= 199 is 0.00013609
The L^inf norm for n= 199 is 1.6667e-05
```

□

Solution to d). The figure shows a log-log plot of the errors that we found on part c):

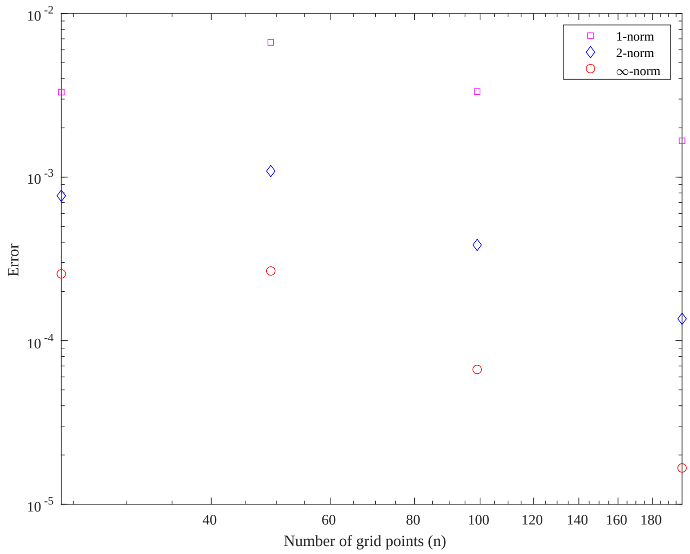

Figure 3: Log-Log plot of the errors using 1-, 2-, and ∞-norms, as a function of the number of grid points n.

The results for $n = 24$ stray from what would otherwise be a (fairly) straight line. Nevertheless, we could get a slope for all three norms by considering the values at any two n-values other than $n = 24$. For instance, we may add in our code

```
%Get the slopes
slope_l1 = ( norml1_vec(end-1) -  norml1_vec(end-2) )/ ( N(end-1) - N(end-2));
slope_l2 = ( norml2_vec(end-1) -  norml2_vec(end-2) )/ ( N(end-1) - N(end-2));
slope_linf = ( normlinf_vec(end-1) -  normlinf_vec(end-2) )/ ( N(end-1) - N(end-2));
```

This yields the slopes

```
slope_l1 =   -6.6480e-05
slope_l2 =   -1.4082e-05
slope_linf = -4.0000e-06
```

There are both unexpected and expected results here. Firstly, I was not expecting to find lower errors for $n = 24$ than for $n = 49$. This rather surprising behavior is consistent for all three norms tested. On the other hand, the way in which the norms compare with one another for a fixed number of grid points n was indeed expected, since we know that $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1$ for all $\boldsymbol{x} \in \mathbb{R}^m$. Proof: Let $\hat{i} \in [1, m]$ be the index that maximizes $|x_i|$; that is,

$$\|\boldsymbol{x}\|_\infty = \max_{1\leq i\leq m} |x_i| = |x_{\hat{i}}|.$$

Then,

$$\|\boldsymbol{x}\|_2 = \left(\sum_{i=1}^{m} |x_i|^2\right)^{1/2} = \left(|x_{\hat{i}}|^2 + \sum_{i\neq\hat{i}} |x_i|^2\right)^{1/2} \geq |x_{\hat{i}}| = \|\boldsymbol{x}\|_\infty. \qquad \surd$$

As for the other inequality, note that

$$\|\boldsymbol{x}\|_2^2 = \sum_{i=1}^{m} |x_i|^2 \leq \sum_{i=1}^{m} |x_i|^2 + 2\sum_{i\neq j} |x_i||x_j| = \|\boldsymbol{x}\|_1^2 \quad \Longrightarrow \quad \|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1. \qquad \surd \qquad \square$$

Solution to e). The following code solves the system for $n \in \{9, 49, 99, 999, 4999, 9999\}$*, and documents the CPU times (in seconds):*

```
%Set constants:
a = -1;
b = 1;
alph = 5;
bet = 7;
N = [9, 49, 99, 999, 4999, 9999];

for n = N

    %Start CPU clock
    tStart = cputime;

    h = (b-a)/(n+1);

    %Generate the matrix A (size nxn):
    A = zeros(n);  %initialize nxn matrix
    for i = 1:n
        for j = 1:n
            if i == j
                A(i,i) = -2/h^2;
            elseif (j == i+1) || (i == j+1)
                A(i,j) = 1/h^2;
            end
        end
    end

    %define the x grid:
    x = linspace(a,b,n+2);  %size n+2 (n interior pts + 2 BCs)

    %define function vector f:
    f = zeros(n,1);     %initialize nx1 vector
    for i = 1:n
        if i==1
            f(i) = 1 - abs(x(i+1)) - alph/h^2;
        elseif i == n
            f(i) = 1 - abs(x(i+1)) - bet/h^2;
        else
            f(i) = 1 - abs(x(i+1));
        end
    end

    %Use linear solver to solve Au=f for u:
    u  = linsolve(A,f);
    usol = [alph; u; bet]; %extend solution to include BCs

    tEnd = cputime - tStart;
    %displays the cpu time
    cpu_display = ['For n= ',num2str(n),' the CPU time was ', num2str(tEnd),' seconds.'
    ];
    disp(cpu_display)
end
```

The results were as follows:

```
For n= 9  the CPU time was 0.01 seconds.
For n= 49  the CPU time was 0 seconds.
```

```
For n= 99  the CPU time was 0.01 seconds.
For n= 999  the CPU time was 0.16 seconds.
For n= 4999  the CPU time was 3.03 seconds.
For n= 9999  the CPU time was 19.56 seconds.
```

I do not understand how MATLAB claims that it took "0" seconds to run the code for $n = 49$... *Not sure how reliable this* `cputime` *function really is...* □

Problem 4. Use the centered finite difference scheme to approximate solutions to the linear BVP

$$u'' = u + \frac{2}{3}e^x, \quad u(0) = 0, \quad u(1) = \frac{1}{3}e. \tag{8}$$

- *a) Specify the entries of the* $n \times n$ *matrix* A *and the vector* $\mathbf{F}$ *in the linear system* $A\mathbf{U} = \mathbf{F}$ *resulting from the approximation.*
- *b) Plot the approximate solution for* $n = 69$ *(number of interior grid points) together with the exact solution* $u(x) = 1/3\,xe^x$.
- *c) Plot the absolute error as a function of* x *in a semi-log plot (i.e.* x_i *vs.* $\log|e(x_i)| = \log|U_i - u(x_i)|$*). You can use MATLAB's* `semilogy` *to produce the plot.*

Solution to a). The centered discretization of the BVP (8) is of the form

$$\begin{aligned}
&\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} - u_i = \frac{2}{3}e^x \\
\implies &\frac{u_{i-1} - 2u_i + u_{i+1} - h^2 u_i}{h^2} = \frac{2}{3}e^x \\
\implies &\frac{u_{i-1} - \left(2 + h^2\right) u_i + u_{i+1}}{h^2} = \frac{2}{3}e^x \qquad i = 1, \dots, n.
\end{aligned}$$

In matrix form,

$$\underbrace{\frac{1}{h^2}\begin{bmatrix} -(2+h^2) & 1 & & & & \\ 1 & -(2+h^2) & 1 & & & \\ & 1 & -(2+h^2) & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -(2+h^2) & 1 \\ & & & & 1 & -(2+h^2) \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{bmatrix}}_{\mathbf{U}} = \underbrace{\begin{bmatrix} 2/3\, e^{x_1} \\ 2/3\, e^{x_2} \\ \vdots \\ \vdots \\ 2/3\, e^{x_{n-1}} \\ 2/3\, e^{x_n} - e/(3h^2) \end{bmatrix}}_{\mathbf{F}}. \quad \square$$

Solution to b). The following MATLAB code solves this linear system for $\mathbf{U}$ *and plots the numerical and exact solutions:*

```
%Set constants:
a = 0;
b = 1;
alph = 0;
bet = (1/3)*exp(1);
n = 69;
h = (b-a)/(n+1);

```

```
%Generate the matrix A (size nxn):
A = zeros(n);  %initialize nxn matrix
for i = 1:n
    for j = 1:n
        if i == j
            A(i,i) = -(2+h^2)/h^2;
        elseif (j == i+1) || (i == j+1)
            A(i,j) = 1/h^2;
        end
    end
end

%define the x-grid:
x = linspace(a,b,n+2);  %size n+2 (n interior pts + 2 BCs)

%define function vector f:
f = zeros(n,1);     %initialize nx1 vector
for i = 1:n
    if i==n
        f(i) = (2/3)*exp(x(i)) - exp(1)/(3*h^2);
    else
        f(i) = (2/3)*exp(x(i));
    end
end

%Use linear solver to solve Au=f for u:
u  = linsolve(A,f);
usol = [alph; u; bet]; %extend solution to include BCs


%generate vector of errors and vector of exact solution:
abserror_vec =  zeros(n,1);
funct_vec =  zeros(n,1);
funct = @(t) (1/3)*t*exp(t); %closed-form solution
for i = 1:n
    funct_vec(i) = funct (x(i));
    abserror_vec(i) = abs( usol(i) - funct_vec(i) );
end

%extend exact solution vector to include BCs
funct_vec = [alph; funct_vec; bet];

%generate vector of errors
abserror_vec =  zeros(n,1);
funct = @(t) (1/3)*t*exp(t); %closed-form solution
for i = 1:n
    abserror_vec(i) = abs( usol(i) - funct(x(i)) );
end

%extend abs error vector to include BCs
abserror_vec = [0; abserror_vec; 0];

%Semilog plot of the absolute error vs x:
semilogy(x,abserror_vec, "m+")
ylabel('Error')
xlabel('x')
exportgraphics(gcf,'abserror_Prob4.pdf')
close


%Plot results:
plot(x,usol, "r*")
hold on
plot(x,funct_vec, "g+-", "LineWidth",2)
ylabel('u(x)')
xlabel('x')
legend("Numerical Solution", "Exact Solution", 'Location','northwest')
exportgraphics(gcf,'BVP_Prob4.pdf')
close
```

Here is the resulting plot of the numerical and exact solutions:

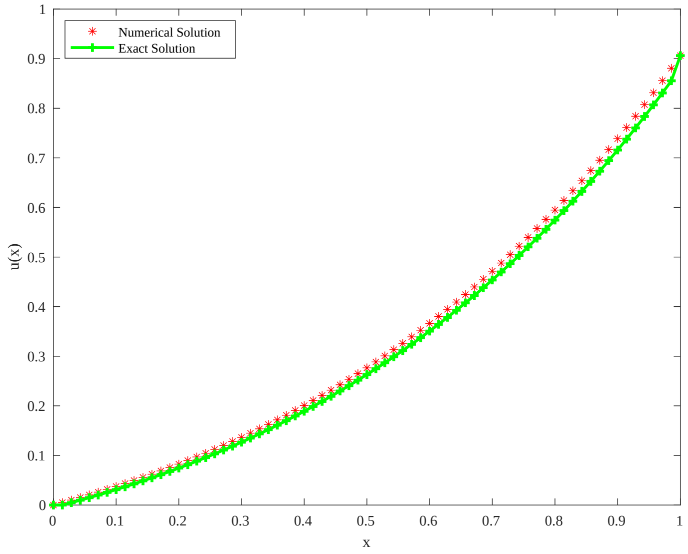

Figure 4: Exact and numerical solutions (for $n = 69$ *grid points) of the BVP (8).*

□

Solution to c). Here is the resulting semi-log plot from the code presented in b):

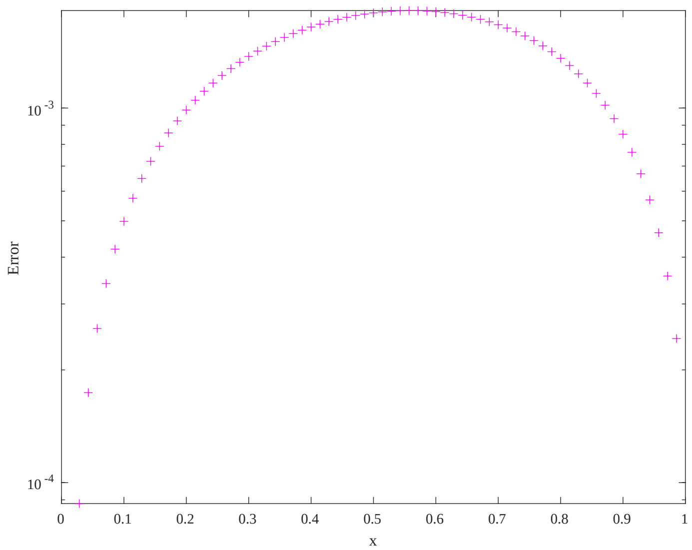

Figure 5: Semilog plot of the absolute error as a function of x.

□

Problem 5. Repeat the Problem 4 for the linear BVP

$$u'' = \left(2 + 4x^2\right) u, \quad u(0) = 1, \quad u(1) = e \tag{9}$$

with the exact solution $u(x) = e^{x^2}$.

Solution to a). *In what follows we shall use the notation*

$$ {}^{h}\Psi_i \equiv h^2 \left(2 + 4x_i^2\right). $$

Now, the centered discretization of the BVP (9) is of the form

$$
\begin{aligned}
&\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} - \left(2 + 4x_i^2\right) u_i = 0 \\
\Longrightarrow \quad &\frac{u_{i-1} - 2u_i + u_{i+1} - {}^{h}\Psi_i u_i}{h^2} = 0 \\
\Longrightarrow \quad &u_{i-1} - \left(2 + {}^{h}\Psi_i\right) u_i + u_{i+1} = 0 \qquad i = 1, \ldots, n.
\end{aligned}
$$

In matrix form,

$$
\underbrace{\begin{bmatrix}
-(2 + {}^{h}\Psi_1) & 1 & & & & \\
1 & -(2 + {}^{h}\Psi_2) & 1 & & & \\
 & 1 & -(2 + {}^{h}\Psi_3) & 1 & & \\
 & & \ddots & \ddots & \ddots & \\
 & & & 1 & -(2 + {}^{h}\Psi_{n-1}) & 1 \\
 & & & & 1 & -(2 + {}^{h}\Psi_n)
\end{bmatrix}}_{A}
\underbrace{\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_i \\ \vdots \\ u_n \end{bmatrix}}_{U}
=
\underbrace{\begin{bmatrix} -1 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ -e \end{bmatrix}}_{F}.
\qquad \square
$$

Solution to b). *The following MATLAB code solves this linear system for* $\mathbf{U}$ *and plots the numerical and exact solutions:*

```
%Set constants:
a = 0;
b = 1;
alph = 1;
bet = exp(1);
n = 69;
h = (b-a)/(n+1);

%define the x-grid
%size n (n interior pts only for now; 2 bd pts will be added later):
x = linspace(a+h,b-h,n);

% %define the Psi vector
 Psi = zeros(n,1);      %initialize nx1 vector
 for i = 1:n
     Psi(i) = h^2 * (2 + 4*x(i)^2);
 end

%Generate the matrix A (size nxn):
A = zeros(n);  %initialize nxn matrix
for i = 1:n
    for j = 1:n
        if i == j
            A(i,i) = - ( 2 + Psi(i) );
        elseif (j == i+1) || (i == j+1)
            A(i,j) = 1;
        end
    end
end


%define function vector f:
f = zeros(n,1);      %initialize nx1 vector
for i = 1:n
    if i==1
        f(i) = - 1;
    elseif i == n
        f(i) = - exp(1);
```

```
    end
end


%Use linear solver to solve Au=f for u:
u  = linsolve(A,f);
usol = [alph; u; bet]; %extend solution to include BCs


%redefine x to include bd pts
x = linspace(a,b,n+2);  %size n+2 (n interior pts + 2 BCs)


%generate vector of errors and vector of exact solution:
abserror_vec =  zeros(n,1);
funct_vec =  zeros(n,1);
funct = @(t) exp(t^2); %closed-form solution
for i = 1:n
    funct_vec(i) = funct (x(i));
    abserror_vec(i) = abs( usol(i) - funct_vec(i) );
end

%extend abs error vector to include BCs
abserror_vec = [0; abserror_vec; 0];

%Semilog plot of the absolute error vs x:
semilogy(x,abserror_vec, "m+")
ylabel('Error')
xlabel('x')
exportgraphics(gcf,'abserror_Prob5.pdf')
close


%extend exact solution vector to include BCs
funct_vec = [alph; funct_vec; bet];

%Plot results:
plot(x,usol, "b*")
hold on
plot(x,funct_vec, "m-o", "LineWidth",2)
ylabel('u(x)')
xlabel('x')
legend("Numerical Solution", "Exact Solution", 'Location','northwest')
exportgraphics(gcf,'BVP_Prob5.pdf')
close
```

Here is the resulting plot of the numerical and exact solutions:

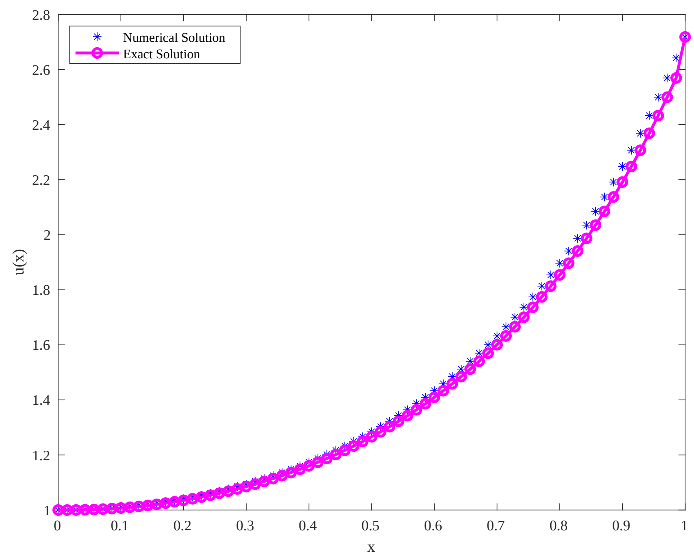

Figure 6: Exact and numerical solutions (for $n = 69$ grid points) of the BVP (9).

□

Solution to c). Here is the resulting semi-log plot from the code presented in b):

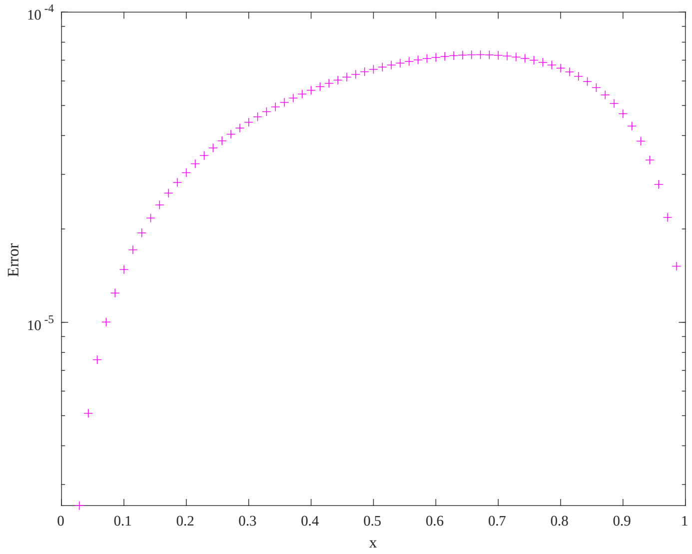

Figure 7: Semilog plot of the absolute error as a function of x.

□