ENM 5220: Numerical Methods for PDEs

Assignment 1: Numerical Differentiation and Intergration

Luyando Kwenda

1 Problem 1

No submission

2 Problem 2

Numerically evaluate the derivative of the following function:

$$f(x) = \sin(2x) \cdot \cos(20x) + e^{\sin(2x)}, \quad x \in [0, 2\pi], \quad (1)$$

using:
i) first-order forward finite difference scheme,

$$fi_j = \frac{f_{i+1} - f_j}{h}$$

ii) second-order central finite difference scheme,

$$fi_j = \frac{f_{i+1} - f_{i-1}}{2h}$$

iii) fourth-order central finite difference scheme, and

$$fi_j = \frac{f_{i-2} - 8f_{i-1} + 8f_{i+1} - f_{i+2}}{12h}$$

iv) fourth-order Padé scheme.

$$f'_{j+1} + f'_{j-1} + 4f'_j = \frac{3}{h}(f_{j+1} - f_{j-1}) + \frac{h^4}{30} f_j^{(v)}$$

For the boundaries:

$$f'_0 + 2f'_1 = \frac{1}{h}\left(-\frac{5}{2}f_0 + 2f_1 + \frac{1}{2}f_2\right)$$

$$f'_n + 2f'_{n-1} = \frac{1}{h}\left(\frac{5}{2}f_n - 2f_{n-1} - \frac{1}{2}f_{n-2}\right)$$

Consider $N = 256, 512, 1024$ and 2048, where N is the number of intervals to discretize the x domain.
For the periodic case we have the following boundary conditions:

$$f'_{N-1} + f'_1 + 4f'_0 = \frac{3}{h}(f_1 - f_{N-1})$$

$$f'_{N-1} + f'_1 + 4f'_N = \frac{3}{h}(f_1 - f_{N-1})$$

The matrix for the third order one sided boundary conditions will be

$$A = \begin{bmatrix} 1 & 2 & 0 & \cdots & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 0 & 0 & \cdots & 2 & 1 \end{bmatrix}$$

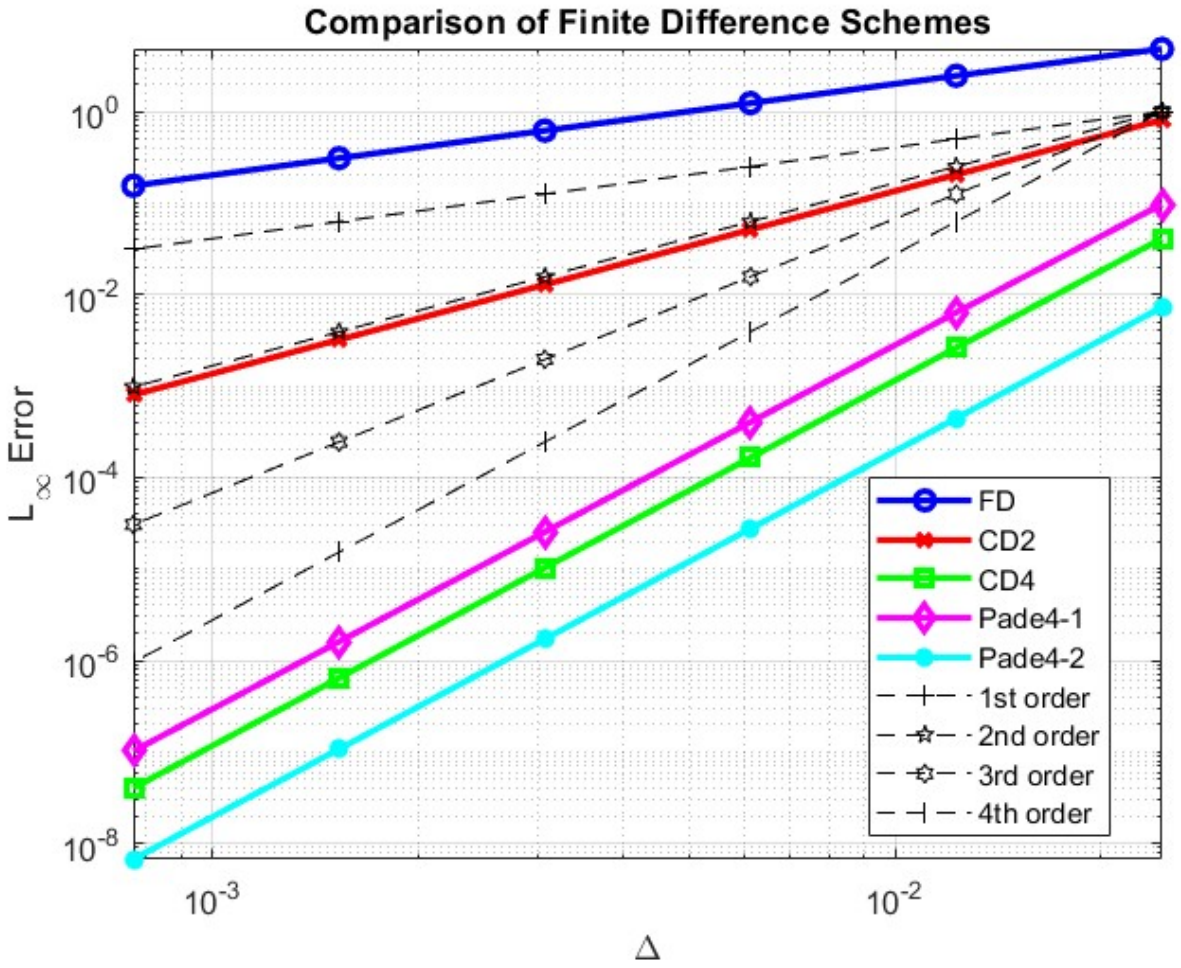
$$b = \begin{bmatrix} \frac{1}{h}\left(-\frac{5}{2}f_0 + 2f_1 + \frac{1}{2}f_2\right) \\ \frac{3}{h}(f_2 - f_1) \\ \frac{3}{h}(f_3 - f_2) \\ \vdots \\ \frac{3}{h}(f_N - f_{N-1}) \\ \frac{1}{h}\left(\frac{5}{2}f_n - 2f_{n-1} - \frac{1}{2}f_{n-2}\right) \end{bmatrix}$$

The matrix for the periodic boundary conditions will be

$$A = \begin{bmatrix} 4 & 1 & 0 & \cdots & 1 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 4 & 1 \\ 0 & 1 & 0 & \cdots & 1 & 4 \end{bmatrix}$$

$$b = \frac{3}{h} \begin{bmatrix} (f_1 - f_N) \\ (f_2 - f_1) \\ (f_3 - f_2) \\ \vdots \\ (f_N - f_{N-2}) \\ (f_0 - f_2) \end{bmatrix}$$

The Thomas Algorithm uses LU decomposition to solve tridiagonal matrices, however, for the case with the periodic boundary conditions, we have two off diagonal terms which means we cannot use it to solve this problem.



3 Problem 2.1

a.

To determine whether the given finite difference expression holds, let's apply the central finite difference operator to the expression $u_n v_n$:

$$\frac{\delta(u_n v_n)}{\delta x} = \frac{u_{n+1}v_{n+1} - u_{n-1}v_{n-1}}{2h}.$$

Expanding this expression (LHS):

$$\frac{\delta(u_n v_n)}{\delta x} = \frac{u_{n+1}v_{n+1}}{2h} - \frac{u_{n-1}v_{n-1}}{2h}.$$

Now, let's express this in terms of finite differences (RHS):

$$\frac{\delta(u_n v_n)}{\delta x} = \frac{u_{n+1} - u_{n-1}}{2h} \cdot v_n + \frac{v_{n+1} - v_{n-1}}{2h} \cdot u_n.$$

Comparing this with the analogous finite difference expression:

$$\frac{\delta(uv)}{\delta x} = u\frac{\delta v}{\delta x} + v\frac{\delta u}{\delta x},$$

we can see that the given finite difference expression does not directly match the analogous finite difference expression from calculus. Therefore, the given finite difference expression does not hold as stated.

b.

Let's evaluate the right-hand side:

$$\bar{u}_n \frac{\delta v_n}{\delta x} + \bar{v}_n \frac{\delta u_n}{\delta x}$$

Given:

$$\bar{u}_n = \frac{1}{2}(u_{n+1} + u_{n-1}), \quad \bar{v}_n = \frac{1}{2}(v_{n+1} + v_{n-1})$$

Now, let's evaluate each term:

$$\bar{u}_n \frac{\delta v_n}{\delta x} = \frac{1}{2}(u_{n+1} + u_{n-1})\frac{v_{n+1} - v_{n-1}}{2h}$$

$$= \frac{1}{4h}(u_{n+1}v_{n+1} - u_{n+1}v_{n-1} + u_{n-1}v_{n+1} - u_{n-1}v_{n-1})$$

Similarly,

$$\bar{v}_n \frac{\delta u_n}{\delta x} = \frac{1}{2}(v_{n+1} + v_{n-1})\frac{u_{n+1} - u_{n-1}}{2h}$$

$$= \frac{1}{4h}(v_{n+1}u_{n+1} - v_{n+1}u_{n-1} + v_{n-1}u_{n+1} - v_{n-1}u_{n-1})$$

Adding both terms together:

$$\frac{\delta(u_n v_n)}{\delta x} = \bar{u}_n \frac{\delta v_n}{\delta x} + \bar{v}_n \frac{\delta u_n}{\delta x} = \frac{1}{2h}(u_{n+1}v_{n+1} - u_{n-1}v_{n-1})$$

So, both sides of the equation are equal, thus confirming the validity of the given expression.

c.

Show that

$$\phi\frac{\delta\psi}{\delta x} = \frac{\delta}{\delta x}(\bar{\phi}\psi) - \overline{\psi\frac{\delta\phi}{\delta x}}$$

Starting with the RHS:

$$\frac{\bar{\phi}_{n+1}\psi_{n+1} - \bar{\phi}_{n-1}\psi_{n-1}}{2h} - \frac{1}{2}\left[\psi_{n+1}\frac{\delta\phi_{n+1}}{\delta x} + \psi_{n-1}\frac{\delta\phi_{n-1}}{\delta x}\right]$$
$$\frac{(\phi_{n+2}+\phi_n)\psi_{n+1} - (\phi_n+\phi_{n-2})\psi_{n-1}}{4h} - \frac{\psi_{n+1}(\phi_{n+1}-\phi_n) + \psi_{n-1}(\phi_n-\phi_{n-2})}{4h}$$

This reduces to a a form of the finite difference which is equal to the LHS.

$$\frac{\phi_n(\psi_{n+1}-\psi_{n-1})}{2h} = \phi\frac{\delta\psi}{\delta x}$$

d.

To derive a finite difference formula for the second-derivative operator from two applications of the first-derivative finite difference operator, let's start by defining the first-derivative finite difference operator as:

$$\frac{\delta u}{\delta x} \approx \frac{u_{i+1}-u_{i-1}}{2h} + \frac{h^2}{6}u_i'''$$

where h is the spacing between grid points.
Now, applying this operator twice to u, we get:

$$\frac{\delta^2 u}{\delta x^2} \approx \frac{\delta}{\delta x}\left(\frac{u_{i+1}-u_{i-1}}{2h} - \frac{h^2}{6}u_i'''\right)$$

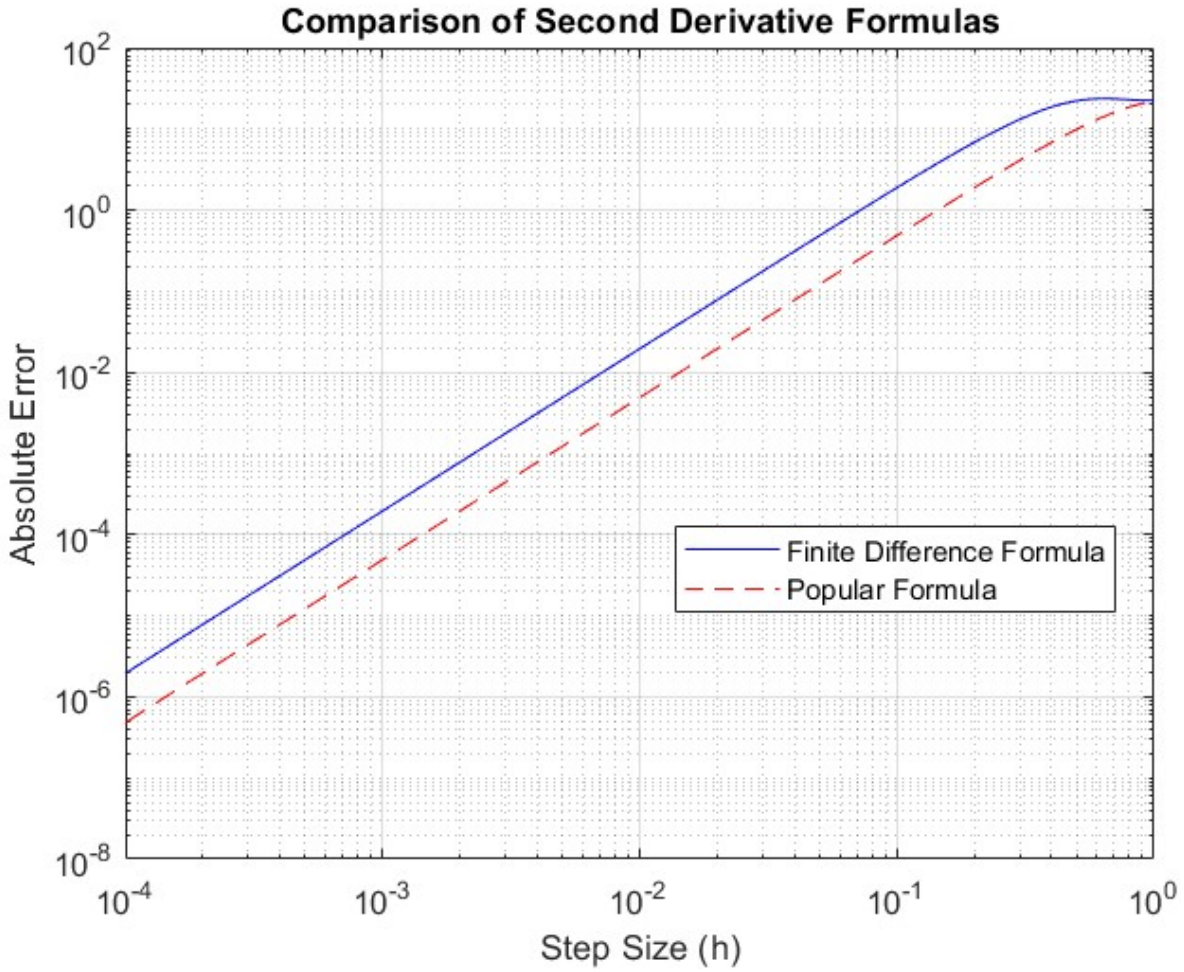
Expanding the derivative on the right-hand side:

$$\frac{\delta}{\delta x}\left(\frac{u_{i+1}-u_{i-1}}{2h}\right) \approx \frac{\left(\frac{u_{i+2}-u_i}{2h} - \frac{h^2}{6}u_{i+1}'''\right) - \left(\frac{u_i-u_{i-2}}{2h} - \frac{h^2}{6}u_{i-1}'''\right)}{2h}$$

Simplifying:

$$\frac{\delta^2 u}{\delta x^2} \approx \frac{u_{i+2}-2u_i+u_{i-2}}{4h^2} - \frac{h}{12}(u_{i+1}''' + u_{i-1}''') - \frac{1}{6}h^2 u_i^{(iv)}$$
$$\frac{\delta^2 u}{\delta x^2} \approx \frac{u_{i+2}-2u_i+u_{i-2}}{4h^2} - \frac{h^2}{6}(u_i''' + u_i^{iv})$$

This gives us the finite difference formula for the second-derivative operator using two applications of the first-derivative finite difference operator.



Looking at the graphs, we can see that they have the same slope (1.9) which means that they have the same order of accuracy of 2. Because the graphs are slightly shifted from one another, we can also deduce that the given popular equation leading term error has a magnitude smaller that the one derived (about 4 times smaller: i divided the points at the intercept to get this value). Looking at my derivation for the leading order error term, I assume I made an arithmetic error hence the two derivative terms that remain in the error.

4 Problem 2.2

Find the most accurate formula for the first derivative at x_i utilizing known values of f at x_{i-1}, x_i, x_{i+1} and x_{i+2}. The points are uniformly spaced. Give the leading error term and state the order of the method
We will begin by first creating the taylor table for this scheme

	f_i	f_i^i	f_i^{ii}	f_i^{iii}	f_i^{iv}
$a_0 f_{i-1}$	a_0	$-a_0 h$	$\frac{1}{2}a_0 h^2$	$-\frac{1}{6}a_0 h^3$	$\frac{1}{24}a_0 h^4$
$a_1 f_i$	a_1	0	0	0	0
$a_2 f_{i+1}$	a_2	$a_2 h$	$\frac{1}{2}a_2 h^2$	$\frac{1}{6}a_2 h^3$	$\frac{1}{24}a_2 h^4$
$a_3 f_{i+2}$	a_3	$2a_3 h$	$2a_3 h^2$	$\frac{4}{3}a_3 h^3$	$\frac{2}{3}a_3^4$
f_i^i	0	1	0	0	0

$$a_0 f_{i-1} + a_1 f_i + a_2 f_{i+1} + a_3 f_{i+2} = (a_0 + a_1 + a_2 + a_3) f_i + (-a_0 + a_2 + 2a_3) h f_i^i + (\frac{1}{2}a_0 + \frac{1}{2}a_2 + 2a_3) h^2 f_i^{ii} + (-\frac{1}{6}a_0 + \frac{1}{6}a_2 + \frac{4}{3}a_3) h^3 f_i^{iii} + (\frac{1}{24}a_0 + \frac{1}{24}a_2 + \frac{2}{3}a_3) h^4 f_i^{iv}$$

We want the coefficient of the first derivative to be equal to 1 and have the rest equal to zero

$$\begin{aligned} a_0 + a_1 + a_2 + a_3 &= 0 \\ a_0 + 0 + a_2 + 4a_3 &= 0 \\ -a_0 + 0 + a_2 + 2a_3 &= 1 \\ -a_0 + 0 + a_2 + 8a_3 &= 0 \end{aligned}$$

Solving this simultaneously, we get

$$\begin{aligned} a_0 &= -\frac{1}{3h} \\ a_1 &= -\frac{1}{2h} \\ a_2 &= \frac{1}{h} \\ a_3 &= -\frac{1}{6h} \end{aligned}$$

We plug these values back into our combined equation

$$-\frac{1}{3h} f_{i-1} - \frac{1}{2h} f_i + \frac{1}{h} f_{i+1} - \frac{1}{6h} f_{i_2} = h f_i^i - \frac{1}{12} h^4 f_i^{iv}$$

$$f_i^i = \frac{-2f_{i-1} - 3f_i + 6f_{i+1} - f_{i+2}}{6h^2} + O(h^3)$$

The leading error term is given by $\frac{-1}{12} h^3 f_i^{iv}$ and the order of this method is third order accurate

5 Problem 2.3

Verify that the modified wavenumber for the fourth-order Pade' scheme for the first derivative is

$$k' = \frac{3\sin(k\Delta)}{\Delta(2 + \cos(k\Delta)} \tag{1}$$

The fourth order Pade' formula for numerical differentiation is given below in the textbook as described by equation 2.16 in the textbook

$$f'_{j+1} + f'_{j-1} + 4f'_j = \frac{3}{\Delta}(f_{j+1} + f_{j-1}) + \frac{\Delta^4}{30} f_j^v$$

$$f(x) = e^{ikx}$$

$$f'(x) = ike^{ikx}$$

$$ike^{ikx_{j+1}} + ike^{ikx_{j-1}} + 4ike^{ikx_j} = \frac{3}{\Delta}(e^{ikx_{j+1}} - e^{ikx_{j-1}})$$

We then collect all the ik terms to one side of the equation with h equal to Δ

$$ike^{ik(x+\Delta)} + ike^{ik(x-\Delta)} + 4ike^{ikx} = \frac{3}{\Delta}(e^{ik(x+\Delta)} - e^{ik(x+\Delta)})$$

$$ike^{ikx}(e^{ik\Delta} + e^{-ik\Delta} + 4) = \frac{3}{\Delta}e^{ikx}(e^{ik\Delta} - e^{-ik\Delta})$$

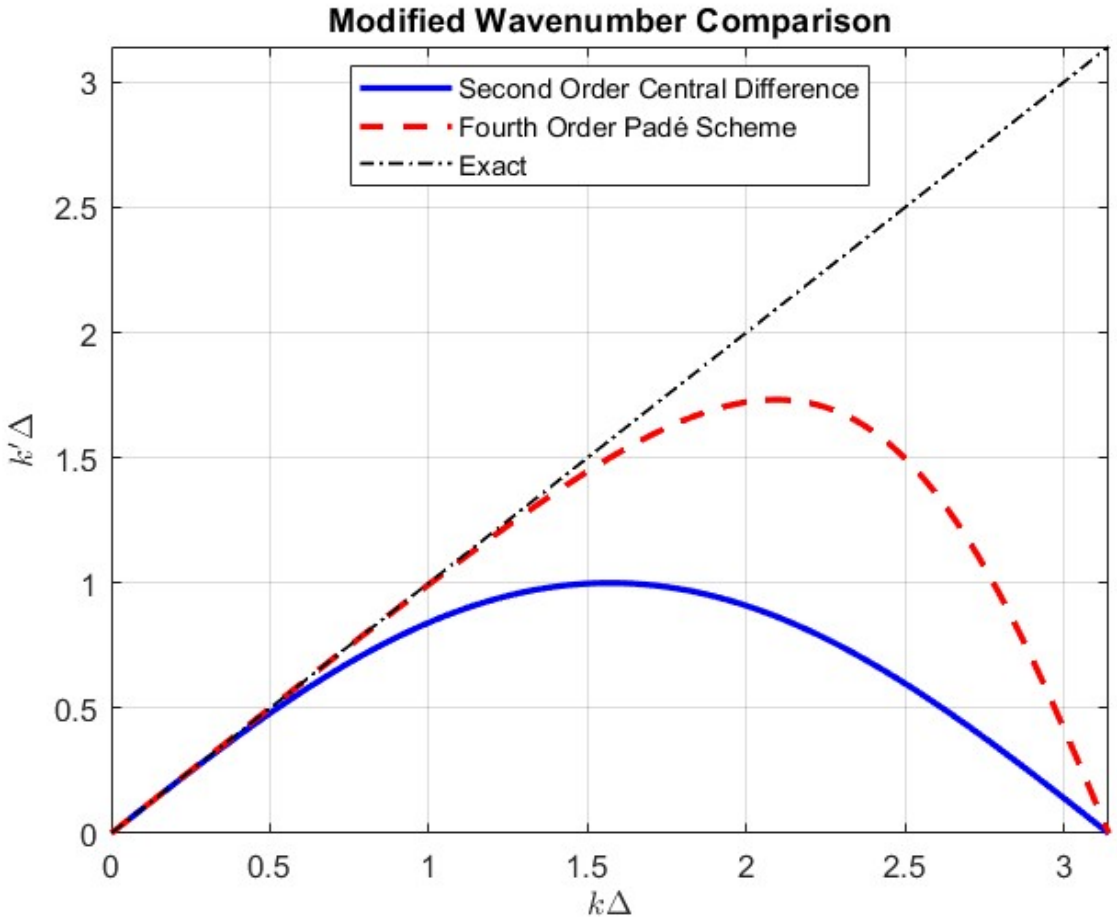
$$\cos(x) = \frac{e^{ix} + e^{-ix}}{2}$$

$$\sin(x) = \frac{e^{ix} - e^{-ix}}{2i}$$

$$ik(2\cos(k\Delta) + 4) = \frac{3}{\Delta}(2i\sin(k\Delta))$$

$$ik = \frac{3(2i\sin(k\Delta))}{\Delta(2\cos(k\Delta) + 4)}$$

$$k' = \frac{3\sin(k\Delta)}{\Delta(2 + \cos(k\Delta))}$$



6 Problem 2.4

A general Pade' type boundary scheme (at i=0) for the first derivative which does not alter the tri-diagonal structure of the matrix in equation 2.16 can be written as

$$f'_0 + \alpha f'_1 = \frac{1}{h}(af_0 + bf_1 + cf_2 + df_3) \tag{2}$$

a. Show that requiring this scheme to be at least third order accurate would constrain the coefficients to

$$a = -\frac{11+2\alpha}{6}, \qquad b = \frac{6-\alpha}{2}, \qquad c = \frac{2\alpha-3}{2}, \qquad d = \frac{2-\alpha}{6}$$

b. Find all the coefficients such that the scheme would be fourth-order accurate.

We will first create a Taylor table evaluated about f_0 We can then sum up the coefficients and then set them to

	f_0	f_0^i	f_0^{ii}	f_0^{iii}	f_0^{iv}	f_0^v
f_0^i	0	1	0	0	0	0
αf_1^i	0	α	αh	$\frac{1}{2}\alpha h^2$	$\frac{1}{6}\alpha h^3$	$\frac{1}{24}\alpha h^4$
af_0	a	0	0	0	0	0
bf_1	b	bh	$\frac{1}{2}bh^2$	$\frac{1}{6}bh^3$	$\frac{1}{24}bh^4$	$\frac{1}{120}bh^5$
cf_2	c	2ch	$2ch^2$	$\frac{4}{3}ch^3$	$\frac{2}{3}ch^4$	$\frac{4}{15}ch^5$
df_3	d	3dh	$\frac{9}{2}dh^2$	$\frac{9}{2}dh^3$	$\frac{27}{8}dh^4$	$\frac{81}{40}dh^5$

zero so that we can get rid of the lower order terms, giving us higher accuracy.

$$\begin{gathered} a+b+c+d=0 \\ b+2c+3d=1+\alpha \\ \frac{1}{2}bh+2ch+\frac{9}{2}dh+\alpha h=0 \\ \frac{1}{6}bh^2+\frac{4}{3}ch^2+\frac{9}{2}dh^2+\frac{1}{2}\alpha h^2=0 \end{gathered}$$

Rewriting the equations we get

$$\begin{gathered} a=-(b+c+d) \\ b+2c+3d=1+\alpha \\ b+4c+9d=2\alpha \\ b+8c+27d=3\alpha \end{gathered}$$

We can then solve the equations simultaneously

$$\begin{gathered} b=-4c-9d+2\alpha \\ (-4c-9d+2\alpha)+2c+3d=1+\alpha \\ (-4c-9d+2\alpha)+8c+27d=3\alpha \end{gathered}$$

Simplify further to

$$\begin{gathered} -2c-6d=1-\alpha \\ 4c+18d=\alpha \end{gathered}$$

Multiply the first equation by 2 and add

$$\begin{gathered} -4c-12d=2-2\alpha \\ 4c+18d=-\alpha \end{gathered}$$

This gives

$$\begin{gathered} 6d=2-\alpha \\ d=\frac{2-\alpha}{6} \end{gathered} \qquad \text{(d equation)}$$

We plug this to get the value of c

$$4c + 18(\frac{2-\alpha}{6}) = -\alpha$$
$$4c + 6 - 3\alpha = \alpha$$
$$4c = 4\alpha - 6$$
$$c = \frac{2\alpha - 3}{2} \quad \text{(c equation)}$$

We can then solve for b and a

$$b = -4(\frac{2\alpha-3}{2}) - 9(\frac{2-\alpha}{6}) + 2\alpha$$
$$b = 2\alpha - 4\alpha + 6 - 3 + \frac{3\alpha}{2}$$
$$b = \frac{6-\alpha}{2} \quad \text{(b equation)}$$

$$a = -(\frac{6-\alpha}{2} + \frac{2\alpha-3}{2} + \frac{2-\alpha}{6})$$
$$a = -(\frac{18 - 3\alpha + 6\alpha - 9 + 2 - \alpha}{6}$$
$$a = -\frac{11+2\alpha}{6} \quad \text{(a equation)}$$

In order to make this fourth order accurate, we want the coefficients of the fourth derivative to be equal to zero as well. This gives us:

$$\frac{1}{6}\alpha = \frac{1}{24}b + \frac{2}{3}c + \frac{27}{8}d$$
$$4\alpha = b + 16c + 81d$$
$$4\alpha = \frac{6-\alpha}{2} + 16(\frac{2\alpha-3}{2}) + 81(\frac{2-\alpha}{6})$$
$$\alpha = 3$$

We can then plug this alpha value back into our a,b,c and d equations

$$a = \frac{-17}{6}$$
$$b = \frac{3}{2}$$
$$c = \frac{3}{2}$$
$$d = -\frac{1}{6}$$

7 Problem 2.8

a.

For the left boundary, derive a third-order Pade' scheme to approximate y_0'' in the following form:

$$y_1'' + b_2 y_2'' = a_1 y_1 + a_2 y_2 + a_3 y_3 + a_4 y_b' + O(h^3)$$

	y_1	y_1^i	y_1^{ii}	y_1^{iii}	y_1^{iv}	y_1^v
a_1y_1	a_1	0	0	0	0	0
a_2y_2	a_2	a_2h	$\frac{1}{2}a_2h^2$	$\frac{1}{6}a_2h^3$	$\frac{1}{24}a_2h^4$	$\frac{1}{120}a_2h^5$
a_3y_3	a_3	$2a_3h$	$\frac{4}{2}a_3h^2$	$\frac{8}{6}a_3h^3$	$\frac{16}{24}a_3h^4$	$\frac{32}{120}a_3h^5$
a_4y_b'	0	a_4	$\frac{-h}{2}a_4$	$\frac{1}{2}a_4(\frac{-h}{2})^2$	$\frac{1}{6}a_4(\frac{-h}{2})^3$	$\frac{1}{24}a_4(\frac{-h}{2})^4$
y_1''	0	0	1	0	0	0
b_2y_2	0	0	b_2	b_2h	$\frac{1}{2}b_2h^2$	$\frac{1}{6}b_2h^2$

We have 5 unknowns so we need 5 equations to solve this problem

$$a_1 + a_2 + a_3 = 0$$
$$a_2h + 2a_3h + a_4 = 0$$
$$\frac{1}{2}a_2h^2 + \frac{4}{2}a_3h^2 - \frac{a_4h}{2} = 2 + 2b_2$$
$$\frac{1}{6}a_2h^3 + \frac{8}{6}a_3h^3 + \frac{1}{8}a_4h^2 = b_2h$$
$$\frac{1}{24}a_2h^4 + \frac{16}{24}a_3h^4 - \frac{1}{48}a_4h^3 = \frac{1}{2}b_2h^2$$

These equations can be solved using the solve function in Matlab giving results in terms of h. See Appendix **??**

$$a_1 = \frac{-36}{23h^2}$$
$$a_2 = \frac{48}{23h^2}$$
$$a_3 = \frac{-12}{23h^2}$$
$$a_4 = \frac{-24}{23h^2}$$
$$b_2 = \frac{-11}{23}$$

We then obtain a final equation for the left boundary condition:

$$y_1'' - \frac{11}{23}y_2'' = -\frac{36}{23h^2}y_1 + \frac{48}{23h^2}y_2 - \frac{12}{23h^2}y_3 - \frac{24}{23h^2}y_b'$$
$$y_1'' - \frac{11}{23}y_2'' = \frac{12}{23h^2}[-3y_1 + 4y_2 - y_3 - 2hy_b']$$

b.

For the right boundary, we repeat the same process but instead take positive $h/2$ for the boundary term. This time we will evaluate our taylor series about y_N

$$a_1 + a_2 + a_3 = 0$$
$$-a_2h - 2a_3h + a_4 = 0$$
$$\frac{1}{2}a_2h^2 + \frac{4}{2}a_3h^2 + \frac{a_4h}{2} = 2 + 2b_2$$
$$-\frac{1}{6}a_2h^3 - \frac{8}{6}a_3h^3 + \frac{1}{8}a_4h^2 = -b_2h$$
$$\frac{1}{24}a_2h^4 + \frac{16}{24}a_3h^4 + \frac{1}{48}a_4h^3 = \frac{1}{2}b_2h^2$$

	y_N	y_N^i	y_N^{ii}	y_N^{iii}	y_N^{iv}	y_N^v
$a_1 y_N$	a_1	0	0	0	0	0
$a_2 y_{N-1}$	a_2	$-a_2 h$	$\frac{1}{2}a_2 h^2$	$-\frac{1}{6}a_2 h^3$	$\frac{1}{24}a_2 h^4$	$-\frac{1}{120}a_2 h^5$
$a_3 y_{N-2}$	a_3	$-2a_3 h$	$\frac{4}{2}a_3 h^2$	$-\frac{8}{6}a_3 h^3$	$\frac{16}{24}a_3 h^4$	$-\frac{32}{120}a_3 h^5$
$a_4 y_b'$	0	a_4	$\frac{h}{2}a_4$	$\frac{1}{2}a_4(\frac{h}{2})^2$	$\frac{1}{6}a_4(\frac{h}{2})^3$	$\frac{1}{24}a_4(\frac{h}{2})^4$
y_N''	0	0	1	0	0	0
$b_2 y_{N-1}$	0	0	b_2	$-b_2 h$	$\frac{1}{2}b_2 h^2$	$-\frac{1}{6}b_2 h^2$

$$a_1 = \frac{-36}{23h^2}$$
$$a_2 = \frac{48}{23h^2}$$
$$a_3 = \frac{-12}{23h^2}$$
$$a_4 = \frac{24}{23h^2}$$
$$b_2 = \frac{-11}{23}$$

$$y_N'' - \frac{11}{23}y_{N-1}'' = \frac{12}{23h^2}[-3y_N + 4y_{N-1} - y_{N-2} + 2hy_b']$$

c.

What are the elements of the matrices A and B operating on the interior and boundary nodes?

$$A\begin{bmatrix} y_1'' \\ y_2'' \\ \vdots \\ y_{N-1}'' \\ y_N'' \end{bmatrix} = B\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix}$$

For the interior nodes, we have the following equation:

$$\frac{1}{12}f_{i-1}'' + \frac{10}{12}f_i'' + \frac{1}{12}f_{i+1}'' = \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2}$$

The matrices A and B will have the following form

$$A = \begin{bmatrix} 1 & \frac{-11}{23} & 0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\ \frac{1}{12} & \frac{10}{12} & \frac{1}{12} & \cdots & \cdots & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{12} & \frac{10}{12} & \frac{1}{12} & 0 & 0 & \cdots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{1}{12} & \frac{10}{12} & \frac{1}{12} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{12} & \frac{10}{12} & \frac{1}{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-11}{23} & 1 \end{bmatrix}$$

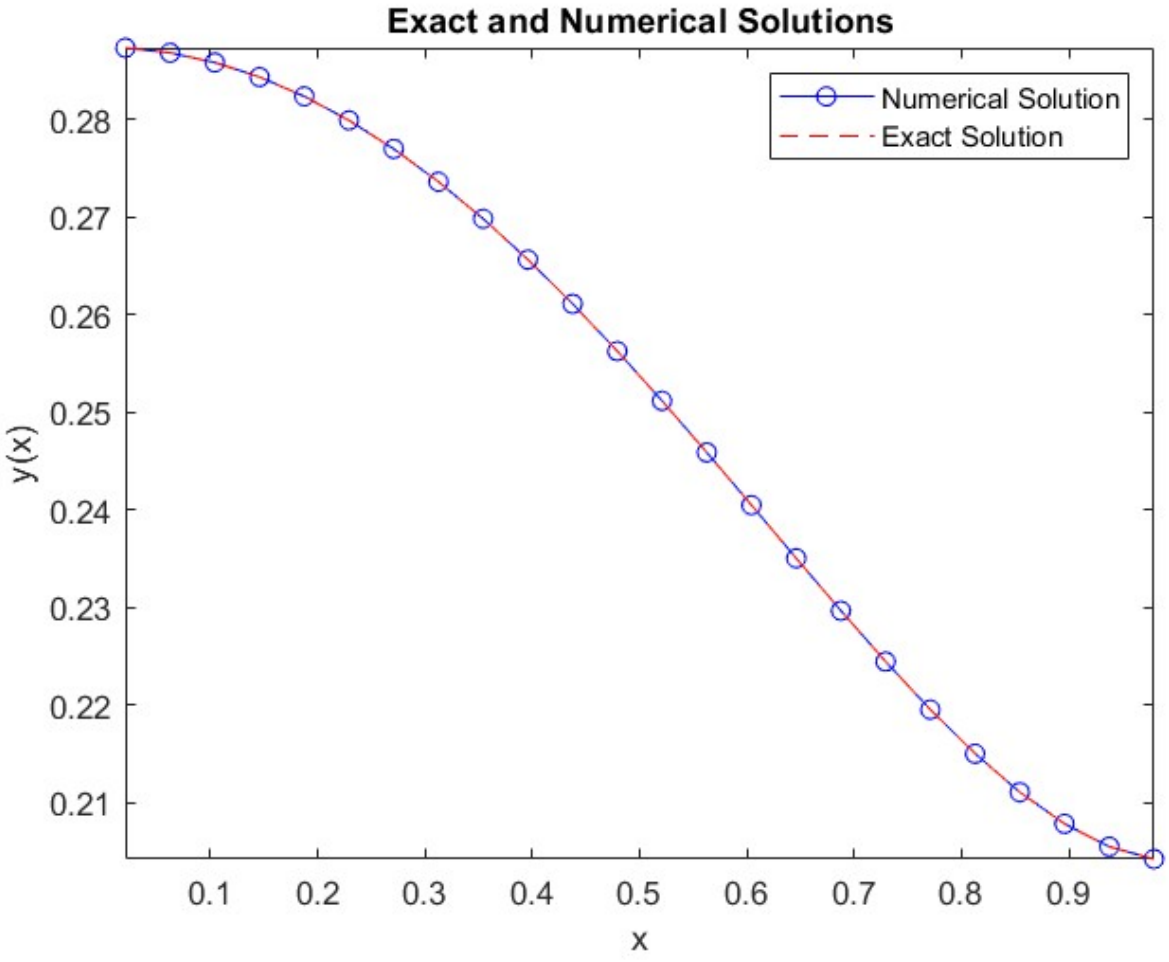
$$B = \frac{1}{h^2}\begin{bmatrix} \frac{-36}{23} & \frac{48}{23} & \frac{-12}{23} & \cdots & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & \frac{-12}{23} & \frac{48}{23} & \frac{-36}{23} \end{bmatrix}$$

d.

Use this relationship to transform the ODE into a system with y_i's as unknowns. Use N = 24 and solve this system. Do you actually have to invert A? Plot the exact and numerical solutions. Discuss your result. How are the Neumann boundary conditions enforced into the discretized boundary value problem?

$$\frac{\partial^2 y}{\partial x^2} + y = x^3$$

We have an equation of the form $Ay'' = By$ and we can substitute this into our ODE $Ay'' + Ay = Ax^3$. This gives us $Ay + By = Ax^3$, simplifying this further we get $(A + B)y = Ax^3$. To solve this system of equations, we define a new matrix $C = A + B$ and a vector b representing the right-hand side function x^3. Thus, we have the system $Cy = b$ which can be solved for y. It's important to note that we don't necessarily need to invert the matrix C explicitly. We can use Matlab's backslash operator can efficiently solve the linear system.



The matrices A and B are constructed to include the boundary conditions in their entries, ensuring that the correct boundary conditions are incorporated into the discretized problem.

8 Problem 2.10

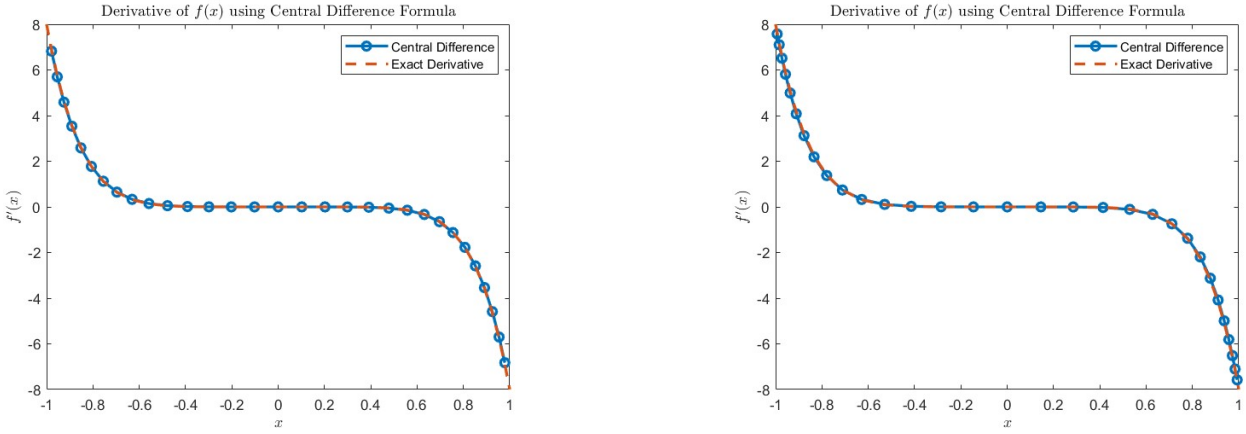
a.

Consider the function $f(x) = 1 - x^8$ and a grid defined as follows:

$$\begin{cases} j = 0, 1, 2, \ldots, N \\ \xi_j = -1 + 2j/N \\ x_j = \frac{1}{a} \tanh(\xi_j \tanh^{-1}[a]), \quad 0 < a < 1 \end{cases}$$

The parameter a can be used to adjust the spacing of the grid points, with larger a placing more points near the boundaries. For this problem, take $a = 0.98$ and $N = 32$.
(a) Compute and plot the derivative of f with the central difference formula (2.20) and the coordinate transformation method described in Section 2.5 and compare with the exact derivative in $-1 \leq x < 1$. How would the results change with $a = 0.9$?
Using the central difference formula

$$f' = \frac{f_{j+1} - f_{j-1}}{x_{j+1} - f_{x-1}}$$



(a) a =0.9 **(b) a =0.98**

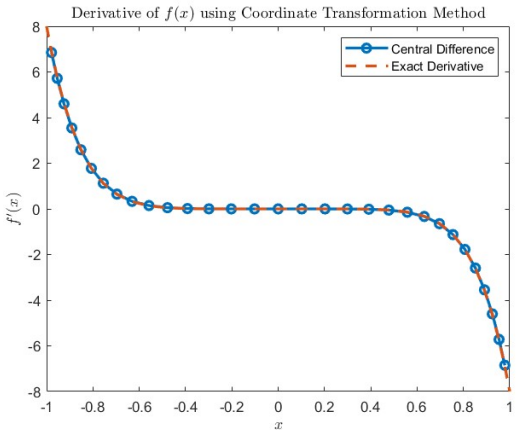
Figure 1: Non-Uniform Grids using center difference

For the coordinate transform:

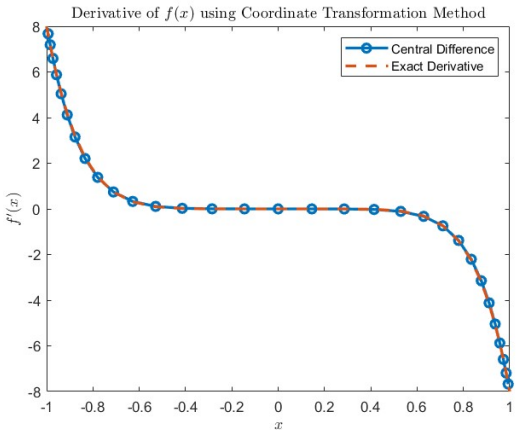
$$\frac{df}{dx} = \frac{d\xi}{dx}\frac{df}{d\xi}$$

where

$$\frac{df}{d\xi} = \frac{f_{j+1} - f_{j-1}}{2\Delta\xi}$$
$$\frac{d\xi}{dx} = \frac{a}{\arctan(a)(1 - (ax)^2)}$$



(a) a =0.9



(b) a =0.98

Figure 2: Non-Uniform Grids using Coordinate Transform

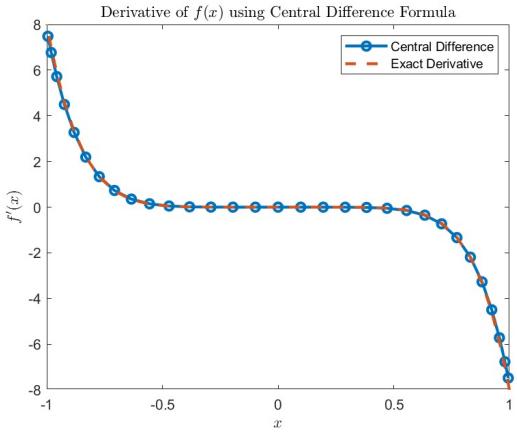
Comparing the two methods, there is no discernible difference in the graphs, even when varying the value of a from 0.9 to 0.98. Upon calculating the L_∞ error, it becomes apparent that as we decrease the value of a, the error diminishes. It is noteworthy that the error resulting from the coordinate transformation method is consistently lower than that from the finite difference method.

b.

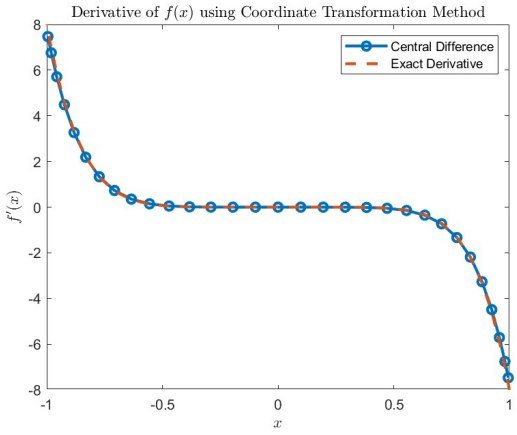
Repeat part (a) with the transformation:

$$\begin{cases} j = 0, 1, 2, \ldots, N \\ \xi_j = \frac{\pi j}{N} \\ x_j = \cos(\xi_j) \end{cases}$$

We repeat the same procedure as above but only $\frac{d\xi}{dx}$ changes to $\frac{-1}{\sqrt{1-x^2}}$. The graphs produces look the same however, the coordinate transform method had a larger error than previously. The method used in a is more accurate hence the preferred method



(a) Center Difference



(b) Coordinate Transform

Figure 3: Non-Uniform Grids using Center Difference and Coordinate Transform

c.

How many uniformly spaced grid points would be required to achieve the same accuracy as the transformation method in (a)? The maximum error in the derivative over the domain for the uniform case should be less than or equal to the maximum error over the domain for the transformed case.

Using trial and error, I changed the number of panels but increasing their value. At about 75 panels, the error was 0.0508 which was the exact value obtained in a when $a = 0.98$. Hence I would approximate 76 grid points.

9 Problem 3.2

Show that

$$\sum_{i=1}^{N-1} u_i \frac{\partial v}{\partial x}\bigg|_i = -\sum_{i=1}^{N-1} v_i \frac{\partial u}{\partial x}\bigg|_i + Boundary - Terms$$

Where $\frac{\partial}{\partial x}\big|_i$ = the second-order central difference operator. From the LHS

$$\frac{\partial v}{\partial x}\bigg|_i = \frac{v_{i+1} - v_{i-1}}{2h}$$

We expand our LHS we get:

$$\sum_{i=1}^{N-1} u_i \frac{\partial v}{\partial x}\bigg|_i = \frac{1}{2h}[u_1(v_2 - v_0) + u_2(v_3 - v_1) + u_3(v_4 - v_2) + ... + u_{N-2}(v_{N-1} - v_{N-3}) + u_{N-1}(v_N - v_{N-1})]$$

Our summation bounds are $i = 1$ to $N - 1$ so any term with $i = 0$ or $i = N$ are our boundary terms.

$$\frac{1}{2h}[-v_0u_1 - v_1u_2 - v_2(u_3 - u_1) - v_3(u_4 - u_2) + ... - v_{N-2}(u_{N-1} - u_{N-3}) + v_{N-1}u_{N-2} + v_Nu_{N-1}]$$

All the grouped terms are in the form of the second order central difference, just as we started but in this case the operator is acting in u rather than v. Thsi gives us;

$$\sum_{i=1}^{N-1} u_i \frac{\partial v}{\partial x}\bigg|_i = -\sum_{i=1}^{N-1} v_i \frac{\partial u}{\partial x}\bigg|_i + \frac{1}{2h}[-v_0u_1 - v_1u_2 + v_{N-1}u_{N-2} + v_Nu_{N-1}]$$

In comparison to integration by parts;

$$\sum_{i=1}^{N-1} u_i \frac{\partial v}{\partial x}\bigg|_i = \int_1^{N-1} u\frac{dv}{dx}dx = uv|_i^{N-1} - \int_1^{N-1} v\frac{du}{dx}dx$$

We can then write this in a discretized form

$$\sum_{i=1}^{N-1} u_i \frac{\partial v}{\partial x}\bigg|_i = -\sum_{i=1}^{N-1} v_i \frac{\partial u}{\partial x}\bigg|_i + uv|_i^{N-1}$$

This equation is of the same form as our given equation with the bounds in the case being a multiple of u and v evaluated at the end points

10 Problem 3.3

Using the error analysis for the trapezoidal and rectangle rules, show that Simpson's rule for integration over the entire interval is fourth-order accurate.
The rectangle and trapezoidal rule are evaluated over an x_i and x_{i+1} and are both third order accurate

$$\int_{x_i}^{x_{i+1}} f(x)dx = h_i f(y_i) + \frac{h_i^3}{24}f''(y_i) + \frac{h_i^5}{1920}f^{iv}(y_i) + ... \qquad \text{(Midpoint Rule)}$$

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}[f(y_i) + f(y_{i+1})] - \frac{h_i^3}{12}f''(y_i) + \frac{h_i^5}{480}f^{iv}(y_i) + ... \qquad \text{(Trapezoid Rule)}$$

Simpsons rule is evaluated over x_i and x_{i+2} so the number of divisions increases by a factor of 2, allowing us to plug in 2h in place of h

$$\int_{x_i}^{x_{i+2}} f(x)dx = \begin{cases} 2h_i f(y_i) + \frac{h_i^3}{3} f''(y_i) + \frac{h_i^5}{60} f^{iv}(y_i) \\ h[f(y_i) + f(y_{i+1})] - \frac{2h_i^3}{3} f''(y_i) + \frac{h_i^5}{15} f^{iv}(y_i) \end{cases}$$

We want to get rid of the second derivative term to achieve a higher accuracy so we multiply the first equation by 2 and add the two equations dividing 3 since $2I + I$ is our result.

$$\int_{x_i}^{x_{i+2}} f(x)dx = \frac{h}{3}[f_i + 4f_{i+1} + f_{i+2}] - \frac{h^5}{60} f_{i+1}^{iv} \tag{3}$$

The local accuracy for this equation in fifth order accuracy so the global accuracy over the whole interval is fourth order

11 Problem 3.5

Explain why the rectangle and trapezoidal rules can integrate a straight line exactly and the Simpson's rule can integrate a cubic exactly.
Both the Trapezoidal Rule and Rectangle Rule exhibit third-order accuracy, as demonstrated in the previous question. These methods utilize two points for integration, resulting in quadratic approximations when applied to linear functions. In cases of exact integration, any error term diminishes to zero. Similarly, Simpson's Rule integrates over a quadratic curve, yielding a cubic function as a result. This approach provides exact integration for cubic functions, aligning with the leading order term of the polynomial approximation

12 Problem 3.6

We have been asked to find a way of solving the Fredholm equation where $K(x,t)$ and $f(x)$ are known

$$f(x) = \phi(x) + \int_a^b K(x,t)\phi(t)dt$$

As we discussed in class, we can use the Trapezoid Rule to solve this discrete and solve this problem. By comparison, the bounds a and b are equal to x_1 and x_N

$$I = \int_a^b f(x)dx = \frac{h}{2}\left[f(a) + f(b) + 2\sum_{j=2}^{n-1} f_j\right] \tag{4}$$

$$f(x) = \phi(x) + \int_a^b K(x,t)\phi(t)dt$$

This gives us a discretized form of the Fredholm equation

$$f(x_i) = \phi(x_i) + \frac{h}{2}[K(x_i,x_1)\phi(x_1) + K(x_i,x_N)\phi(x_N) + 2\sum_{j=2}^{N-1} K(x_i,x_j)\phi(x_j)]$$

If we put this into matrix form, we will get an equation of the form $Ax = b$ i.e $A\phi = f$

$$A = \begin{bmatrix} 1 + \frac{h}{2}K(x_1,x_1) & hK(x_1,x_2) \cdots\cdots\cdots\cdots hK(x_1,x_{N-1}) & \frac{h}{2}K(x_1,x_N) \\ \frac{h}{2}K(x_2,x_1) & 1 + hK(x_2,x_2) \cdots\cdots\cdots hK(x_2,x_{N-1}) & \frac{h}{2}K(x_2,x_N) \\ \vdots & \vdots \quad \ddots \quad \vdots & \vdots \\ \frac{h}{2}K(x_{N-1},x_1) & hK(x_N-1,x_2)\cdots\cdots 1 + hK(x_{N-1},x_{N-1}) & \frac{h}{2}K(x_N-1,x_N) \\ \frac{h}{2}K(x_N,x_1) & hK(x_N,x_2)\cdots\cdots\cdots\cdots hK(x_N,x_{N-1}) & 1 + \frac{h}{2}K(x_N,x_N) \end{bmatrix}$$

Our f matrix will be of the form

$$f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-1} \\ f_N \end{bmatrix}$$

To solve the equation below, we need to rearrange it into the form described above where our $f(x) = \pi x^2$

$$\phi(x) = \pi x^2 + \int_0^{\pi} 3(0.5\sin(3x) - tx^2)\phi(t)dt$$

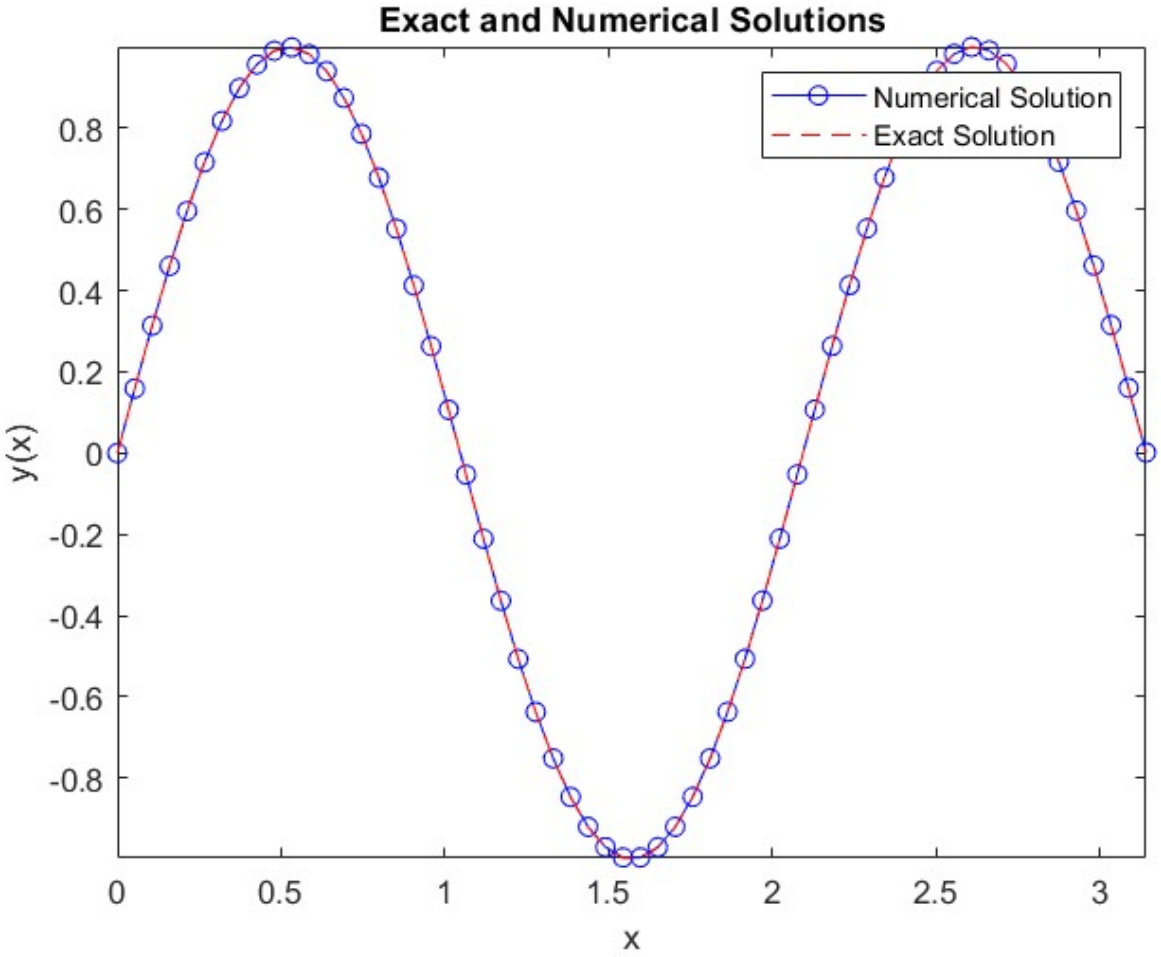


Figure 4: Solution to $\cos(3x)$

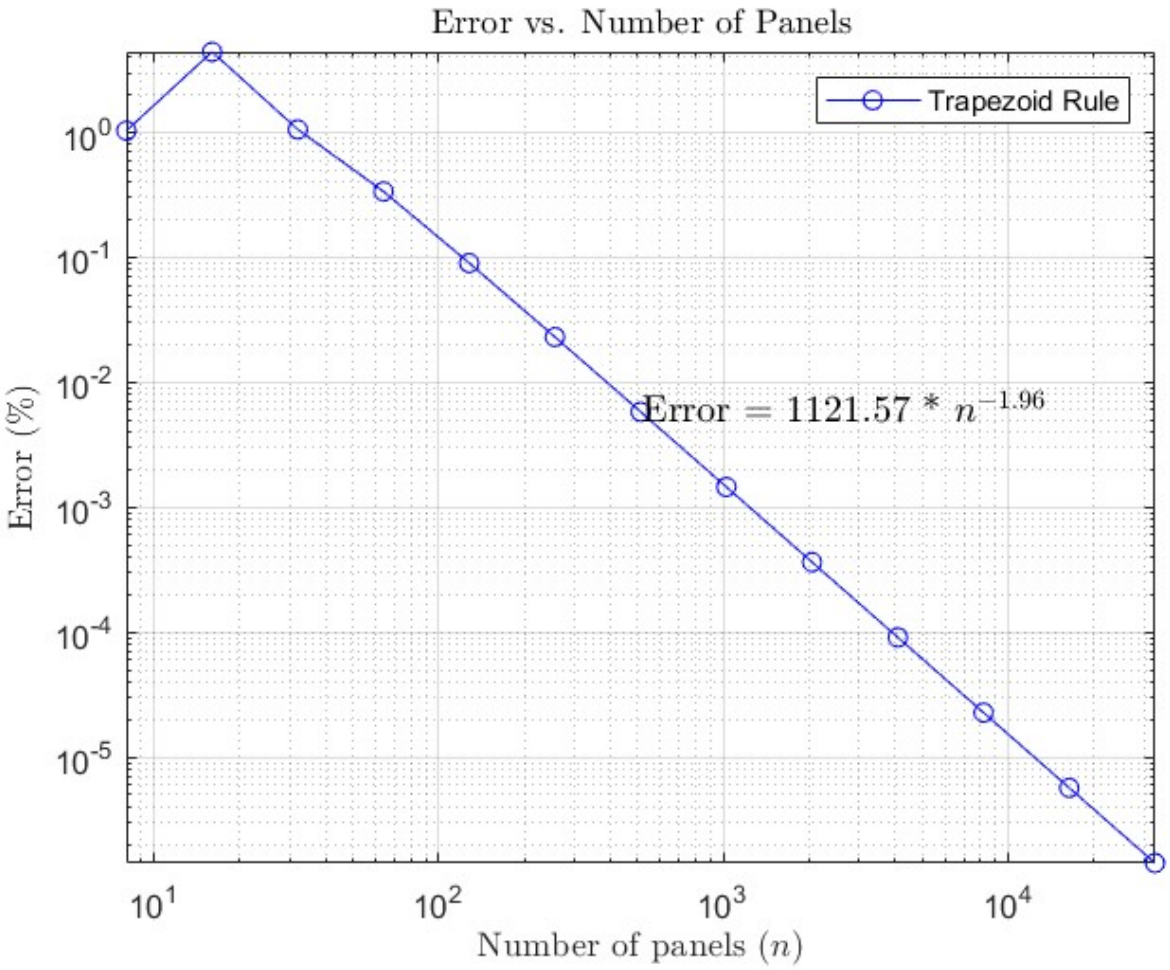
13 Problem 3.8

Numerically evaluate this integral using the trapezoidal rule with n panels of uniform length h. Make a log–log plot of the percent error vs. n and discuss the accuracy of the method. Take n = 8, 16, 32 etc

$$\int_0^1 = \left[\frac{100}{\sqrt{x+0.01}} + \frac{1}{(x-0.3)^2+0.001} - \pi\right]$$

The Trapezoidal Rule is a method used to approximate the value of a definite integral by dividing the interval of integration into smaller segments and approximating the area under the curve within each segment as shown in

Eq. (4). We evaluate from $j = 2$ to $j = n$ which is the total number of panels so the number of grid points will be $n + 1$

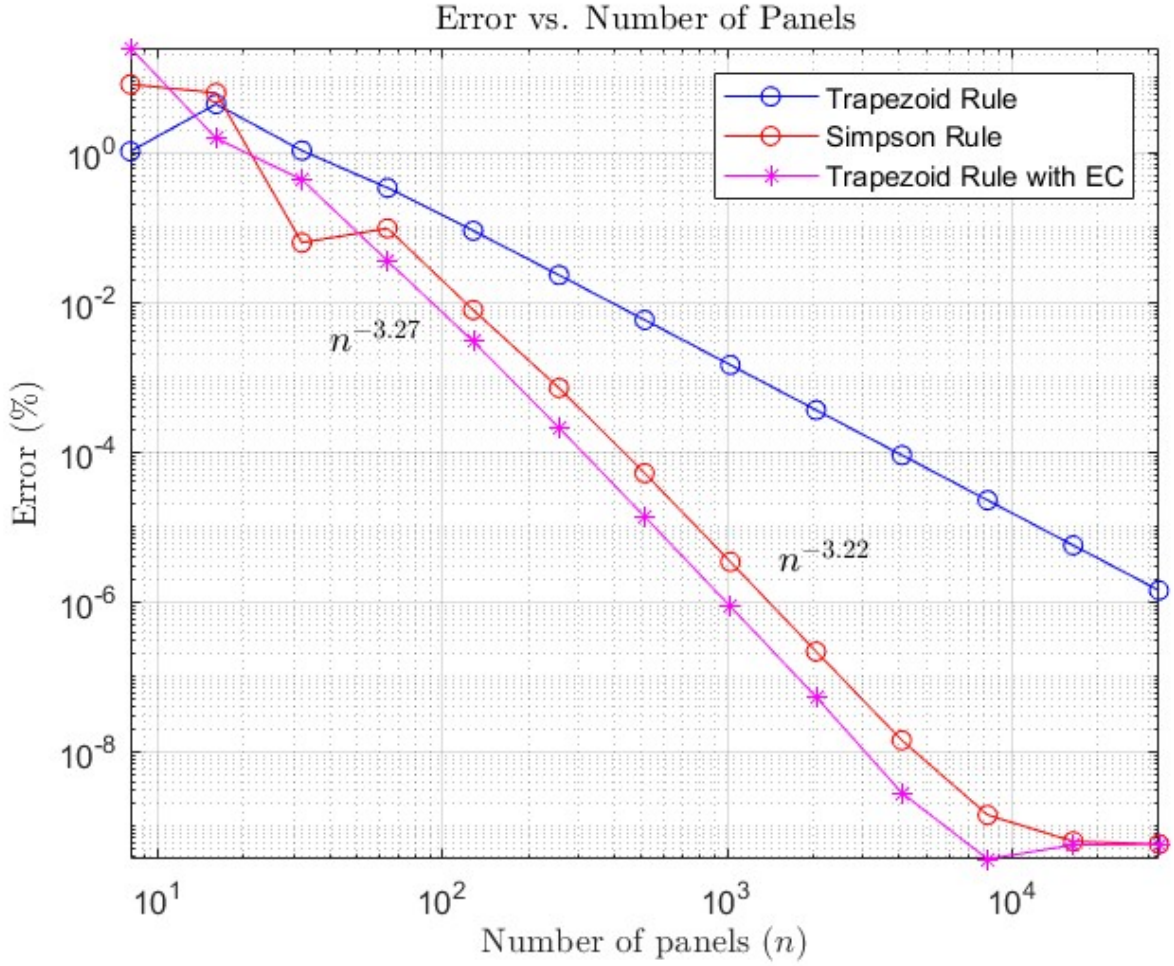


Ideally, as n increases (meaning more panels are used to approximate the integral), we expect the error percentage to decrease, indicating higher accuracy. If the trapezoidal rule accurately approximates the integral, we should see a decrease in error percentage with increasing n. In the log-log plot, we see a straight line with negative slope indicating that the error decreases as n increases, suggesting that the method converges towards the true value of the integral as the number of panels increases. The rate at which the error decreases with increasing n provides insight into the order of accuracy of the trapezoidal rule. For example, if the error decreases by a factor of n^{-2}, it indicates second-order accuracy; we have -1.96 which is approximately -2 i.e. a global accuracy of 2
Now we repeat using the Simpson's Rule (Eq. (3)) and trapezoidal rule with end correction. However we can also rewrite the Simpson's Rule as the following by separating into even and odd terms

$$I = \frac{h}{3}\left[f_0 + f_n + 4\sum_{\substack{j=1 \\ j=\text{odd}}}^{n-1} f_j + 2\sum_{\substack{j=2 \\ j=\text{even}}}^{n-2} f_j\right]$$

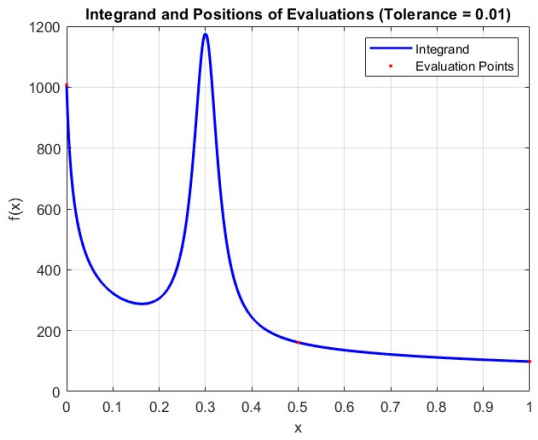
As for the trapezoidal rule with end correction, we have new term added

$$\frac{-h^2}{12}[f'(b) - f'(a)]$$

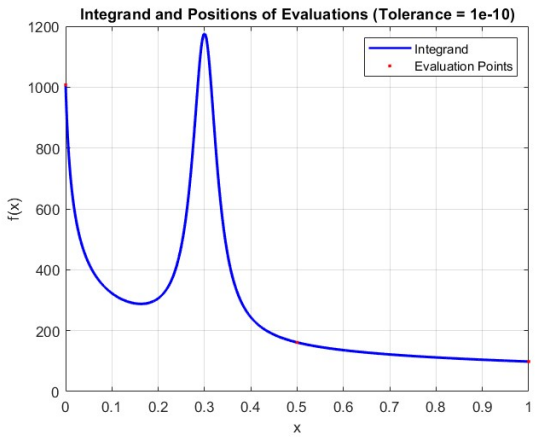


From the slope of the lines we can confirm that the global accuracy for the simpsons and trapezoidal rule is 3 from them having the same slope. Lastly, we have to perform adaptive methods to solve the integral. Adaptive quadrature keeps adjusting the mesh size until a given tolerance. In this case we will use the Simpson's rule to evaluate this integral.

The code written uses adaptive quadrature, specifically adaptive Simpson's rule, to approximate the integral of a given function across a defined range. It breaks down the interval into smaller segments and adjusts the approximation until it meets the desired accuracy set by the error tolerance. While the code works, there's still room for better understanding its inner workings. I didnt spend enough time to think through the results as much as I would have liked



(a)



(b)

14 Problem 3.9

Richardson extrapolation involves using the results obtained from two computations with different step sizes to estimate the value of the integral with higher accuracy. Given the results for two different step sizes, we have to use the Simpson's rule to find a more accurate value for the integral, I. In the equation below, the subscript t will refer to a more accurate value while a is the approximation. We know the Simpson's rule has an order of accuracy of 4 so we can say:

$$I_t(h) = I_a(h) + ch^4 + O(h^6)$$
$$I_t(\frac{h}{2}) = I_a(\frac{h}{2}) + c(\frac{h}{2})^4 + O(h^6)$$

We want to get rid of the c term on h^4 to increase the accuracy so we can multiply the whole equation by 16 and subtract it from the first equation

$$I_t(h) = I_a(h) + ch^4 + O(h^6)$$
$$16I_t(\frac{h}{2}) = 16I_a(\frac{h}{2}) + ch^4 + O((\frac{h}{2})^6)$$
$$16I_t(\frac{h}{2}) - I_t(h) = 16I_a(\frac{h}{2}) - I_a(h) + O(h^6)$$

We then get a final solution of the form

$$I_t(\frac{h}{2}) = \frac{16I_t(\frac{h}{2}) - I_t(h)}{16 - 1}$$
$$I_t(\frac{h}{2}) = \frac{16(11.801) - 12.045}{16 - 1} = 11.7847$$

Additionally, I was able to come across a formula for the Richardson extrapolation where p is the order of convergence, and in our case 4 for Simpson's rule.

$$I_{extrapolation} = I_2 + \frac{I_2 - I_1}{(\frac{h_1}{h_2})^p - 1}$$

15 Appendix

15.1 Question 2

```
clear;close all;clc
% Given function
f = @(x) sin(2*x) .* cos(20*x) + exp(sin(2*x));

% Analytic first derivative
df_exact = @(x) 2*cos(2*x).*(exp(sin(2*x))+cos(20*x)) - 20*sin(2*x).*sin(20*x);

% Forward difference
forward = @(x,h) (f(x+h) - f(x))/h;

% Second Order Central difference
central_second = @(x,h) ( f(x+h) - f(x-h))/(2*h);

% Fourth order Central difference
central_fourth = @(x,h) ( f(x-2*h) - 8*f(x-h) + 8*f(x+h) - f(x+2*h) )/(12*h);

% Upper and lower limits of integration
lb = 0;
ub = 2*pi;

% Number of intervals that discretize x
N = [256 512 1024 2048];
N = 2.^(8:13);
% Initialize arrays to store errors
error_forward = zeros(size(N));
error_central_second = zeros(size(N));
error_central_fourth = zeros(size(N));
error_pade1 = zeros(size(N));
error_pade2 = zeros(size(N));
H = zeros(size(N));

for ii = 1:length(N)
    n = N(ii);
    h = (ub - lb) / n;
    H(ii) = h;
    grids = n+1;
    x = linspace(lb, ub, grids);

    % Compute numerical derivatives
    df_forward = forward(x, h);
    df_central_second = central_second(x, h);
    df_central_fourth = central_fourth(x, h);

    %Pade1
    A1 = diag(4 * ones(grids, 1)) + diag(ones(grids-1, 1), 1) + diag(ones(grids
        ↪ -1, 1), -1);
    A1(1,1:2) = [1 2];
    A1(grids,grids-1:grids) = [2 1];

    b1 = zeros(grids,1);
    b1(1) = -5*f(x(1))/2 + 2*f(x(2)) + f(x(3))/2;
    b1(2:end-1) = 3*(f(x(3:end))-f(x(1:end-2)));
```

```
    b1(end) = 5*f(x(grids))/2 - 2*f(x(grids-1)) - f(x(grids-2))/2;
    b1 = b1/h;

    % Compute approximate derivative
    df_pade1 = A1\b1;

     %Pade2
    A2 = diag(4 * ones(grids, 1)) + diag(ones(grids-1, 1), 1) + diag(ones(grids
       ↪ -1, 1), -1);
    A2(1,end-1) = 1;
    A2(grids,2) = 1;

    b2 = zeros(grids,1);
    b2(1) = f(x(2)) - f(x(grids-1));
    b2(2:end-1) = (f(x(3:end))-f(x(1:end-2)));
    b2(end) = f(x(2)) - f(x(end-1));
    b2 = 3*b2/h;

    % Compute approximate derivative
    df_pade2 = A2\b2;

    % Compute errors
    error_forward(ii) = norm(df_exact(x) - df_forward, Inf);
    error_central_second(ii) = norm(df_exact(x) - df_central_second, Inf);
    error_central_fourth(ii) = norm(df_exact(x) - df_central_fourth, Inf);
    error_pade1(ii) = norm(df_exact(x) - df_pade1', Inf);
    error_pade2(ii) = norm(df_exact(x) - df_pade2', Inf);
end

% Plot the results
loglog(H, error_forward, 'bo-','LineWidth',2, 'DisplayName', 'FD');
hold on;
loglog(H, error_central_second, 'rx-','LineWidth',2, 'DisplayName', 'CD2');
loglog(H, error_central_fourth, 'gs-','LineWidth',2, 'DisplayName', 'CD4');
loglog(H, error_pade1, 'md-','LineWidth',2, 'DisplayName', 'Pade4-1');
loglog(H, error_pade2, 'c*-','LineWidth',2, 'DisplayName', 'Pade4-2');
% Plot asymptotes for 1st, 2nd, 3rd, and 4th order lines
order1 = (H/H(1)).^1;
order2 = (H/H(1)).^2;
order3 = (H/H(1)).^3;
order4 = (H/H(1)).^4;

loglog(H, order1, 'k+--', 'LineWidth', 0.5, 'DisplayName', '1st order');
loglog(H, order2, 'kp--', 'LineWidth', 0.5, 'DisplayName', '2nd order');
loglog(H, order3, 'kh--', 'LineWidth', 0.5, 'DisplayName', '3rd order');
loglog(H, order4, 'k|--', 'LineWidth', 0.5, 'DisplayName', '4th order');
xlabel('\Delta');
ylabel('L_{\infty} Error');
title('Comparison of Finite Difference Schemes');
legend('location', 'best');
grid on
axis tight
set(gcf,'Color','w')
```

15.2 Question 3 Problem 2.1

```
clear;clc
% Define the function
f = @(x) sin(5*x);

% Define the exact second derivative
exact_second_derivative = -25*sin(5*1.5);

% Define the range of h values
h_values = logspace(-4, 0, 100);

% Initialize arrays to store errors
error_fd2 = zeros(size(h_values));
error_popular = zeros(size(h_values));

% Calculate errors for different values of h
for i = 1:length(h_values)
    h = h_values(i);

    % Finite
    % difference formula derived from two applications of the first-derivative
       ↪ operator
    second_derivative_fd2 = (f(1.5 + 2*h) - 2*f(1.5) + f(1.5 - 2*h))/(4*h^2) -
       ↪ h^2/6 * (f(1.5 + h) - 2*f(1.5) + 2*f(1.5 - h));

    % Popular second-derivative formula
    second_derivative_popular = (f(1.5 + h) - 2*f(1.5) + f(1.5 - h))/(h^2);

    % Calculate errors
    error_fd2(i) = abs(second_derivative_fd2 - exact_second_derivative);
    error_popular(i) = abs(second_derivative_popular - exact_second_derivative)
       ↪ ;
end

% Plot errors on a log-log scale
loglog(h_values, error_fd2, 'b-', 'DisplayName', 'Finite Difference Formula');
hold on
loglog(h_values, error_popular, 'r--', 'DisplayName', 'Popular Formula');
xlabel('Step Size (h)');
ylabel('Absolute Error');
title('Comparison of Second Derivative Formulas');
legend('location', 'best');
grid on;
set(gcf,'Color','w')
```

15.3 Question 4: Problem 2.2

```
clear;clc
syms a0 a1 a2 a3 h
eqn1 = a0 + a1 + a2 + a3 == 0;
eqn2 = -a0*h + 2*a3*h + a2*h == 1;
eqn3 = a0 + a2+4*a3 == 0;
eqn4 = -a0 +a2 + 8*a3 == 0;
```

```

% Solve te system of equations for a1, a2, a3, a4, and b2 in terms of
sol_ = solve([eqn1, eqn2, eqn3, eqn4], [a0, a1, a2, a3], 'ReturnConditions',
    ↪ true);

% Extracting solutions in terms of
a0_ = sol_.a0;
a1_ = sol_.a1;
a2_ = sol_.a2;
a3_ = sol_.a3;



% Display te solutions
disp(a0_)
disp(a1_)
disp(a2_)
disp(a3_)
```

15.4 Question 5 Problem 2.3

```
% Define the range of k*delta values
k_delta = linspace(0, pi, 100);

% Calculate modified wavenumber for second-order central difference scheme
k_prime_central = sin(k_delta);

% Calculate modified wavenumber for fourth-order Pad  scheme
k_prime_pade = 3 * sin(k_delta) ./ (2 + cos(k_delta));

% Plot the results
plot(k_delta, k_prime_central, 'b-', 'LineWidth', 2, 'DisplayName', 'Second
    ↪ Order Central Difference');
hold on;
plot(k_delta, k_prime_pade, 'r--', 'LineWidth', 2, 'DisplayName', 'Fourth Order
    ↪  Pad  Scheme');
plot(k_delta, k_delta, 'k-.', 'LineWidth', 1, 'DisplayName', 'Exact'); %
    ↪ Straight line y=x
xlabel('$k\Delta$', 'Interpreter', 'latex');
ylabel('$k^\prime\Delta$', 'Interpreter', 'latex');
title('Modified Wavenumber Comparison');
legend('location', 'best');
grid on;
axis tight
set(gcf,'Color','w')
```

15.5 Question 7 Problem 2.8 a and b

```
clear;clc
syms a1 a2 a3 a4 b2 h
eqn1 = a1 + a2 + a3 == 0;
eqn2 = -a2*h - 2*a3*h + a4 == 0;
```

```
eqn3 = a2*h^2 + 4*a3*h^2 + a4*h == 2*b2 + 2;
eqn4 = -(1/6)*a2*h^3 - (8/6)*a3*h^3 +(1/8)*a4*h^2 == -b2*h;
eqn5 = (1/24)*a2*h^4 +(16/24)*a3*h^4 + (1/48)*a4*h^3 == (1/2)*b2*h^2;

% Solve the system of equations for a1, a2, a3, a4, and b2 in terms of h
sol_h = solve([eqn1, eqn2, eqn3, eqn4, eqn5], [a1, a2, a3, a4, b2], '
    ↪ ReturnConditions', true);

% Extracting solutions in terms of h
a1_h = sol_h.a1;
a2_h = sol_h.a2;
a3_h = sol_h.a3;
a4_h = sol_h.a4;
b2_h = sol_h.b2;

% Display the solutions
disp(a1_h)
disp(a2_h)
disp(a3_h)
disp(a4_h)
disp(b2_h)
```

15.6 Question 7 Problem 2.8d

```
% This script solves the ode y''+y = x^2
clear;close all;clc

%Define the parameters
N = 24; % Number of grid points
h = 1 / N; % Step size
lb = 0.5*h; %lower bound
ub = 1-0.5*h; %upper bound
x = linspace(lb, ub, N)'; % Grid points

% Define the tridiagonal matrices A and B for the finite difference
    ↪ approximation of the second derivative
A = diag((10/12) * ones(N, 1)) + diag((1/12)*ones(N-1, 1), 1) + diag((1/12)*
    ↪ ones(N-1, 1), -1);
A(1,1) = 1;
A(1,2) = -11/23;
A(N,N) = 1;
A(N,N-1) = -11/23;

B = diag(-2 * ones(N, 1)) + diag(ones(N-1, 1), 1) + diag(ones(N-1, 1), -1);
B(1,1) = -36/23;
B(1,2) = 48/23;
B(1,3) = -12/23;
B(N,N) = -36/23;
B(N,N-1) = 48/23;
B(N,N-2) = -12/23;
B = B / (h^2);

% Define matrix C = A + B
C = A + B;
```

```
29
30 % Define the right-hand side function
31 b = A * (x.^3);
32
33 % Solve the linear system to find y
34 y = C \ b;
35
36 % Exact solution
37 exact_solution = x.^3 - 6*x + 6*sin(x) + 3*(2*cos(1) -1)*csc(1)*cos(x);
38
39 % Plot the exact and numerical solutions
40 plot(x, y, 'b-o', x, exact_solution, 'r--');
41 xlabel('x');
42 ylabel('y(x)');
43 title('Exact and Numerical Solutions');
44 legend('Numerical Solution', 'Exact Solution');
45 set(gcf,'Color','w')
46 axis tight
```

15.7 Question 8 Problem 2.10a

```
1  % Calculation of Derivatives on a Non-uniform Mesh
2  clear;close all;clc
3
4  % Given function
5  f = @(x) 1 - x.^8;
6  %Exact derivative
7  Df = @(x) -8 * x.^7;
8  Dg = @(x,a) a./((1-(a*x).^2)*atanh(a));
9
10 % Setup
11 a = 0.9;
12 N = 32;
13 J = 0:N;
14 xi = -1 + 2*J./N;
15 x = 1/a * tanh(xi*atanh(a));
16
17 % Central difference formula
18 dx = x(3:end) - x(1:end-2);
19 df = f(x(3:end)) - f(x(1:end-2));
20 df_dx = df ./ dx;
21
22 % Plotting
23 figure;
24 plot(x(2:end-1), df_dx,'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 6);
25 xlabel('$x$', 'Interpreter', 'latex');
26 ylabel('$f''(x)$', 'Interpreter', 'latex');
27 title('Derivative of $f(x)$ using Central Difference Formula', 'Interpreter', '
       ↪ latex');
28
29 % Exact derivative computation
30 exact_derivative = Df(x);
31
32 % Plotting exact derivative
```

```
hold on;
plot(x, exact_derivative,'--', 'LineWidth', 2);
legend('Central Difference', 'Exact Derivative');
set(gcf,'Color','w')
hold off

% Calculate derivative
delta_xi = xi(6) - xi(5); % Any consecutive indices work
df_dxi = (f(x(3:end)) - f(x(1:end-2))) / (2 * delta_xi);
df_dx2 = df_dxi .* Dg(x(2:end-1), a);
error1 = max(abs(exact_derivative(2:end-1) - df_dx));
error2 = max(abs(exact_derivative(2:end-1) - df_dx2));

disp(error1)
disp(error2)

% Plotting
figure;
plot(x(2:end-1), df_dx2, 'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 6);
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$f''(x)$', 'Interpreter', 'latex');
title('Derivative of $f(x)$ using Coordinate Transformation Method', '
   ↪ Interpreter', 'latex');
% Plotting exact derivative
hold on;
plot(x, exact_derivative,'--', 'LineWidth', 2);
legend('Central Difference', 'Exact Derivative');
set(gcf,'Color','w')
hold off
```

15.8 Question 8 Problem 2.10b

```
% Calculation of Derivatives on a Non-uniform Mesh
clear;close all;clc

% Given function
f = @(x) 1 - x.^8;
%Exact derivative
Df = @(x) -8 * x.^7;
Dg = @(x) -1./sqrt(1-x.^2);

% Setup
N = 75;
J = 0:N;
xi = (pi*J)/N;
x = cos(xi);

% Central difference formula
dx = x(3:end) - x(1:end-2);
df = f(x(3:end)) - f(x(1:end-2));
df_dx = df ./ dx;

% Plotting
figure;
```

```
plot(x(2:end-1), df_dx,'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 6);
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$f''(x)$', 'Interpreter', 'latex');
title('Derivative of $f(x)$ using Central Difference Formula', 'Interpreter', '
    ↪ latex');

% Exact derivative computation
exact_derivative = Df(x);


% Plotting exact derivative
hold on;
plot(x, exact_derivative,'--', 'LineWidth', 2);
legend('Central Difference', 'Exact Derivative');
set(gcf,'Color','w')
hold off
% Save image with specific name
saveas(gcf, 'q2_10bcentral_difference.jpg'); % Save the image as "
    ↪ central_difference.png"

% Calculate derivative
delta_xi = xi(6) - xi(5); % Any consecutive indices work
df_dxi = (f(x(3:end)) - f(x(1:end-2))) / (2 * delta_xi);
df_dx2 = df_dxi .* Dg(x(2:end-1));

error1 = max(abs(exact_derivative(2:end-1) - df_dx));
error2 = max(abs(exact_derivative(2:end-1) - df_dx2));

disp(error1)
disp(error2)

% Plotting
figure;
plot(x(2:end-1), df_dx2, 'LineWidth', 2, 'Marker', 'o', 'MarkerSize', 6);
xlabel('$x$', 'Interpreter', 'latex');
ylabel('$f''(x)$', 'Interpreter', 'latex');
title('Derivative of $f(x)$ using Coordinate Transformation Method', '
    ↪ Interpreter', 'latex');
% Plotting exact derivative
hold on;
plot(x, exact_derivative,'--', 'LineWidth', 2);
legend('Central Difference', 'Exact Derivative');
set(gcf,'Color','w')
hold off
% Save image with specific name
saveas(gcf, 'q2_10bCoordinate Transformation.jpg');
```

15.9 Question 12 Problem 3.6

```
% This script solves for the Fredholm Equation
clear;close all;clc
N = 60;% Number of grid points
lb = 0; %lower bound
ub = pi; %upper bound
```

```
h = (ub-lb) / (N-1); % Step size

x = linspace(lb,ub,N)';

%Create f column vector
f = -pi*x.^2;

%Preallocate A matrix
A = zeros(N,N);

for ii = 1: length(x) %fill the rows
    A(ii,1) = 0.5*h*3*(0.5*sin(3*x(ii)) - x(1)*x(ii)^2);
    A(ii,N) = 0.5*h*3*(0.5*sin(3*x(ii)) - x(N)*x(ii)^2);

    for jj = 2:length(x)-1 %fill the columns
        A(ii,jj) = 3*h*(0.5*sin(3*x(ii)) - x(jj)*x(ii)^2);
    end

    A(ii,ii) = -1+ A(ii,ii);

end

%Solve for phi
phi = A\f;

% Exact solution
exact_solution =sin(3*x);

% Plot the exact and numerical solutions
plot(x, phi, 'b-o', x, exact_solution, 'r--');
xlabel('x');
ylabel('y(x)');
title('Exact and Numerical Solutions');
legend('Numerical Solution', 'Exact Solution');
set(gcf,'Color','w')
axis tight
```

Question 13 Problem 3.8

```
clear;close all;clc

% Given function
f = @(x) 100./sqrt(x+ 0.01) + 1./((x - 0.3).^2 + 0.001) - pi;
fdiff = @(x) -50./(x+ 0.01)^1.5 - (2*(x-0.3))./((x - 0.3).^2 + 0.001)^2;

% Upper and lower limits of integration
a = 0;
b = 1;

% Exact value of the integral (if available)
exact_integral = integral(f, a, b);

% Values of n (number of panels)
n_values = 2.^(3:15);
```

```
16
17 % Preallocate arrays to store errors and panel lengths
18 errors = zeros(size(n_values));
19 h_values = zeros(size(n_values));
20
21 % Calculate errors for different values of n
22 for i = 1:length(n_values)
23     n = n_values(i);
24     h = (b - a) / n; % Calculate step size
25     h_values(i) = h;
26
27     % Calculate the approximate integral using the trapezoidal rule
28     x = linspace(a, b, n + 1); % Generate x-values for the panels
29     y = f(x); % Evaluate the function at x-values
30
31     y_odd = y(2:2:end-1);
32     y_even = y(3:2:end-2);
33
34     approximate_integral_trapezoid = 0.5*h * ((y(1) + y(end)) + 2*sum(y(2:n)));
       ↪  % Trapezoidal rule formula
35     approximate_integral_simpsons = (h/3) * (y(1) + 4*sum(y_odd) + 2*sum(y_even
       ↪ ) + y(end)); % Simpson's rule formula
36     approximate_integral_trapezoid_EC= approximate_integral_trapezoid -h^2/12
       ↪ *(fdiff(x(end)) - fdiff(x(1)));
37
38     % Calculate the error percentage
39     errors(i,1) = abs(exact_integral - approximate_integral_trapezoid)/
       ↪ exact_integral *100;
40     errors(i,2) = abs(exact_integral - approximate_integral_simpsons)/
       ↪ exact_integral *100;
41     errors(i,3) = abs(exact_integral - approximate_integral_trapezoid_EC)/
       ↪ exact_integral *100;
42 end
43
44 % Plot error percentage vs. number of panels on a log-log scale
45 loglog(n_values, errors(:,1), 'bo-','DisplayName','Trapezoid Rule');
46 hold on
47 loglog(n_values, errors(:,2), 'ro-','DisplayName','Simpson Rule');
48 loglog(n_values, errors(:,3), 'm*-','DisplayName','Trapezoid Rule with EC');
49 xlabel('Number of panels ($n$)', 'Interpreter', 'latex');
50 ylabel('Error (\%)', 'Interpreter', 'latex');
51 title('Error vs. Number of Panels', 'Interpreter', 'latex');
52 axis tight
53 legend
54 grid on
55
56 Simp= errors(:,2);
57 TrapEC= errors(:,3);
58 % Find the slope and intercept of the line using linear regression
59 p2 = polyfit(log(n_values(2:end)), log(Simp(2:end)), 1);
60 slope2 = p2(1);
61 intercept2 = p2(2);
62
63 % Find the slope and intercept of the line using linear regression
64 p3 = polyfit(log(n_values(2:end)), log(TrapEC(2:end)), 1);
65 slope3 = p3(1);
```

```
intercept3 = p3(2);

% Create the equation of the line
line_equation3 = sprintf('$n^{%.2f}$', slope3);
line_equation2 = sprintf(' $n^{%.2f}$', slope2);
% Add the equation of the line to the plot
text(1500, 3.4e-6, line_equation2, 'Interpreter', 'latex', 'FontSize', 12);
text(40, 0.00299, line_equation3, 'Interpreter', 'latex', 'FontSize', 12);

% Specify a range of error tolerances
tolerances = [1e-2, 1e-10];

% Evaluate the integral using adaptive quadrature with different tolerances
for i = 1:length(tolerances)
    % Initialize a cell array to store x points for the current tolerance
    x_points = [];

    tolerance = tolerances(i);
    fa = f(a); % Evaluate function at endpoints
    fb = f(b);

    % Evaluate the integral using adaptive quadrature
    integral_value = adaptive_quadrature(f, a, b, fa, fb, tolerance);

    % Store evaluation points (endpoints and midpoints)
    x_points = unique([x_points, a, b, (a+b)/2]);

    % Plot the integrand and the positions of its evaluations for the current
       ↪ tolerance
    figure;
    x_values = linspace(a, b, 1000); % Generate points for smooth plotting
    plot(x_values, f(x_values), 'b-', 'LineWidth', 1.5); % Plot the integrand
    hold on;
    scatter(x_points, f(x_points), 'r.'); % Scatter plot of evaluation points
    xlabel('x');
    ylabel('f(x)');
    title(['Integrand and Positions of Evaluations (Tolerance = ', num2str(
       ↪ tolerance), ')']);
    legend('Integrand', 'Evaluation Points');
    grid on;
end

% Define the recursive adaptive quadrature function
function integral_value = adaptive_quadrature(func, a, b, fa, fb,tolerance)
c = (a + b) / 2; % Midpoint
h = b - a; % Width of the interval
fc = func(c); % Function evaluation at the midpoint
S = (h / 6) * (fa + 4*fc + fb); % Coarse Simpson's rule approximation
d = (a + c) / 2;
e = (c + b) / 2;
fd = func(d);
fe = func(e);
S_left = (h / 12) * (fa + 4*fd + fc); % Left half interval approximation
S_right = (h / 12) * (fc + 4*fe + fb); % Right half interval approximation
S2 = S_left + S_right; % Fine Simpson's rule approximation
if abs(S2 - S) <= 15 * tolerance
```

```
    integral_value = S2 + (S2 - S) / 15; % Error corrected result
else
    integral_value = adaptive_quadrature(func, a, c, fa, fc, tolerance) + ...
        adaptive_quadrature(func, c, b, fc, fb,tolerance); % Recursive call on
            ↪ subintervals
end
end
```