

MATH 231 : Numerical ODEs

Pratham Lalwani

April 15, 2025

Question 1: Polynomial Interpolation

- (a) Consider the data $\{(-1, -5), (0, 1), (1, 3)\}$. This exercise is to compute by hand the Lagrange interpolant $f_n(x)$.

(i) Compute $f_2(x)$ by hand using Newton basis functions and verify it interpolates the data.

(ii) Suppose a new data point $(2, 7)$ is added, repeat (i) and verify $f_3(x)$ interpolates the data.

- (b) Instead, suppose you have derivative information on the data $\{(-1, -5, 4), (0, 1, 1), (2, 7, 121)\}$, compute by hand the Hermite interpolant $H_5(x)$ and verify that it interpolates the data.

- (c) Recall for $n+1$ data points $\{(x_i, y_i)\}_{i=0}^n$, the Lagrange interpolant written in terms of Lagrange basis is given by

$$f_n(x) = \sum_{j=0}^n y_j \ell_j(x), \quad \text{where } \ell_j(x) = \frac{\prod_{i \neq j}^n (x - x_i)}{\prod_{i \neq j}^n (x_j - x_i)}. \quad (1)$$

This exercise is about evaluating $f_n(x)$ more efficiently on many different values of x , such as when plotting $f_n(x)$.

(i) Show that evaluating $f_n(x)$ at a fixed value of x using (1) directly involves $\mathcal{O}(n^2)$ costs.

(ii) Instead, deduce that 1 can be rewritten as

$$f_n(x) = \ell(x) \sum_{j=0}^n y_j \frac{w_j}{x - x_j}, \quad \text{where } \ell(x) = \prod_{i=0}^n (x - x_i) \text{ and } w_j = (\ell'(x_j))^{-1} = \left(\prod_{i \neq j}^n (x_j - x_i) \right)^{-1}. \quad (2)$$

- (iii) Show that computing $\{w_j\}_{j=0}^n$ takes $\mathcal{O}(n^2)$ costs but it needs to be done only once for different values x .

Hint: Unlike the expressions for $\ell_j(x)$, notice that w_j only depends the nodes $\{x_j\}_{j=0}^n$.

- (iv) To further remove the need to compute $\ell(x)$, use (2) to deduce that (1) can be rewritten as

$$f_n(x) = \left(\sum_{j=0}^n y_j \frac{w_j}{x - x_j} \right) / \left(\sum_{j=0}^n \frac{w_j}{x - x_j} \right) \quad (3)$$

This is also called the barycentric interpolation formula for $f_n(x)$.

Hint: Recall $f_n(x) = f(x)$ for any polynomial function $f(x)$ up to degree n , in particular for $f(x) = 1$.

- (v) Conclude that once $\{w_j\}_{j=0}^n$ is computed and stored, evaluating $f_n(x)$ using (3) takes $\mathcal{O}(n)$ costs for each x .

- (vi) Implement a program to interpolate $f(x) = \sin^3(\pi x)$ on $[-1, 1]$ using $n = 5, 10, 15, 20$ uniformly-spaced nodes and plot resulting interpolants $f_n(x)$ versus $f(x)$ on 100 uniformly-spaced nodes using (3).

Hint: When evaluating $f_n(x_i)$ with (3), replace any NaN's in your output with $f(x_i)$.

- (a) (i) First we compute the secant slopes as they will give us the coefficients for the Newton Polynomial

1	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$
-1	-5		
		6	
0	1		-2
		2	
1	3		

. Therefore, the corresponding Newton Polynomial is :

$$\begin{aligned}
 f_2(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) \\
 &= -5 + 6(x + 1) - 2(x + 1)(x) \\
 &= -5 + 6x + 6 - 2x^2 - 2x \\
 &= 1 + 4x - 2x^2.
 \end{aligned}$$

- (ii) We update our previous secant table by adding the point:

1	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$	$f[x_i, x_{i+1}, x_{i+2}, x_{i+3}]$
-1	-5			
		6		
0	1		-2	
		2		1
1	3		1	
		4		
2	7			

. Therefore the corresponding Newton polynomial is,

$$\begin{aligned}
 f_3(x) &= f_2(x) + f[x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) \\
 &= 1 + 4x - 2x^2 + (x + 1)x(x - 1) \\
 &= 1 + 4x - 2x^2 + x^3 - x \\
 &= 1 + 3x - 2x^2 + x^3
 \end{aligned}$$

- (b)

i	u_i	$f[u_i]$	$f[u_i, u_{i+1}]$				
0	-1	-5					
			4				
1	-1	-5		$\frac{6-4}{0-(-1)} = 2$			
			$\frac{-5-1}{-1-0} = 6$		$\frac{-5-2}{0-(-1)} = -7$		
2	0	1		$\frac{1-6}{0-(-1)} = -5$		$\frac{2-(-7)}{2-(-1)} = 3$	
			1		$\frac{1-(-5)}{2-(-2)} = 2$		$\frac{9-3}{2+1} = 2$
3	0	1		$\frac{3-1}{2-0} = 1$		$\frac{29-2}{2-(-1)} = 9$	
			$\frac{7-1}{2-0} = 3$		$\frac{59-1}{2-0} = 29$		
4	2	7		$\frac{121-3}{2-0} = 59$			
			121				
5	2	7					

- (c)

(i)

Algorithm 1: Naive Polynomial Interpolation

```

1 function NaivePolynomialEval(a,x,x,y):
    Input:
    a: The array of coefficients
    x: Point of evaluation
    Output:
    sum: Polynomial evaluation

2 s = 0
3 n = len(a)
4 for i=0 ... n:
5     li=1
6     for j= 1... n:
7         if i ≠ j:
8             li = li ·  $\frac{x-x_j}{x_i-x_j}$ 
9     sum = yi · li
10 return sum

```

From the above we can see that it takes $\mathcal{O}(n^2)$ to evaluate a polynomial.

(ii)

$$\begin{aligned}
 f_n(x) &= \sum_{j=0}^n y_j \ell_j(x) \\
 &= \sum_{j=0}^n y_j \frac{\prod_{i \neq j}^n (x - x_i)}{\prod_{i \neq j}^n (x_j - x_i)} \\
 &= \sum_{j=0}^n y_j \frac{\prod_{i=0}^n (x - x_i)}{(x - x_j) \prod_{i \neq j}^n (x_j - x_i)} \\
 &= \ell(x) \sum_{j=0}^n \frac{y_j}{(x - x_j) \prod_{i \neq j}^n (x_j - x_i)} \\
 &= \ell(x) \sum_{j=0}^n y_j \frac{w_j}{(x - x_j)}
 \end{aligned}$$

(iii)

Algorithm 2: Richardson Iteration

```
1 function NaivePolynomialEval(a,x,x,y):  
    Input:  
    a: The array of coefficients  
    x: Point of evaluation  
    Output:  
    sum: Polynomial evaluation  
  
2     s = 0  
3     n = len(a)  
4     for j=0 ... n:  
5         li=1  
6         for j= 1... n:  
7             if i ≠ j:  
8                 li = li ·  $\frac{x-x_j}{x_i-x_j}$   
9             sum = yi · li  
10    return sum
```

From the above we can see that it takes $O(n^2)$ to evaluate a polynomial.

(iv)

Question 2: Spline Interpolation

- (a) Compute the constant, linear, natural cubic spline by hand for the data $\{(1, 2), (2, 3), (3, 5)\}$.
- (b) Implement an $O(n)$ program to interpolate $f(x) = \cos(x^2)$ with a clamped cubic spline $S(x)$ using $n = 5, 10, 15, 20$ uniformly-spaced nodes on $[0, 2\sqrt{\pi}]$ and plot resulting interpolants $S_n(x)$ versus $f(x)$ on 100 uniformly-spaced nodes. Hint: Use your $O(n)$ algorithm from **A0** to solve the tridiagonal system for the coefficients c_i and use Horner's method to evaluate the piecewise cubic polynomials for plotting.
- (c) Consider interpolating f using a not-a-knot cubic spline on $a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$ (not necessarily uniformly spaced). Write down the linear system for determining the coefficients c_i in the form

$$Ac = f$$

where A is a tridiagonal matrix of $(n-1) \times (n-1)$, $c \in \mathbb{R}^{n-1}$ is the vector of c_i coefficients, and $f \in \mathbb{R}^{n-1}$ is the vector involving divided difference of f .

Hint: Show that the not-a-knot boundary condition implies $d_0 = d_1$ and $d_{n-1} = d_{n-2}$. Use these relations to deduce relations for c_0 and c_n by substituting them into the boxed formula on Lecture 15-page 12.

Solution

- (a) The constant spline is,

$$S(x) = \begin{cases} 2, & x \in [1, 2) \\ 3, & x \in [2, 3) \\ 5, & x = 3 \end{cases}.$$

The linear spline is,

$$S(x) = \begin{cases} y_0 + f[x_0, x_1](x - 1), & x \in [1, 2) \\ y_1 + f[x_1, x_2](x - 1), & x \in [2, 3] \end{cases} = \begin{cases} 2 + (x - 1), & x \in [1, 2) \\ 3 + 2(x - 2), & x \in [2, 3] \end{cases}.$$

The cubic spline has the following form ,

$$S(x) = \begin{cases} a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, & x \in [x_i, x_{i+1}) \\ a_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3, & x \in [x_{n-1}, x_n] \end{cases}.$$

To solve for the natural cubic spline we have to solve the following equation first,

$$\begin{aligned} (2(h_0 + h_1)) (c_1) &= 3 (f[x_1, x_2] - f[x_0, x_1]) \\ (4) (c_1) &= 3 (f[x_1, x_2] - f[x_0, x_1]) \\ 4c_1 &= 3(2 - 1) \\ c_1 &= \frac{3}{4}. \end{aligned}$$

So we have, $c_0 = c_2 = 0$ and $c_1 = \frac{3}{4}$. The corresponding d_i are,

$$\begin{aligned} d_0 &= \frac{c_1 - c_0}{3h_0} = \frac{\frac{3}{4}}{3} = \frac{1}{4} \\ d_1 &= \frac{c_2 - c_1}{3h_1} = \frac{-\frac{3}{4}}{1} = -\frac{1}{4}. \end{aligned}$$

Next we compute the b_i ,

$$\begin{aligned} b_0 &= f[x_0, x_1] - \frac{h_0}{3} (c_1 + 2c_0) = 1 - \frac{1}{3} \left(\frac{3}{4} \right) = \frac{3}{4} \\ b_1 &= f[x_1, x_2] - \frac{h_1}{3} (c_2 + 2c_1) = 2 - \frac{1}{3} \left(\frac{6}{4} \right) = \frac{3}{2}. \end{aligned}$$

So the corresponding natural cubic spline is,

$$S(x) = \begin{cases} 2 + \frac{3}{4}(x - 1) + \frac{3}{4}(x - 1)^3, & x \in [1, 2) \\ 3 + \frac{3}{2}(x - 2) + \frac{3}{4}(x - 2)^2 - \frac{1}{4}(x - 2)^3, & x \in [2, 3] \end{cases}.$$

(b) In jupyter notebook.

(c) The **not-a-knot** cubic spline condition is,

$$p_0'''(x_0) = p_1'''(x_1) \quad \text{and} \quad p_{n-2}'''(x_{n-1}) = p_{n-1}'''(x_{n-1})$$

$$p_0'''(x_0) = d_0 = d_1 = p_1'''(x_1) \quad \text{and} \quad p_{n-1}'''(x_{n-1}) = d_{n-1} = d_{n-2} = p_{n-2}'''(x_{n-1}).$$

Using the above in the relations derived between c_i and d_i , we get

$$d_0 = \frac{c_1 - c_0}{3h_0} = \frac{c_2 - c_1}{3h_1} = d_1.$$

Simplifying gives,

$$-c_0h_1 + (h_0 + h_1)c_1 - c_2h_1 = 0.$$

On the other hand,

$$d_{n-1} = \frac{c_n - c_{n-1}}{3h_{n-1}} = \frac{c_{n-2} - c_{n-1}}{3h_{n-2}} = d_{n-2}.$$

Simplifying the above gives,

$$-c_{n-2}h_{n-1} + (h_{n-1} + h_{n-2})c_{n-1} - c_n h_{n-2} = 0.$$

Which gives the following linear system,

$$A = \begin{pmatrix} -h_1 & (h_0 + h_1) & -h_1 & & & 0 \\ h_0 & 2(h_0 + h_1) & h_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} & \\ & & -h_{n-1} & (h_{n-2} + h_{n-1}) & h_{n-2} & \end{pmatrix}.$$

$$\vec{c} = \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix}.$$

Question 3: Trigonometric interpolation and Discrete Fourier Transform

Denote the vectors $\mathbf{f} = (f_0, f_1, \dots, f_{N-1})^T$ and $\hat{\mathbf{f}} = (\hat{f}_0, \hat{f}_1, \dots, \hat{f}_{N-1})^T$. Recall that the Discrete Fourier Transform (DFT) of \mathbf{f} is $\hat{\mathbf{f}}$ and the inverse Discrete Fourier Transform of $\hat{\mathbf{f}}$ is \mathbf{f} given by in components

$$\hat{f}_k = \sum_{j=0}^{N-1} f_j \omega_N^{-jk}, \quad f_j = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}_k \omega_N^{jk}, \quad \text{where } \omega_N := e^{\frac{2\pi i}{N}}, \text{ for } 0 \leq k \leq N-1 \quad (4)$$

- Compute explicitly the DFT $\hat{\mathbf{f}}$ of $\mathbf{f} = (5, 1, 5, 1, 5, 1, 5, 1)^T$ using (4). Sketch the associated truncated Fourier series $\hat{f}_N(x)$ on $[0, 2\pi]$?
- Set all frequency components $\hat{f}_4, \dots, \hat{f}_7$ of $\hat{\mathbf{f}}$ to zero and call this vector $\tilde{\mathbf{f}}$. Now compute the inverse DFT of $\tilde{\mathbf{f}}$ and sketch the new truncated Fourier series $\tilde{f}_N(x)$? Explain why this result is expected. Remark: This is an example of a "low-pass" filter and has applications in signal processing.
- The following exercise outlines the main idea behind the Fast Fourier Transform (FFT) algorithm introduced by Cooley and Tukey in 1965. For simplicity, we assume $N = 2^m$ for some integer m .
 - For $0 \leq k \leq N-1$, show that the sum in (4) of the DFT can be written in terms of even/odd components as

$$\hat{f}_k = \sum_{j=0}^{\frac{N}{2}-1} f_{2j} \left(\omega_{\frac{N}{2}} \right)^{-jk} + \omega_N^{-k} \sum_{j=0}^{\frac{N}{2}-1} f_{2j+1} \left(\omega_{\frac{N}{2}} \right)^{-jk}$$

Hint: Express $\omega_{\frac{N}{2}}$ in terms of ω_N .

- Denote $FFT(\mathbf{f}, N)$ as the DFT of \mathbf{f} with N components and define $\boldsymbol{\omega} := (1, \omega_N^{-1}, \omega_N^{-2}, \dots, \omega_N^{-(N-1)})^T$. By writing out the N equations from part (i), deduce that $FFT(\mathbf{f}, N)$ can be written recursively as

$$FFT(\mathbf{f}, N) = \begin{pmatrix} FFT(\mathbf{f}_{\text{even}}, \frac{N}{2}) \\ FFT(\mathbf{f}_{\text{even}}, \frac{N}{2}) \end{pmatrix} + \boldsymbol{\omega} \odot \begin{pmatrix} FFT(\mathbf{f}_{\text{odd}}, \frac{N}{2}) \\ FFT(\mathbf{f}_{\text{odd}}, \frac{N}{2}) \end{pmatrix}$$

where \odot denotes element-wise multiplication and $\mathbf{f}_{\text{even}}/\mathbf{f}_{\text{odd}}$ denotes the even/odd components of \mathbf{f} . Using this, write a recursive pseudocode for the function *FFT* to compute the DFT of \mathbf{f} .

- (iii) Let $T(N)$ be the FLOPs to compute $FFT(\mathbf{f}, N)$ and assume $T(1)$ is a constant C . Express $T(N)$ as a recursive function of $T(N/2)$ and show that the overall computational cost to compute $FFT(\mathbf{f}, N)$ is $\mathcal{O}(N \log_2 N)$. How much faster is FFT compared to DFT using direct matrix-to-vector multiplication for $m = 5, 10, 15, 20$?

Remark: FFT was considered one of top 10 most impactful algorithm of 20th century - see <https://archive.siam.org/pdf/news/637.pdf>.

(a) Given that $\mathbf{f} = (5, 1, 5, 1, 5, 1, 5, 1)$ we want to compute $\hat{\mathbf{f}}$

$$\hat{f}_0 = \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{7}} \right)^0 = 24$$

$$\begin{aligned} \hat{f}_1 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-j} \\ &= 5 + e^{-\frac{2\pi i}{8}} + 5e^{-\frac{4\pi i}{8}} + e^{-\frac{6\pi i}{8}} + 5e^{-\frac{8\pi i}{8}} + e^{-\frac{10\pi i}{8}} + 5e^{-\frac{12\pi i}{8}} + e^{-\frac{14\pi i}{8}} \\ &= 5 + e^{-\frac{\pi i}{4}} + 5e^{-\frac{\pi i}{2}} + e^{-\frac{3\pi i}{4}} - 5 - e^{-\frac{\pi i}{4}} - 5e^{-\frac{\pi i}{2}} - e^{-\frac{3\pi i}{4}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \hat{f}_2 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-2j} \\ &= 5 + e^{-\frac{4\pi i}{8}} + 5e^{-\frac{8\pi i}{8}} + e^{-\frac{12\pi i}{8}} + 5e^{-\frac{16\pi i}{8}} + e^{-\frac{20\pi i}{8}} + 5e^{-\frac{24\pi i}{8}} + e^{-\frac{28\pi i}{8}} \\ &= 5 + e^{-\frac{\pi i}{2}} + 5e^{-\pi i} + e^{-\frac{3\pi i}{2}} + 5 + e^{-\frac{5\pi i}{2}} + 5e^{-3\pi i} + e^{-\frac{7\pi i}{2}} \\ &= 5 + e^{-\frac{\pi i}{2}} - 5 - e^{-\frac{\pi i}{2}} + 5 + e^{-\frac{\pi i}{2}} - 5 - e^{-\frac{\pi i}{2}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \hat{f}_3 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-3j} \\ &= 5 + e^{-\frac{3\pi i}{4}} + 5e^{-\frac{6\pi i}{4}} + e^{-\frac{9\pi i}{4}} + 5e^{-\frac{12\pi i}{4}} + e^{-\frac{15\pi i}{4}} + 5e^{-\frac{18\pi i}{4}} + e^{-\frac{21\pi i}{4}} \\ &= 5 + e^{-\frac{3\pi i}{4}} - 5e^{-\frac{\pi i}{2}} + e^{-\frac{\pi i}{4}} - 5 - e^{-\frac{3\pi i}{4}} + 5e^{-\frac{\pi i}{2}} - e^{-\frac{\pi i}{4}} \\ &= 0 \end{aligned}$$

$$\begin{aligned} \hat{f}_4 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-4j} \\ &= 5 + e^{1\pi i} + 5e^{2\pi i} + e^{3\pi i} + 5e^{4\pi i} + e^{5\pi i} + 5e^{6\pi i} + e^{7\pi i} \\ &= 5 - 1 + 5 - 1 + 5 - 1 + 5 - 1 \\ &= 16 \end{aligned}$$

$$\begin{aligned} \hat{f}_5 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-5j} \\ &= 5 + e^{-\frac{5\pi i}{4}} + 5e^{-\frac{5\pi i}{2}} + e^{-\frac{15\pi i}{4}} + 5e^{-5\pi i} + e^{-\frac{25\pi i}{4}} + 5e^{-\frac{15\pi i}{2}} + e^{-\frac{35\pi i}{4}} \\ &= 5 - e^{-\frac{\pi i}{4}} + 5e^{-\frac{\pi i}{2}} + e^{-\frac{\pi i}{4}} - 5 - e^{-\frac{3\pi i}{4}} + 5e^{-\frac{\pi i}{2}} - e^{-\frac{\pi i}{4}} \\ &= 0 \end{aligned}$$

(b) Given that $\mathbf{f} = (5, 1, 5, 0, 0, 0, 0, 0)$ we want to compute $\hat{\mathbf{f}}$

$$\begin{aligned}
\hat{f}_0 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^0 = 11 \\
\hat{f}_1 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-j} \\
&= 5 + e^{\frac{-2\pi i}{8}} + 5e^{\frac{-4\pi i}{8}} + 0 + 0 + 0 + 0 + 0 \\
&= 5 + e^{\frac{-\pi i}{4}} + 5e^{\frac{-\pi i}{2}} \\
&= 0 \\
\hat{f}_2 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-2j} \\
&= 5 + e^{\frac{-4\pi i}{8}} + 5e^{\frac{-8\pi i}{8}} + e^{\frac{-12\pi i}{8}} + 5e^{\frac{-16\pi i}{8}} + e^{\frac{-20\pi i}{8}} + 5e^{\frac{-24\pi i}{8}} + e^{\frac{-28\pi i}{8}} \\
&= 5 + e^{\frac{-\pi i}{2}} + 5e^{-\pi i} + e^{\frac{-3\pi i}{2}} + 5 + e^{\frac{-5\pi i}{2}} + 5e^{-3\pi i} + e^{\frac{-7\pi i}{2}} \\
&= 5 + e^{\frac{-\pi i}{2}} - 5 - e^{\frac{-\pi i}{2}} + 5 + e^{\frac{-\pi i}{2}} - 5 - e^{\frac{-\pi i}{2}} \\
&= 0 \\
\hat{f}_3 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-3j} \\
&= 5 + e^{\frac{-3\pi i}{4}} + 5e^{\frac{-6\pi i}{4}} + e^{\frac{-9\pi i}{4}} + 5e^{\frac{-12\pi i}{4}} + e^{\frac{-15\pi i}{4}} + 5e^{\frac{-18\pi i}{4}} + e^{\frac{-21\pi i}{4}} \\
&= 5 + e^{\frac{-3\pi i}{4}} - 5e^{\frac{-\pi i}{2}} + e^{\frac{-\pi i}{4}} - 5 - e^{\frac{-3\pi i}{4}} + 5e^{\frac{-\pi i}{2}} - e^{\frac{-\pi i}{4}} \\
&= 0 \\
\hat{f}_4 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-4j} \\
&= 5 + e^{1\pi i} + 5e^{2\pi i} + e^{3\pi i} + 5e^{4\pi i} + e^{5\pi i} + 5e^{6\pi i} + e^{7\pi i} \\
&= 5 - 1 + 5 - 1 + 5 - 1 + 5 - 1 \\
&= 16 \\
\hat{f}_5 &= \sum_{j=0}^7 f_j \left(e^{\frac{2\pi i}{8}} \right)^{-5j} \\
&= 5 + e^{\frac{-5\pi i}{4}} + 5e^{\frac{-5\pi i}{2}} + e^{\frac{-15\pi i}{4}} + 5e^{-5\pi i} + e^{\frac{-25\pi i}{4}} + 5e^{\frac{-15\pi i}{2}} + e^{\frac{-35\pi i}{4}} \\
&= 5 - e^{\frac{-\pi i}{4}} + 5e^{\frac{-\pi i}{2}} + e^{\frac{-\pi i}{4}} - 5 - e^{\frac{-3\pi i}{4}} + 5e^{\frac{-\pi i}{2}} - e^{\frac{-\pi i}{4}} \\
&= 0
\end{aligned}$$

Question 4: Numerical Differentiation and Integration

- (a) In solving boundary value problems with Neumann boundary conditions, derivative values at the boundary need to be approximated from one side of the boundary. For some constants a, b, c , design an one-sided finite difference approximation for $f'(x)$ of the form,

$$D_h f(x) = af(x) + bf(x+h) + cf(x+2h),$$

with the highest degree of accuracy. Derive its error term and find the highest p so that the error is

bounded by $\mathcal{O}(h^p)$.

- (b) Consider integrating $f(x) = e^{-x^2}$ which often arises in probability and statistics for computing the cumulative distribution function of the Gaussian distribution.
- (i) Write down the quadrature formulas involving h and n for $I(f) = \int_a^b f(x)dx$ using the composite midpoint, trapezoidal and Simpson's method. State their degree of accuracy.
 - (ii) Recall the error for each of the composite rules is bounded by $\frac{M_2}{24}(b-a)h^2$, $\frac{M_2}{12}(b-a)h^2$, and $\frac{M_4}{180}(b-a)h^4$ respectively, where $M_n := \max_{x \in [a,b]} |f^{(n)}(x)|$. Estimate the number of subintervals n needed for each method in order to guarantee an error tolerance of 10^{-6} for $I(f)$ on $[0, 1]$. How do these estimates compare to part (iv)?
 - (iii) The 4-point Gauss Quadrature has Gauss points on $\left\{ \pm\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, \pm\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}} \right\}$ with respective weights $\left\{ \frac{18+\sqrt{30}}{36}, \frac{18+\sqrt{30}}{36}, \frac{18-\sqrt{30}}{36}, \frac{18-\sqrt{30}}{36} \right\}$. Write down the 4-point Gauss Quadrature of $I(f)$ on $[0, 1]$.
 - (iv) Using parts (i)-(iii), implement a program to generate a table of values for the four quadrature methods to reach the tolerance criterion of 10^{-5} . Using the exact value of $I(f)$ on $[0, 1]$ is $0.7468241328\dots$, verify the three composite methods have the expected error of order $\mathcal{O}(h^p)$, i.e. graph a log-log plot of error versus h to verify the order p . Rank their computational costs based on their number of function evaluations.
- (c) Recall the composite Trapezoidal rule has the error $I(f) = I_h(f) + \mathcal{O}(h^2)$.
- (i) Apply Richardson's extrapolation to the composite Trapezoidal rule to derive a more accurate quadrature formula. What is the expected improved order?
 - (ii) Demonstrate the improved convergence with your new quadrature rule for the integral from part (b) by graphing a log-log plot of error versus h to verify the order p . Explain briefly if you do not get the expected order.

Solution

- (a) Because we have three unknowns a, b, c we can have 3 equations and hence highest degree of accuracy we can achieve is 2. Therefore consider,

If $f(x) = 1$,

$$D_h f(x) = f'(x) = 0 = a + b + c.$$

If $f(x) = x$

$$D_h f(x) = f'(x) = 1 = ax + b(x+h) + c(x+2h).$$

$$\implies 1 = (a+b+c)x + (b+2c)h = (b+2c)h \implies \frac{1}{h} = b+2c.$$

If $f(x) = x^2$

$$D_h f(x) = f'(x) = 2x = ax^2 + b(x+h)^2 + c(x+2h)^2.$$

$$\implies 2x = (a+b+c)x^2 + 2(b+2c)xh + (b+4c)h^2 = 2x + (b+4c)h^2.$$

$$\implies 0 = b + 4c = \frac{1}{h} + 2c \implies \frac{-1}{2h} = c.$$

Also,

$$0 = b + 4c = \frac{2}{h} - b \implies b = \frac{2}{h}.$$

Combining gives,

$$a = \frac{-3}{2h}.$$

Therefore,

$$D_h f(x) = \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}.$$

Which has degree order of accuracy of 2.

Using the Taylor Remainder Theorem around x gives,

$$\begin{aligned} D_h f(x) &= \frac{1}{2h} \left(-3f(x) + 4f(x) + 4f'(x)h + 2f''(x)h^2 + \frac{2f'''(\xi_1)h^3}{3} \right. \\ &\quad \left. - f(x) - 2f'(x)h - 2f''(x)h^2 - \frac{4f'''(\xi_2)h^3}{3} + O(h^4) \right) \\ &= \frac{1}{2h} \left(2f'(x)h - \frac{2}{3} (f'''(\xi_1) - 2f'''(\xi_2)) h^3 + O(h^4) \right) \\ |D_h f(x) - f'(x)| &= \left| \frac{1}{3} (f'''(\xi_1) - f'''(\xi_2)) h^2 + O(h^3) \right| \leq \frac{2}{3} \max_{\xi \in [x, x+2h]} f'''(\xi) h^2 + O(h^3). \end{aligned}$$

(b) (i) Doing the integral using the mid-point rule we get,

$$I(f) \approx I_{\text{mid-point}}(f) = \sum_{i=1}^n f\left(a + ih - \frac{h}{2}\right) h.$$

$$I(f) \approx \sum_{i=1}^n \exp\left(\left(a + ih - \frac{h}{2}\right)^2\right) h$$

Doing the integral using the trapezoidal rule we get,

$$I(f) \approx I_{\text{trapezoidal}}(f) = h \left(\frac{f(a)}{2} + \sum_{i=1}^n f(a + ih) + \frac{f(b)}{2} \right).$$

Doing the integral using the simpsons rule we get,

$$I(f) \approx I_{\text{simpson}}(f) = \frac{h}{3} \left(\frac{f(a)}{2} + 4 \sum_{i=1}^{\frac{n}{2}} f(a + (2i-1)h) + 2 \sum_{i=1}^{\frac{n}{2}-1} f(a + 2ih) + \frac{f(b)}{2} \right).$$

(ii)