Chapter 1

Pseudospectral Methods

1.1 Least Squares Approach

The **Least Squares approach** is a mathematical method used to find the best-fitting solution to a set of data by minimizing the sum of the squares of the differences between the observed and predicted values. Suppose you have a set of observed data points (x_i, y_i), and you want to fit a model $f(x)$ to these data points. The least squares method minimizes the following sum:

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{1.1}$$

In the least squares method, the approximated function is not guaranteed to be equal to the value of the actual function at the node points.

1.2 Interpolation

Interpolation is a mathematical technique used to estimate unknown values that fall within the range of known data points. It is widely used in numerical analysis, data science, engineering, and other fields where it is necessary to predict values for intermediate points based on a set of discrete data points. Unlike the least squares approach, in interpolation, the modeled function passes through the selected data points.

One common interpolation method is the **Lagrange Interpolation**, which constructs the interpolating polynomial using a linear combination of Lagrange basis polynomials. The interpolating polynomial $P(x)$ is given by:

$$P(x) = \sum_{i=0}^{n} y_i \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \tag{1.2}$$

1.3 Class Discussion - MATLAB Plot: Least Squares vs Interpolation

As discussed in class, the interpolation method ensures that the value of the approximated function is exactly the value of the actual function at the selected node points. Figure

1.1 shows the difference between the two approximation methods (Least Squares and Interpolation) via a simple MATLAB plot discussed during the lecture.

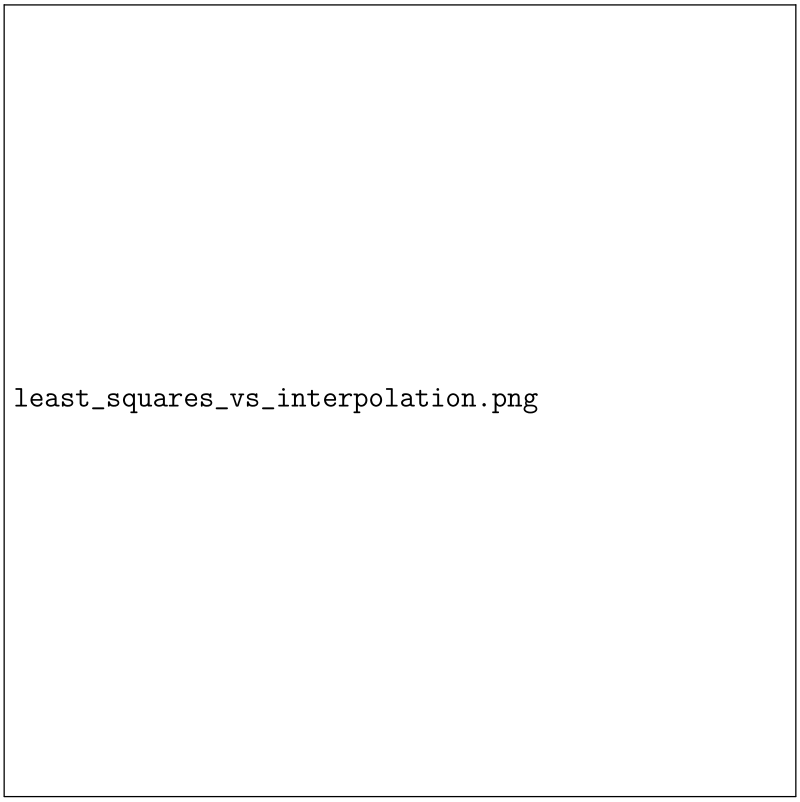

Figure 1.1: Comparison of Least Squares and Interpolation Methods.

1.4 Collocation Method

The collocation method is a technique used to approximate solutions of differential equations by forcing the solution to satisfy the differential equation at a finite number of points called collocation points. For a given set of nodes, the polynomial can be approximated in either of the two ways:

$$P_n(x) = \sum_{i=0}^{n} a_i f(x_i) \tag{1.3}$$

or

$$P_n(x) = \sum_{i=0}^{n} b_i x_i \tag{1.4}$$

Both of these are types of interpolation matrices, and the value of the approximated polynomial is the same as the actual function at the node points.

1.4.1 Vandermonde Matrix

Given a vector of elements $x = [x_1, x_2, \ldots, x_n]$, the Vandermonde matrix V associated with this vector is an $n \times n$ matrix, where each row corresponds to a geometric progression of the elements x_i. Specifically, the matrix is defined as follows:

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{pmatrix}$$

Each element V_{ij} in the matrix is given by:

$$V_{ij} = x_i^{j-1} \tag{1.5}$$

The determinant of the Vandermonde matrix has a special form and is given by:

$$\det(V) = \prod_{1 \leq i < j \leq n} (x_j - x_i) \tag{1.6}$$

This product is non-zero if and only if all the x_i are distinct, meaning the determinant of the Vandermonde matrix is non-zero (and hence the matrix is invertible) if all the elements in the vector x are distinct. The proof for this will be discussed later.

1.5 Runge Phenomenon

When interpolating a set of points, one common approach is to fit a polynomial that passes through all the given data points. The idea is that the polynomial will closely approximate the underlying function from which the data points are sampled. However, when the function is sampled at equally spaced points and a high-degree polynomial is used, the interpolation error can become quite significant near the endpoints of the interval. This is known as **Runge's Phenomenon**.

1.6 Chebyshev Nodes

Chebyshev nodes are specific points used in numerical interpolation, particularly in polynomial interpolation, to reduce the problems associated with Runge's phenomenon and to achieve more accurate approximations.

Chebyshev nodes are the roots of Chebyshev polynomials of the first kind. For a given interval $[-1, 1]$, the Chebyshev nodes x_i for n interpolation points are defined as:

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \quad i = 1, 2, \ldots, n \tag{1.7}$$

These nodes are spaced more densely near the endpoints of the interval $[-1, 1]$ and more sparsely near the center. When interpolating over a different interval $[a, b]$, you can linearly transform the Chebyshev nodes from $[-1, 1]$ to $[a, b]$ using the transformation:

$$x_i' = \frac{b-a}{2}(x_i + 1) + a$$

1.7 Weierstrass Theorem

The **Weierstrass Approximation Theorem** states that for every continuous function f defined on a closed interval $[a, b]$, and for every $\epsilon > 0$, there exists a polynomial $P(x)$ such that:

$$\|f(x) - P(x)\| < \epsilon \tag{1.8}$$

for all x in $[a, b]$. In other words, any continuous function on a closed interval can be uniformly approximated as closely as desired by a polynomial function. The proof of the Weierstrass Approximation Theorem typically involves constructing a sequence of polynomials that converge uniformly to the given continuous function. One standard approach uses Bernstein polynomials, which are constructed from binomial coefficients and shown to converge uniformly to the function $f(x)$.

1.8 Trapezoidal Method

The **Trapezoidal Method** is used to approximate the area under a curve by dividing the region into a series of trapezoids. The area of each trapezoid is computed and summed to give an approximation of the definite integral.

Consider a function $f(x)$ defined on the interval $[a, b]$. The goal is to approximate the definite integral:

$$\int_a^b f(x)dx \tag{1.9}$$

Using the trapezoidal method, the interval $[a, b]$ is divided into n subintervals of equal width:

$$h = \frac{b-a}{n}$$

The points dividing the interval are:

$$x_0 = a, \quad x_1 = a + h, \quad x_2 = a + 2h, \ldots, x_n = b$$

The trapezoidal rule then approximates the integral as:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right] \tag{1.10}$$

This formula sums the function values at the endpoints and the midpoints, weighted appropriately, to give a numerical estimate of the integral.

1.9 Trapezoidal Method

1.9.1 Example

The Trapezoidal method provides good accuracy for periodic functions and becomes exact after a certain number of nodes are used for discretization. (The proof for this can

be found in the class notes on Moodle.) Figure 1.2 demonstrates the accuracy of the trapezoidal method, as shown by the MATLAB code discussed in class.

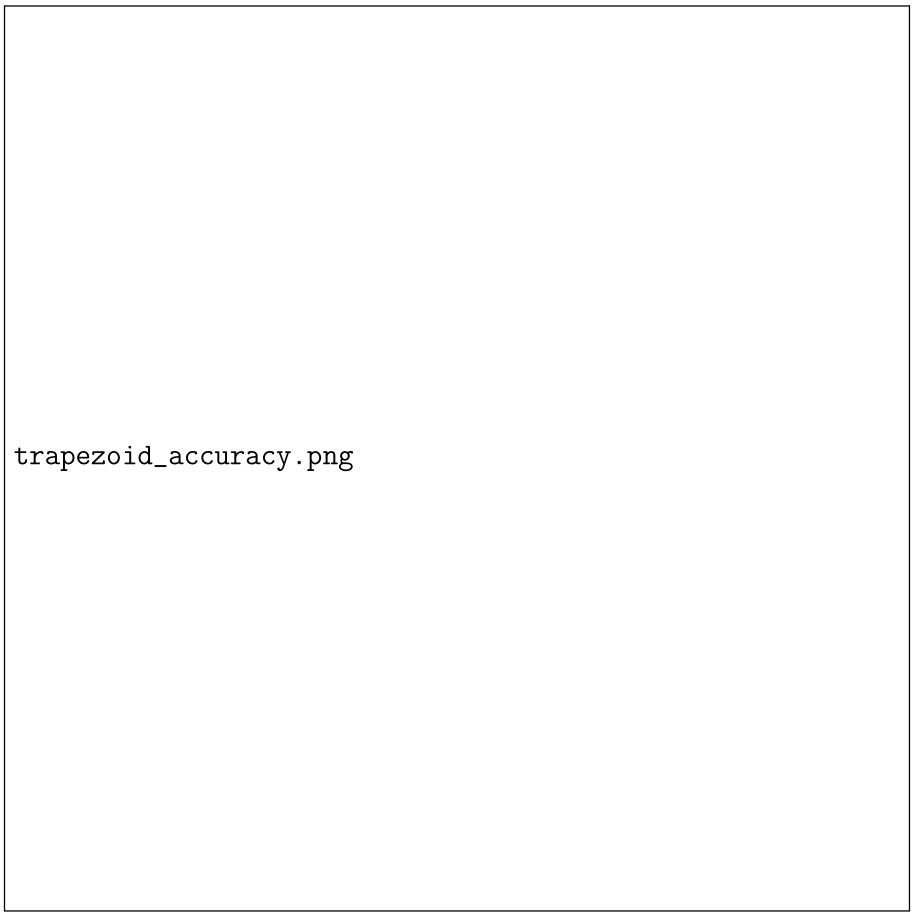

Figure 1.2: Trapezoid method accuracy.

1.10 Central Difference Method

The Central Difference method is a finite difference scheme used to approximate derivatives by expanding around two neighboring points. It is second-order accurate, meaning it approximates derivatives to second-order accuracy in terms of the grid spacing Δx.

1.10.1 Expansion Around $x_{i+1} = x_i + \Delta x$

By expanding the function $f(x)$ around $x_{i+1} = x_i + \Delta x$, we obtain the following Taylor series expansion:

$$f(x_{i+1}) = f(x_i+\Delta x) = f(x_i)+\frac{f'(x_i)}{1!}\Delta x+\frac{f''(x_i)}{2!}(\Delta x)^2+\frac{f'''(x_i)}{3!}(\Delta x)^3+O((\Delta x)^4) \quad (1.11)$$

1.10.2 Expansion Around $x_{i-1} = x_i - \Delta x$

Similarly, expanding the function around $x_{i-1} = x_i - \Delta x$, we get:

$$f(x_{i-1}) = f(x_i - \Delta x) = f(x_i) - \frac{f'(x_i)}{1!}\Delta x + \frac{f''(x_i)}{2!}(\Delta x)^2 - \frac{f'''(x_i)}{3!}(\Delta x)^3 + O((\Delta x)^4) \tag{1.12}$$

1.10.3 Central Difference Formula

Taking the difference between $f(x_{i+1})$ and $f(x_{i-1})$, we get:

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)\Delta x + \frac{2f'''(x_i)}{6}(\Delta x)^3 + O((\Delta x)^4) \tag{1.13}$$

Approximating over small intervals, the first derivative $f'(x_i)$ is approximated as:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} \tag{1.14}$$

Thus, the central difference scheme provides exact results if the highest order of the function is cubic.

1.10.4 Higher-Order Central Difference

By increasing the size of the stencil (i.e., including more neighboring points), we can reduce the approximation error and improve the accuracy of the method. As the stencil size increases, the error tends toward zero for higher-order functions.

1.11 Example Problem

Consider a block with the following boundary conditions:

$$x(0) = 0, \quad v(0) = 0, \quad x(1) = 1, \quad v(1) = 0$$

The system is governed by the equations:

$$\dot{x} = v, \quad \dot{v} = u$$

The objective is to optimize the following function:

$$J = \int_0^1 U(x)^2 \, dx$$

Using the trapezoidal method, we can approximate the integral as:

$$J = \left(\frac{U_0^2}{2} + U_1^2 + \cdots + U_{n-1}^2 + \frac{U_n^2}{2} \right) \Delta t$$

Two Methods for Non-Linear Equality Constraints:

1. Central Difference Method In the central difference method, the constraint conditions are formed as follows for the interior points:

$$\frac{x_{k+1} - x_{k-1}}{2h} = v_k, \quad \frac{v_{k+1} - v_{k-1}}{2h} = u_k$$

For the boundary points, forward and backward Euler methods are used to satisfy the conditions at the boundaries.

2. Trapezoidal Method In the trapezoidal method, the constraint conditions are written as:

$$x_{k+1} - x_k = \frac{v_{k+1} + v_k}{2}\Delta t, \quad v_{k+1} - v_k = \frac{u_{k+1} + u_k}{2}\Delta t$$

The full implementation of this method can be found in the `nonlincon.m` file, as discussed in class.

To perform the optimization, we use the `fmincon` function from MATLAB. `fmincon` is used for solving constrained nonlinear optimization problems, and it finds the minimum of a scalar function subject to the specified constraints.

1.12 Discussion of MATLAB and Mathematica Presentation in Class

With the same boundary conditions as the previous problem, a related problem discussed in class has 5 unknowns and 4 conditions, making it an optimization problem. This requires the use of similar numerical methods for solution.

1.13 Rolle's Theorem

Rolle's Theorem is a fundamental result in calculus that provides a specific condition under which a function must have a horizontal tangent line (or a derivative equal to zero) at least once within a given interval.

The theorem states that if a continuous function $f(x)$ is differentiable on the open interval (a, b) and satisfies $f(a) = f(b)$, then there exists at least one point $c \in (a, b)$ such that:

$$f'(c) = 0$$

This result is often used in optimization and root-finding algorithms.

1.14 Rolle's Theorem

Rolle's Theorem is a fundamental result in calculus that provides a specific condition under which a function must have a horizontal tangent line (or a derivative equal to zero) at least once within a given interval. It's a special case of the Mean Value Theorem.

Let $f(x)$ be a function that satisfies the following three conditions:

1. **Continuity:** $f(x)$ is continuous on the closed interval $[a, b]$.
2. **Differentiability:** $f(x)$ is differentiable on the open interval (a, b).
3. **Equal Endpoints:** $f(a) = f(b)$.

If these conditions are met, then there exists at least one point $c \in (a, b)$ such that the derivative of $f(x)$ at c is zero:

$$f'(c) = 0$$

1.15 Derivatives of Lagrange Polynomial

Consider a function approximated with a second-order Lagrange polynomial using equally spaced nodes at 0, h, and $2h$. The polynomial can be expressed as:

$$P(x) = y_0 \cdot \ell_0(x) + y_1 \cdot \ell_1(x) + y_2 \cdot \ell_2(x)$$

Where the Lagrange basis polynomials are defined as follows:

$$\ell_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$\ell_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$\ell_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

The derivative of the polynomial $P'(x)$ is given by:

$$P'(x) = y_0 \cdot \frac{2x - (x_1 + x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \cdot \frac{2x - (x_0 + x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \cdot \frac{2x - (x_0 + x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

To represent the derivatives at the node points in matrix format, we can express it as:

$$\begin{bmatrix} P'(x_0) \\ P'(x_1) \\ P'(x_2) \end{bmatrix} = \begin{bmatrix} -\frac{3}{2h} & \frac{2}{h} & -\frac{1}{2h} \\ -\frac{1}{2h} & 0 & \frac{1}{h} \\ \frac{1}{h} & -\frac{2}{h} & \frac{3}{2h} \end{bmatrix} \begin{bmatrix} P(x_0) \\ P(x_1) \\ P(x_2) \end{bmatrix}$$

This shows that the derivatives of the approximated function at the node points are a linear combination of the actual function values at the node points.

1.16 Controlling Error by Altering the Location of Nodes

Claim: When we approximate a function using a polynomial of order n, the error in the representation can be altered by changing the position of the nodes (collocation points).

Proof: Consider the error term defined as:

$$W(t) = f(t) - p_n(f; t) - (t - x_0)(t - x_1) \cdots (t - x_n)K(x)$$

Where $K(x)$ is defined as:

$$K(x) = \frac{f(x) - p_n(f;x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}$$

For an n-th order polynomial, we have $(n+1)$ variables. The function W is zero at:

- $t = x_i$ for all $i = 0, 1, \ldots, n$ (the node points)
- $t = x$

By applying Rolle's Theorem, there exists some ξ in the interval between x_0 and x_n such that:

$$W^{(n+1)}(t) = f^{(n+1)}(t) - (n+1)!K(x)$$

Thus, at point ξ:

$$0 = W^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!K(x)$$

From this, we can express $K(x)$ as:

$$K(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)$$

Consequently, the error $e(x)$ can be written as:

$$e(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n)$$

This shows that while we do not have direct control over $f^{(n+1)}$ to reduce the error, we can alter the location of the nodes to change the error.

Figure 4 shows the class representation in Mathematica, illustrating that for a function approximated using a second-order polynomial, taking the derivative at the central control point is the same as applying the central difference at that point.

1.17 Difference Between a Polynomial and Its nth Order Interpolation Approximation

The error in approximating a function $f(x)$ using an nth order polynomial $p_n(x)$ can be expressed as:

$$e(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n)$$

From MATLAB simulations, we found that the optimal node locations for approximating a 2D polynomial with a linear Lagrange polynomial (in the symmetric case) are as follows: - For least RMS error: $x_0 = -\sqrt{\frac{1}{3}}$ and $x_1 = \sqrt{\frac{1}{3}}$ - For least maximum error: $x_0 = -\sqrt{\frac{1}{2}}$ and $x_1 = \sqrt{\frac{1}{2}}$

Figure 5 shows the Mathematica representation of these findings as discussed in class. A similar analysis applies to a 3D polynomial; refer to the MATLAB code for the optimal node points for a third-order polynomial function.

Example Problem

Consider a polynomial function X_{n+1} represented using an nth order polynomial:

$$P_n = \sum_{i=0}^{n} a_i x^i$$

We need to find the leading coefficient of the approximation, a_n. Equally spaced node points are chosen from 0 to 1 (including both endpoints).

Solution: Using the remainder theorem:

$$X_{n+1} - \sum_{i=0}^{n} a_i x^i = A(x-0)(x-\frac{1}{n})(x-\frac{2}{n})\cdots(x-\frac{n-1}{n})(x-1)$$

Here, A comes out to be 1. Comparing the coefficients on the left-hand side and the right-hand side, we find the leading coefficient $a_n = \frac{n+1}{2}$.

1.18 Condition for Vandermonde Matrix Being Invertible

The Vandermonde matrix is always invertible except when two node points are the same. To understand this, we can observe the Vandermonde matrix defined as:

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

If two node points become the same, two rows of the matrix become identical, causing the determinant to be zero, thus making the matrix non-invertible.

Now, consider the Vandermonde matrix with the last row replaced by x instead of x_n:

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x & x^2 & \cdots & x^n \end{pmatrix}$$

If x takes any of the values $x_0, x_1, \ldots, x_{n-1}$, the determinant is zero. Thus, the determinant can be expressed as:

$$\det(V) = A\prod_{k=0}^{n-1}(x_n - x_k)$$

Where A is a coefficient that does not contain any x term. Further observation shows that A is the determinant of a Vandermonde matrix with n terms, applying similar operations to x_n for x_{n-1} now:

$$\det(V) = (x_1 - x_0)\prod_{k=0}^{1}(x_2 - x_k)\prod_{k=0}^{2}(x_3 - x_k)\cdots\prod_{k=0}^{n-1}(x_n - x_k)$$

This form proves that unless two nodes are identical, the determinant of the Vandermonde matrix will exist.

1.19 Lagrange Basis Polynomial

1.19.1 Cramer's Rule

Consider a system of linear equations:

$$Ax = b$$

where: - A is an $n \times n$ matrix of coefficients, - x is a column vector of unknowns, - b is a column vector of constants.

Cramer's Rule states that if $\det(A) \neq 0$, the solution to the system is given by:

$$x_i = \frac{\det(A_i)}{\det(A)} \quad \text{for } i = 1, 2, \ldots, n,$$

where A_i is obtained by replacing the i-th column of A with b.

Proof

For a particular m, consider:

$$x_m = \sum_{j=0}^{n} W(x_j) x_{j,m}$$

where $0 \leq m \leq n$. Note that even though for all m less than n, it appears that the right-hand side will have a higher power of x than m due to the Lagrange polynomial bases, the terms cancel out to match the power on the left-hand side.

The aim is to prove that

$$P_n(x) = \sum_{i=0}^{n} a_i x^i$$

can be represented as

$$P_n(x) = \sum_{i=0}^{n} W_i f(x_i)$$

where the W_i are the Lagrange bases.

Thus,

$$P_n(x) = \sum_{k=0}^{n} a_k(x_k)$$

can be rearranged as follows:

$$P_n(x) = \sum_{k=0}^{n} a_k \sum_{j=0}^{n} W(x_j) x_k^j$$

Rearranging gives:

$$P_n(x) = \sum_{j=0}^{n} W(x_j) \sum_{k=0}^{n} a_k x_j^k$$

Therefore,

$$P_n(x) = \sum_{j=0}^{n} W(x_j) P(x_j)$$

To find the Lagrange basis, we write the system of equations in a matrix format:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} & 1 & x_{n-1}^2 & \cdots & x_{n-1}^n \end{pmatrix} \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ \vdots \\ W_n \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Notably, the Vandermonde matrix V is given by:

$$V^T = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-1} & 1 & x_{n-1}^2 & \cdots & x_{n-1}^n \end{pmatrix}$$

Using Cramer's rule, we know that:

$$W_j = \frac{\Delta_j}{\Delta}$$

where Δ is the determinant of the Vandermonde matrix and Δ_j is the determinant of the Vandermonde matrix with the elements x_j replaced with x.

To find the Lagrange polynomials:

$$\Delta = \det(V) = (x_1 - x_0) \prod_{k=1}^{n-1} (x_2 - x_k) \prod_{k=2}^{n-1} (x_3 - x_k) \cdots \prod_{k=n-1}^{n-1} (x_n - x_k)$$

Separating all terms that have an x_j in them:

$$\det(V) = \lambda_j \prod_{k=0}^{j-1} (x_j - x_k) \prod_{k=j+1}^{n} (x_k - x_j)$$

Here, λ_j is the part of the determinant that does not have any x_j terms. Replacing x_j with x:

$$\Delta_j = \det(V) = \lambda_j \prod_{k=0}^{j-1} (x - x_k) \prod_{k=j+1}^{n} (x_k - x)$$

Thus,

$$W(x_j) = \ell_j(x) = \prod_{\substack{0 \le k \le n \\ k \ne j}} \frac{x - x_k}{x_j - x_k}$$

We can conclude that all the Lagrange basis polynomials are of the same order, and for $n+1$ node points over which the polynomial is approximated, the basis function will be of order n. Note that the Lagrange basis polynomial ℓ_j has a value of 1 at the node x_j and zero at all other node points. The polynomial cannot have more zeros than the number of remaining node points, as verified from the degree of the polynomial.

1.19.2 Important Points

Intuitive Understanding of Why the Sum of Weights is One

Every approximation of a function must be able to model a constant function as well. Thus, the sum of weights or the coefficients must equal one. This can be mathematically represented as:

$$P(x) = \sum_{i=0}^{n} W_k(x) f(x_k)$$

For a constant function, say $f(x) = c$:

$$P(x) = c = \sum_{i=0}^{n} W_k(x) c$$

Canceling c on both sides gives:

$$\sum_{i=0}^{n} W_k(x) = 1$$

1.20 Derivatives of Lagrange Polynomial/ Differentiation Matrix

The derivative of a Lagrange polynomial can be computed using specific formulas, which are essential in interpolation problems.