1 Introduction to Data Fitting

Data fitting is a fundamental topic in scientific computing and numerical methods. It concerns constructing a continuous function that represents a given discrete set of data points. This process is essential for interpreting data, discovering hidden parameters, finding underlying trends, and processing data for subsequent analysis such as differentiation or integration.

There are two primary approaches to data fitting:

- **Interpolation**: Fitting a function that matches the given data points exactly.
- **Least Squares**: Minimizing the error between the function and the given data, particularly useful when dealing with noisy data, such as experimental measurements.

We will explore these approaches, starting with motivation and then delving into the details of interpolation and least squares methods.

1.1 Motivation

Consider a scenario where we are given a set of discrete data points. We often need to estimate values at points between these given data points.

1.1.1 Piecewise Linear Interpolation

The simplest approach is to "connect the dots," which is known as piecewise linear interpolation. This method creates a continuous function by linearly interpolating between consecutive data points.

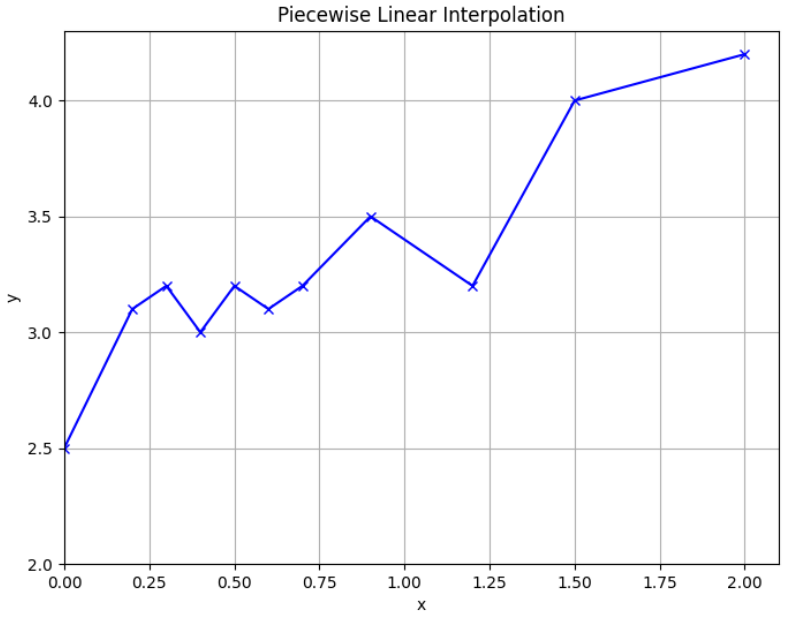

Figure 1: Piecewise Linear Interpolation.

While straightforward and useful in many cases, piecewise linear interpolation introduces "kinks" at each data point, which might be undesirable if a smoother approximation is required.

1.1.2 Polynomial Interpolation

To obtain a smoother approximation, we can consider polynomial interpolation. Given $n + 1$ data points, we can construct a unique polynomial of degree n that passes through all these points. For example, with 11 data points, we can use a degree 10 polynomial.

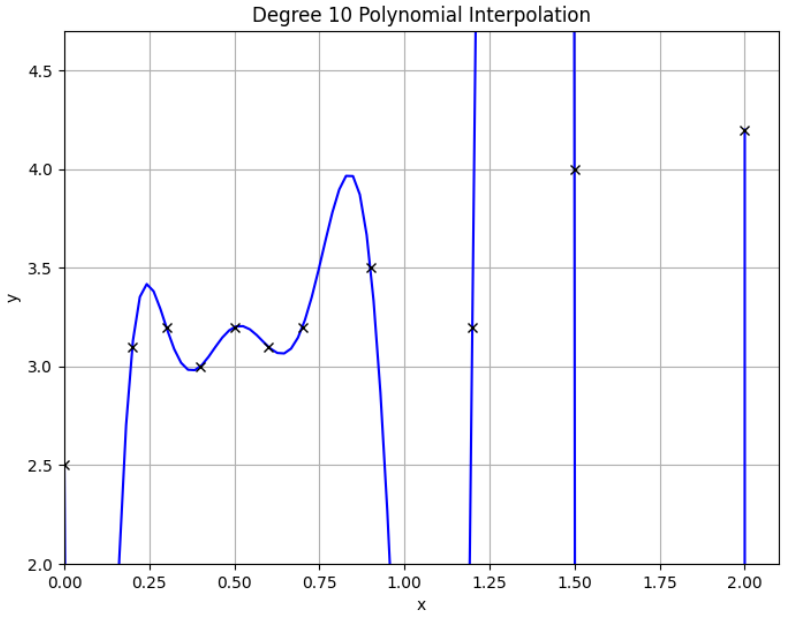

Figure 2: Degree 10 Polynomial Interpolation.

While smoother, high-degree polynomial interpolations can exhibit undesirable oscillations, especially at the ends of the interval, which may not accurately reflect the underlying data.

1.1.3 Least Squares Fitting

Another approach is to use least squares fitting, which is particularly useful when we suspect noise in our data or when we want to capture the general trend rather than exactly passing through all data points.

Linear Regression Linear regression attempts to fit a straight line to the data by minimizing the error between the line and the data points.

As seen in Figure 3, a linear fit might not capture the complexities of the data and may systematically miss important features.

Generalizing to Higher Order Polynomials Linear regression can be generalized to fit higher-order polynomials using least squares.

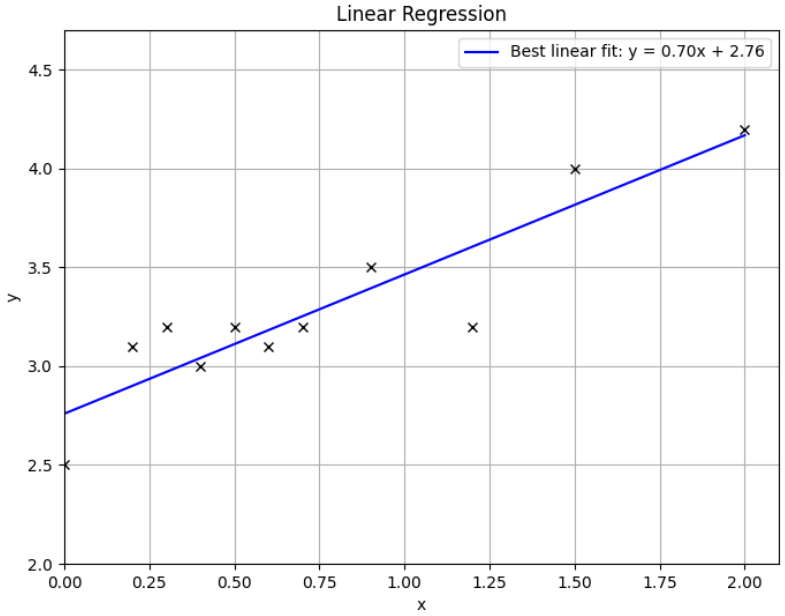

Figure 3: Linear Regression: Best linear fit: $y = 0.24x + 2.94$. Clearly not a good fit!

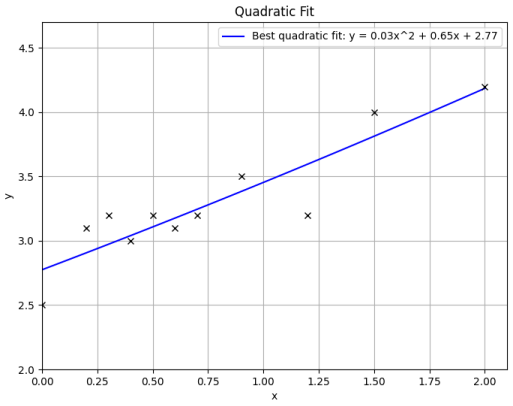

Figure 4: Quadratic Fit: Best quadratic fit: $y = 0.53x^2 - 0.83x + 3.27$. Still not so good ...

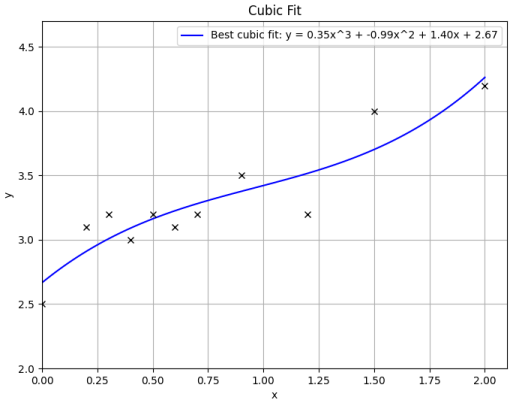

Figure 5: Cubic Fit: Best cubic fit: $y = 0.93x^3 - 2.27x^2 + 1.31x + 3.00$. Looks good! A "cubic model" captures this data well.

As we increase the polynomial order, such as moving from a quadratic fit (Figure 4) to a cubic fit (Figure 5), the fit to the data improves significantly. A cubic model appears to capture the trends in the data well without exhibiting excessive oscillations. In real-world problems, choosing the "right" model for experimental data can be challenging and depends on factors like the presence of noise and the complexity of the underlying phenomenon.

Data Fitting with Multi-Dimensional Data Data fitting is not limited to scalar functions; it can also be extended to multi-dimensional data. For example, we might want to fit a 2D surface to a 3D cloud of data points, finding the best hypersurface in $\mathbb{R}^N$.

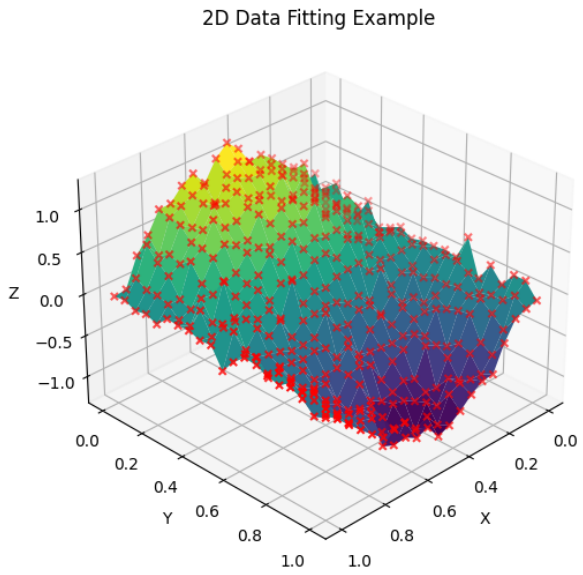

Figure 6: 2D Data Fitting Example: Data fitting is often performed with multi-dimensional data (find the best hypersurface in $\mathbb{R}^N$). 2D Example.

2 Interpolating Discrete Data

We now focus on polynomial interpolation in more detail. Let P_n be the set of all polynomials of degree n on the real numbers. An n^{th} degree polynomial $p(x) \in P_n$ can be written as:

$$p(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$$

where $b_0, b_1, \ldots, b_n$ are the coefficients. We can represent these coefficients as a vector $b = [b_0, b_1, \ldots, b_n]^T \in \mathbb{R}^{n+1}$.

Suppose we are given $n+1$ data points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, which we denote as a set $S = \{(x_i, y_i)\}_{i=0}^n$. We want to find a polynomial $p(x) \in P_n$ such that it passes through all these data points, i.e., $p(x_i) = y_i$ for $i = 0, 1, \ldots, n$. These x_i values are called interpolation points.

2.1 Vandermonde Matrix Approach

The condition $p(x_i) = y_i$ for each data point gives us a system of $n+1$ linear equations:

$$\begin{aligned} b_0 + b_1 x_0 + b_2 x_0^2 + \cdots + b_n x_0^n &= y_0 \\ b_0 + b_1 x_1 + b_2 x_1^2 + \cdots + b_n x_1^n &= y_1 \\ &\vdots \\ b_0 + b_1 x_n + b_2 x_n^2 + \cdots + b_n x_n^n &= y_n \end{aligned}$$

This system can be written in matrix form as $Vb = y$, where:

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix}, \quad b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{pmatrix}, \quad y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Here, V is the Vandermonde matrix, b is the vector of polynomial coefficients we want to find, and y is the vector of the given y-values.

Theorem 2.1. For any set of $n+1$ distinct interpolation points $\{x_0, x_1, \dots, x_n\}$, the Vandermonde system $Vb = y$ has a unique solution.

Preuve. To prove uniqueness, we need to show that if $Vz = 0$ implies $z = 0$. Suppose $Vb = 0$. This means that the polynomial $p(x) = b_0 + b_1 x + \cdots + b_n x^n$ satisfies $p(x_i) = 0$ for $i = 0, 1, \dots, n$. Thus, $p(x)$ has $n+1$ roots at $x_0, x_1, \dots, x_n$. A polynomial of degree n can have at most n roots unless it is the zero polynomial. Therefore, $p(x)$ must be the zero polynomial, which implies all coefficients $b_i = 0$. Hence, $b = 0$ is the only solution to $Vb = 0$, and V is non-singular, ensuring a unique solution for $Vb = y$. □

Although a unique solution exists, solving the Vandermonde system directly is generally not used in practice because the Vandermonde matrix is often ill-conditioned.

2.2 Ill-Conditioning of the Vandermonde Matrix

The Vandermonde matrix is ill-conditioned, especially for higher degrees, because it corresponds to using the monomial basis $\{1, x, x^2, \dots, x^n\}$ for polynomial interpolation.

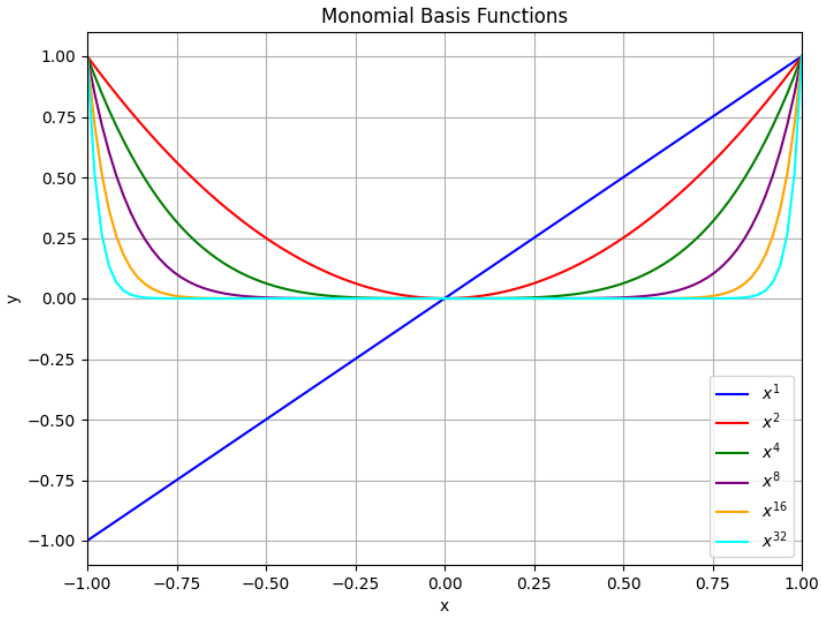

Figure 7: Monomial Basis Functions: Monomial basis functions become increasingly indistinguishable for higher powers.

As shown in Figure 7, monomial basis functions, such as x^{14} and x^{16} (or x^7 and x^9), become increasingly similar in shape, especially on intervals like $[-1, 1]$. This near linear dependence of the columns of the Vandermonde matrix leads to ill-conditioning.

2.2.1 Condition Number of Vandermonde Matrix

The condition number quantifies the sensitivity of the solution of a linear system to perturbations in the input. A large condition number indicates ill-conditioning. Let's examine the condition number of Vandermonde matrices using Python.

Listing 1: Condition number calculation for a diagonal matrix

```
>>> import numpy as np
>>> a = np.array([[0.5,0.0,0.0],[0.0,2.0,0.0],[0.0,0.0,3.0]])
>>> np.linalg.cond(a)
6.0
```

For a diagonal matrix, the condition number is the ratio of the largest to the smallest diagonal element, which is $\frac{3}{0.5} = 6$ in this example, as expected.

Now, let's calculate the condition number for Vandermonde matrices.

Listing 2: Condition number calculation for a 5x5 Vandermonde matrix

```
>>> x=np.linspace(0,1,5)
>>> np.vander(x)
array([[1.        , 0.        , 0.        , 0.        , 0.        ],
       [1.        , 0.25      , 0.0625    , 0.015625  , 0.00390625],
       [1.        , 0.5       , 0.25      , 0.125     , 0.0625    ],
       [1.        , 0.75      , 0.5625    , 0.421875  , 0.31640625],
       [1.        , 1.        , 1.        , 1.        , 1.        ]])
>>> np.linalg.cond(np.vander(x))
686.07...
```

For a 5x5 Vandermonde matrix with points linearly spaced in [0, 1], the condition number is approximately 686, which is already quite large.

Increasing the size to a 10x10 Vandermonde matrix dramatically increases the condition number.

Listing 3: Condition number calculation for a 10x10 Vandermonde matrix

```
>>> x=np.linspace(0,1,10)
>>> np.linalg.cond(np.vander(x))
1.5497...e+07
```

The condition number is now around 15 million.

Changing the interval of interpolation points, e.g., to [1, 2], further worsens the conditioning.

Listing 4: Condition number calculation for a 10x10 Vandermonde matrix with points in [1

```
>>> x=np.linspace(1,2,10)
>>> np.linalg.cond(np.vander(x))
2.675...e+11
```

The condition number becomes approximately 267 billion, demonstrating severe ill-conditioning.

2.2.2 Practical Consequences of Ill-Conditioning

Due to finite precision arithmetic, solving $Vb = y$ numerically yields an approximation $\hat{b}$. While a stable numerical method ensures a small residual $||V\hat{b} - y||$, the error in the coefficients $||b - \hat{b}||$ can be large if V is ill-conditioned. Small perturbations in $\hat{b}$ can lead to large perturbations in $V\hat{b}$ and consequently, large residuals.

Consider a 2D vector space with basis vectors $v_1 = (1, 0)$ and $v_2 = (1, 0.0001)$. These basis vectors are nearly linearly dependent.

Let $y = (1, 0)$ and $\tilde{y} = (1, 0.0005)$. In the basis $\{v_1, v_2\}$, we express $y = b_1v_1 + b_2v_2$ and $\tilde{y} = \tilde{b}_1v_1 + \tilde{b}_2v_2$.

For $y = (1, 0)$: $b = (1, 0)$ because $y = 1 \cdot v_1 + 0 \cdot v_2$.

For $\tilde{y} = (1, 0.0005)$: we need to solve $\begin{pmatrix} 1 & 1 \\ 0 & 0.0001 \end{pmatrix} \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0.0005 \end{pmatrix}$. From the second equation, $0.0001\tilde{b}_2 = 0.0005 \implies \tilde{b}_2 = 5$. From the first equation, $\tilde{b}_1 + \tilde{b}_2 = 1 \implies \tilde{b}_1 = 1 - \tilde{b}_2 = 1 - 5 = -4$. Thus, $\tilde{b} = (-4, 5)$.

Even though y and $\tilde{y}$ are very close, their coefficients $b = (1, 0)$ and $\tilde{b} = (-4, 5)$ are quite different. This illustrates the sensitivity to perturbations caused by an ill-conditioned basis.

2.3 Horner's Method for Polynomial Evaluation

To efficiently evaluate a polynomial $p(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$, we can use Horner's method. For a cubic polynomial $ax^3 + bx^2 + cx + d$, direct evaluation is:

$$ax^3 + bx^2 + cx + d = ax \cdot x \cdot x + bx \cdot x + cx + d$$

Horner's method rewrites it to minimize multiplications and additions:

$$ax^3 + bx^2 + cx + d = ((ax + b)x + c)x + d$$

This reduces the number of operations and is computationally more efficient and numerically stable, especially for higher-degree polynomials. For an n^{th} degree polynomial, Horner's method requires n additions and n multiplications.

2.4 Lagrange Interpolation

To overcome the ill-conditioning issues of the Vandermonde approach, we can use Lagrange interpolation, which employs a well-conditioned basis. The key idea is to construct Lagrange basis polynomials $L_k(x)$ for $k = 0, 1, \ldots, n$ such that:

$$L_k(x_i) = \begin{cases} 1 & \text{if } i = k \\ 0 & \text{if } i \neq k \end{cases}$$

These Lagrange basis polynomials are given by:

$$L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

Using these basis polynomials, the polynomial interpolant $p_n(x)$ can be directly constructed as:

$$p_n(x) = \sum_{k=0}^{n} y_k L_k(x)$$

This form ensures that $p_n(x_i) = \sum_{k=0}^{n} y_k L_k(x_i) = y_i L_i(x_i) = y_i \cdot 1 = y_i$, since $L_k(x_i) = 0$ for $k \neq i$ and $L_i(x_i) = 1$.

Figure 8 illustrates Lagrange basis polynomials. For instance, the blue curve ($L_2(x)$) is 1 at the third interpolation point and 0 at all others, and similarly for the red curve ($L_5(x)$), it is 1 at the last interpolation point and 0 elsewhere, satisfying the desired properties. Lagrange interpolation provides a well-conditioned approach to polynomial interpolation.

3 Conclusion

Data fitting is a crucial tool in scientific computing, with interpolation and least squares methods offering different approaches for constructing continuous functions from discrete data. While Vandermonde interpolation, based on monomial basis, is conceptually straightforward, it suffers from ill-conditioning, especially for higher degrees. Lagrange interpolation provides a more stable and well-conditioned alternative by using Lagrange basis polynomials. For noisy data or when capturing trends is more important than exact interpolation, least squares fitting, including linear and higher-order polynomial regression, becomes invaluable. Efficient and stable algorithms for these methods are essential, especially when dealing with large datasets in real-world applications.

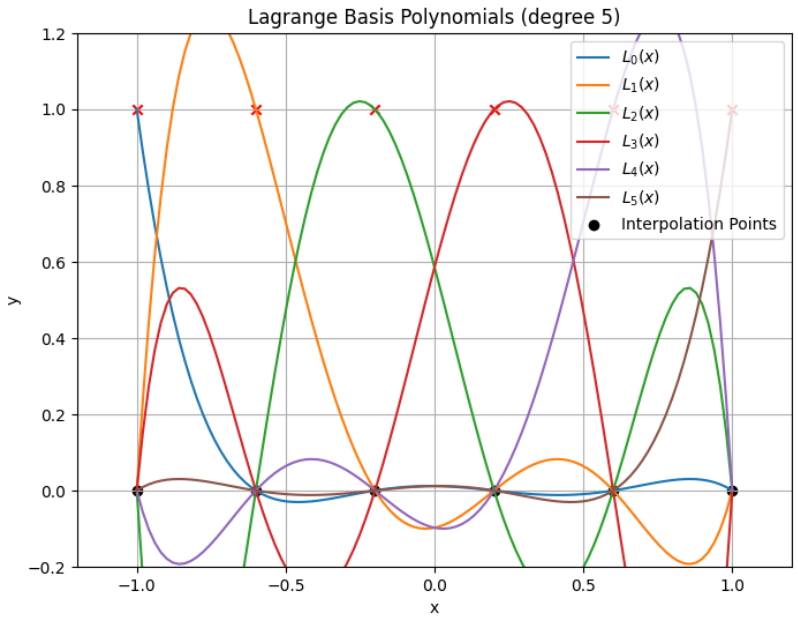

Figure 8: Lagrange Basis Polynomials: Example of Lagrange basis polynomials of degree 5, using 6 interpolation points over [-1, 1].