

*Computational Methods*  
CMSC/AMSC/MAPL 460

Polynomial Interpolation

Ramani Duraiswami,  
Dept. of Computer Science

# Interpolation

- Given a function at  $N$  points, find its value at other point(s)
  - “within” the points
    - Interpolation
  - “outside” the points
    - Extrapolation
- How do we extend the value from known points to unknown points?
  - Have to have prior knowledge about the function
  - Can do what is convenient
- Occam’s razor
  - Parsimony: “One should not increase, beyond what is necessary, the number of entities required to explain anything”
  - *Key in all scientific modeling*
  - Will return to this issue

# Function representation: Polynomials

- In interpolation sampled values of a function are available
- We need to extend these values to points where they are not available
- Can assume that the function is expanded over a basis of functions which span the functional space
  - Polynomials are a basis
  - Fourier series are another
  - Many others
- Most common basis are power series/polynomials
  - Here  $x_*$  is a point about which we expand in series
  - $a_m$  are coefficients
- Polynomial interpolation often preferred
  - Part of theory of Taylor series, solution of differential equations via power series, in computing integrals
  - A theorem guaranteeing that a polynomial can represent a function is available

$$f(y - x_*) = \sum_{m=0}^{\infty} a_m (y - x_*)^m,$$

# Properties of Power Series

- 1) For any power series there exists  $r_*$ , such that the series converges absolutely at  $|y - x_*| < r_*$ , and diverges at  $|y - x_*| > r_*$ .
  - The number  $r_*$ , is called *the convergence radius* of the series,
  - $0 \leq r_* \leq \infty$ .
  - For any number  $q$ , such that  $0 < q < r_*$ , the power series uniformly converges at  $|y - x_*| < q$ .
- 2) Convergent power series can be summed, multiplied by a scalar, or multiplied according to the Cauchy rule.

$$\sum_{m=0}^{\infty} a_m (y - x_*)^m + \sum_{m=0}^{\infty} b_m (y - x_*)^m = \sum_{m=0}^{\infty} (a_m + b_m) (y - x_*)^m,$$

$$\alpha \sum_{m=0}^{\infty} a_m (y - x_*)^m = \sum_{m=0}^{\infty} \alpha a_m (y - x_*)^m,$$

$$\left[ \sum_{m=0}^{\infty} a_m (y - x_*)^m \right] \left[ \sum_{m=0}^{\infty} b_m (y - x_*)^m \right] = \sum_{n=0}^{\infty} \left[ \sum_{m=0}^n a_m b_{n-m} \right] (y - x_*)^n.$$

Cauchy's rule →

# Taylor Series

- Let  $f(y)$  be a real function,  $f(y) \in C^n [x_*, x_* + r)$ 
  - $n^{\text{th}}$  derivative  $f^{(n)}$  exists for  $x_* \leq y < x_* + r$

$$f(y) = f(x_*) + f'(x_*)(y - x_*) + \frac{1}{2!}f''(x_*)(y - x_*)^2 + \dots + \frac{1}{(n-1)!}f^{(n-1)}(x_*)(y - x_*)^{n-1} + \text{Residual}_n(y).$$

- Residual determines accuracy
- Two evaluations of remainder
  - Cauchy evaluation
  - Lagrange evaluation

$$|\text{Residual}_n(y)| \leq \frac{|y - x_*|^n}{n!} \sup_{x_* \leq y < x_* + r_*} |f^{(n)}(y)|.$$

$$\text{Residual}_n(y) = \int_{x_*}^y dx \int_{x_*}^x dx \dots \int_{x_*}^x f^{(n)}(x) dx = \frac{1}{n!} f^{(n)}(X) (y - x_*)^n,$$
$$X \in (x_*, x_* + r_*).$$

# Interpolation

- Weierstrass theorem

**Theorem:** (Weierstrass) For all  $f \in \mathcal{C}[a, b]$ , for all  $\epsilon > 0$ , there exists a degree  $n$  and a polynomial  $p_n$  such that  $\|f - p\|_\infty < \epsilon$ .

- Provides a guarantee that polynomials can interpolate any function
- On the other hand does not tell us how to choose the polynomial
- Also does not guarantee that the polynomial will actually “interpolate” the function ... only that it will be within  $\epsilon$ .
- Does not tell what the degree of the polynomial is

# Polynomial Facts

- A polynomial of degree  $k$  has at most  $k$  distinct zeroes, unless it is identically zero
- Sum of two polynomials of degree  $k$  is another polynomial of degree at most  $k$ .
- Polynomials can be expressed in many ways

For example  $(x - 2)(x - 5) = x^2 - 2x - 5(x - 2) = x^2 - 7x + 10$ .

We have used three different sets of **basis functions** in this example:

1.  $(x - 2)(x - 5)$ ,  $x - 1$ , and 1.
2.  $x^2$ ,  $x$ , and  $x - 2$ .
3.  $x^2$ ,  $x$ , and 1.
  - Degree of basis functions is 2, 1 and 0 ...
  - Basis 3 is the power basis or monomial basis
  - Any basis can be used ... often “orthogonal polynomials” are used

# Uniqueness of interpolant

- We know that the polynomial exists
- Suppose that there are two different polynomials that can interpolate the data
- Let them be  $p_{n-1}$  and  $q_{n-1}$  .
- So we have  $p_{n-1}(x_i) = y_i, i=1, \dots, n$   
 $q_{n-1}(x_i) = y_i, i=1, \dots, n$
- So  $p_{n-1}(x_i) - q_{n-1}(x_i) = 0, i=1, \dots, n$
- $p_{n-1} - q_{n-1}$  is the difference of two polynomials of degree  $n-1$ .
- It has  $n$  zeroes.
- Recall polynomial of degree  $k$  has at most  $k$  zeroes, or is the zero polynomial.
- Here we have more zeroes than degree ... so it is the zero polynomial
- So interpolant is unique



## Interpolating polynomials in power form

- Given  $n$  values of the function  $y_i$  at points  $x_i$
- Fit a polynomial  $P(x)$  that interpolates data at these points
- If we have  $n$  points to interpolate, then a polynomial of degree  $n-1$  can interpolate

$$P(x) = c_1 x^{n-1} + c_2 x^{n-2} + \dots + c_{n-1} x + c_n$$

- We can write the condition that it interpolate as a linear system

$$\begin{pmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \cdots & x_2 & 1 \\ \cdots & \cdots & \cdots & \cdots & 1 \\ x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

# Vandermonde matrices

**Example:** A **Vandermonde** matrix  $A$  is defined by a vector of elements  $x_1, \dots, x_n$ . Its first column is all ones. Each later column is the preceding one times this vector.

- Matlab code
- In matlab Vandermonde matrix is defined in flipped order using the function `vander`
- Example ...
- Also there is a function to fit polynomials to data, `polyfit`
- Vandermonde matrices are nonsingular if the points are distinct
- However they can be very poorly conditioned

```
n = length(x);  
V(:,1) = ones(n,1);  
for j=2:n,  
    V(:,j) = x.*V(:,j-1);  
end
```

# Computing interpolants by hand

- Suppose we have data at  $n$  points and we have fit a polynomial
- How can we do this fit efficiently?
- Fit for one point is  $y=y_1$
- Fit for two points can be written as  $y=a(x-x_1)+b(x-x_2)$
- Fit for three points can be written as

$$y=a(x-x_1)(x-x_2)+b(x-x_1)(x-x_3)+c(x-x_2)(x-x_3)$$

- And so on ...
- Advantage: each coefficient can be calculated independent of others
  - Why?
  - What is the form of the coefficient computed?