Math 340 • Fall 2026 • Victor Matveev

Polynomial Interpolation: Monomials, Lagrange polynomials, Newton's divided differences

Interpolation means "filling in" function values on an entire interval using only a select set of values within this interval. Unlike regression case that we considered previously, the residuals (errors) are exactly zero since the number of parameters (model components) equals the number of data points, which all have distinct x-values.

General task: Match select set of *N* values of a given function using another (simpler) function *P*(*x*), which is easier to compute than our original function $f(x)$:

$$y_k \equiv f(x_k) = P(x_k), \quad k = 1..N.$$

We require that all interpolation points are distinct: $x_k \neq x_j$ if $k \neq j$.

General approach: consider *P*(*x*) is the form of a linear combination of a conveniently chosen basis set of simple linearly independent functions $\{\phi_k(x)\}_{k=1}^{k=N}$:

$$f(x) \sim P(x) \equiv \sum_{k=1}^{N} c_k \phi_k(x).$$

Note: the number of data points *N* should match the number of basis functions (and basis function vectors evaluated at the x_k are linearly independent), otherwise unique solution isn't possible.

Matching the given *N* function values yields:

$$\begin{cases} P(x_1) = c_1\phi_1(x_1) + c_2\phi_2(x_1) + \ldots\ c_N\phi_N(x_1) = y_1, \\ P(x_2) = c_1\phi_1(x_2) + c_2\phi_2(x_2) + \ldots\ c_N\phi_N(x_2) = y_2, \\ \ldots \\ P(x_N) = c_1\phi_1(x_N) + c_2\phi_2(x_N) + \ldots\ c_N\phi_N(x_N) = y_N. \end{cases}$$

This is a linear algebra problem:

$$\begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \phi_3(x_1) & \ldots & \phi_N(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \phi_3(x_2) & \ldots & \phi_N(x_2) \\ \ldots & & & & \\ \phi_1(x_N) & \phi_2(x_N) & \phi_3(x_N) & \ldots & \phi_N(x_N) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}$$

Since this matrix is non-singular (given assumptions above), there is a unique interpolant for *f*(*x*): $P(x) \equiv \sum_{k=1}^{N} c_k \phi_k(x)$.

Interpolation is useful when the original function is costly to compute, while the basis functions are much simpler, so we can use them to approximate *f*(*x*) using only a couple of values of the latter function. It also helps when integration or other operations are needed, since it may be simpler to perform these operations on the chosen basis functions rather than the original (complicated) function.

Uniqueness of interpolant does **not** guarantee quality: interpolation quality depends on how well the property of the chosen functional basis matches the properties of the function being interpolated (a counter-example is the Runge function, $1/(1 + x^2)$ on an interval $[-5, 5]$, in which case interpolation points have to be selected carefully). However, we will mainly consider cases when polynomials provide an appropriate and convenient basis choice.

We will now consider three methods corresponding to three distinct polynomial bases

Method 1: Monomials: $\{\phi_k(x)\}_{k=1}^{k=N} = \{x^n\}_{n=0}^{n=N-1} \quad \Rightarrow \quad P(x) = \sum_{k=0}^{N-1} c_k x^k$

This corresponds to $f(x) \sim P(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \ldots + c_{N-1} x^{N-1}$.

Matching *N* function values leads to:

$$\underbrace{\begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{N-1} \\ \ldots & & & & \\ 1 & x_N & x_N^2 & \ldots & x_N^{N-1} \end{pmatrix}}_{\text{V = Van der Monde matrix}} \begin{pmatrix} c_0 \\ c_2 \\ \ldots \\ c_{N-1} \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}$$

The system has unique solution since $\det V = \prod_{j=1}^{N-1} \prod_{k=j+1}^{N} (x_k - x_j) = (x_N - x_1)\cdot(x_N - x_2)\cdot\ldots\cdot(x_2 - x_1) \neq 0.$

Example: Find a cubic interpolant for a function with the following values: $\begin{cases} x_1 = 0, & y_1 = f(0) = 0, \\ x_2 = 1, & y_2 = f(1) = 2, \\ x_3 = 2, & y_3 = f(2) = 2, \\ x_4 = 3, & y_4 = f(3) = 1. \end{cases}$

Van der Monde matrix: $\begin{pmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \\ 1 & x_4 & x_4^2 & x_4^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \end{pmatrix}$

Inverting the matrix always involves some work; let's use MATLAB:

```
>> V = [1 0 0 0; 1 1 1 1; 1 2 4 8; 1 3 9 27];
>> disp(inv(V));
    1.0000         0         0         0
   -1.8333    3.0000   -1.5000    0.3333
    1.0000   -2.5000    2.0000   -0.5000
   -0.1667    0.5000   -0.5000    0.1667
```

Therefore, the solution is: $\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = V^{-1} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -11/6 & 3 & -3/2 & 1/3 \\ 1 & -5/2 & 2 & -1/2 \\ -1/6 & 1/2 & -1/2 & 1/6 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 10/3 \\ -3/2 \\ 1/6 \end{pmatrix}$

Thus, we have: $f(x) \sim P(x) = \frac{10x}{3} - \frac{3x^2}{2} + \frac{x^3}{6} = \frac{x}{6}(20 - 9x + x^2) = \frac{x}{6}(x-4)(x-5).$

NOTE: If same points are used for multiple functions, the matrix inversion (the hard part) can be re-used!

Method 2: Lagrange's interpolation polynomials: $P(x)=\sum_{k=1}^{N} c_k L_k(x)$ where $L_k(x)=\dfrac{\prod_{m\neq k}(x-x_m)}{\prod_{m\neq k}(x_k-x_m)}$

Important property of this carefully chosen set is that $L_k(x_k)=1,\ L_k(x_{m\neq k})=0$.

This means that instead of the Van der Monde matrix, we get an identity matrix: how nice!

$$\begin{pmatrix} L_1(x_1) & L_2(x_1) & \dots & L_N(x_1) \\ L_1(x_2) & L_2(x_2) & \dots & L_N(x_2) \\ \dots & & & \\ L_1(x_N) & L_2(x_N) & \dots & L_N(x_N) \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_N \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \dots & & & & \\ 0 & 0 & 0 & \dots & 1 \end{pmatrix}}_{\text{Identity matrix!}}\begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix} \Rightarrow \begin{pmatrix} c_1 \\ c_2 \\ \dots \\ c_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{pmatrix}$$

Therefore, the interpolation polynomial is $f(x)\ \sim\ P(x)\ =\ y_1L_1(x)+y_2L_2(x)+\dots+y_NL_N(x).$

Example: Find a cubic interpolant for a function with the following values: $\begin{cases} x_1=0,\ \ y_1=f(0)=0, \\ x_2=1,\ \ y_2=f(1)=2, \\ x_3=2,\ y_3=f(2)=2, \\ x_4=3,\ y_4=f(3)=1. \end{cases}$

$$\begin{aligned} f(x)\ \sim P(x)\ &= y_1L_1(x)+y_2L_2(x)+y_3L_3(x)+y_4L_4(x) \\ &= 0\cdot L_1(x)+2\cdot L_2(x)+2\cdot L_3(x)+1\cdot L_4(x) \\ &= 2\frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)}+2\frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)}+1\frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} \\ &= 2\frac{x(x-2)(x-3)}{(1-2)(1-3)}+2\frac{x(x-1)(x-3)}{2(2-1)(2-3)}+\frac{x(x-1)(x-2)}{3(3-1)(3-2)} \\ &= x(x-2)(x-3)\ -x(x-1)(x-3)+\frac{1}{6}x(x-1)(x-2)\ =\ \frac{x}{6}\left(20-9x+x^2\right)=\ \frac{x}{6}(x-4)(x-5). \end{aligned}$$

NOTE: Naturally, the polynomial is the same as with method 1 on the preceding page (using monomials), but we did not need to solve a single equation!

Method 3: Newton's divided difference method -- use the following basis set:

$$P(x)=\sum_{k=1}^{N} c_k \psi_k(x) \text{ where } \begin{cases} \psi_1(x)=1, \quad \psi_2(x)=x-x_1, \quad \psi_3(x)=(x-x_1)(x-x_2), \\ \psi_k(x)=\prod_{m=1}^{k-1}(x-x_m)=(x-x_1)\cdot\ldots\cdot(x-x_{k-1}). \end{cases}$$

Important property of this carefully chosen set is that $\psi_k(x_{j<k})=0$.

This leads to a **lower-triangular** system, which is much easier to solve than the monomial method 1:

$$\begin{pmatrix} \psi_1(x_1) & 0 & 0 & \ldots & 0 \\ \psi_1(x_2) & \psi_2(x_2) & 0 & \ldots & 0 \\ \ldots & & & & \\ \psi_1(x_N) & \psi_2(x_N) & \psi_3(x_N) & \ldots & \psi_N(x_N) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_N \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \ldots & 0 \\ 1 & x_2-x_1 & 0 & \ldots & 0 \\ \ldots & & & & \\ 1 & x_N-x_1 & (x_N-x_1)(x_N-x_2) & \ldots & \psi_N(x_N) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \ldots \\ c_N \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{pmatrix}$$

The **coefficients** in this basis are called **Newton's divided differences**, with the following notation:

$$\left.\begin{aligned} & c_1 \equiv f[x_1]=y_1 \\ & c_2 \equiv f[x_1,x_2]=\frac{f[x_2]-f[x_1]}{x_2-x_1} \\ & c_3 \equiv f[x_1,x_2,x_3]=\frac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1} \\ & \ldots\ldots \end{aligned}\right\} \Rightarrow \text{Recurrent formula: } c_k=\frac{f[x_2,x_3,\ldots..,x_k]-f[x_1,x_2,\ldots..,x_{k-1}]}{x_k-x_1}$$

Example: Find a cubic interpolant for a function with the following values: $\begin{cases} x_1=0, & y_1=f(0)=0, \\ x_2=1, & y_2=f(1)=2, \\ x_3=2, & y_3=f(2)=2, \\ x_4=3, & y_4=f(3)=1. \end{cases}$

Apply the recurrent formula using table:

k	x_k	$f[x_k]$	$f[x_{k-1},x_k]$	$f[x_{k-2},x_{k-1},x_k]$	$f[x_1,x_2,x_3,x_4]$
1	0	0			
2	1	2	$\frac{2-0}{1-0}=2$		
3	2	2	$\frac{2-2}{2-1}=0$	$\frac{0-2}{2-0}=-1$	
4	3	1	$\frac{1-2}{3-2}=-1$	$\frac{-1-0}{3-1}=-\frac{1}{2}$	$\frac{-0.5-(-1)}{3-0}=\frac{1}{6}$

$$\begin{aligned} f(x) \sim P(x) &= f[x_1]+f[x_1,x_2](x-x_1)+f[x_1,x_2,x_3](x-x_1)(x-x_2)+f[x_1,x_2,x_3,x_4](x-x_1)(x-x_2)(x-x_3) \\ &= 0+2(x-0)-1(x-0)(x-1)+\frac{1}{6}(x-0)(x-1)(x-2) \\ &= x\left[2-(x-1)+\frac{1}{6}(x-1)(x-2)\right]=x\left[3-x+\frac{1}{6}(x-1)(x-2)\right] \\ &= \frac{x}{6}(20-9x+x^2)=\frac{x}{6}(x-4)(x-5). \end{aligned}$$

Naturally, the same result as with methods 1,2