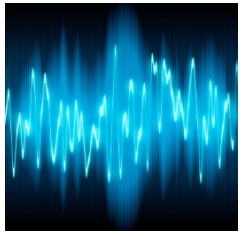
# Computational Linear Algebra: Polynomial Interpolation

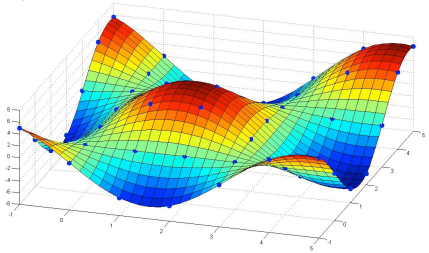
David Shuman

March 8, 2022

# Interpolation and Approximation

*Partial and/or noisy data due to sampling, experimentation, measurement error*

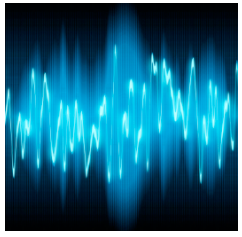


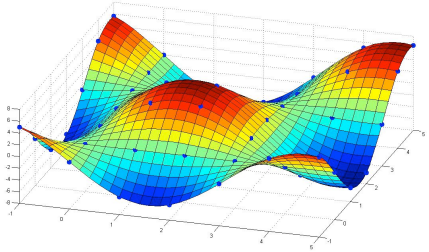


Source: MathWorks

# Interpolation and Approximation

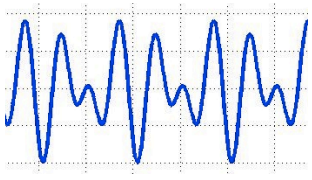
*Partial and/or noisy data due to sampling, experimentation, measurement error*

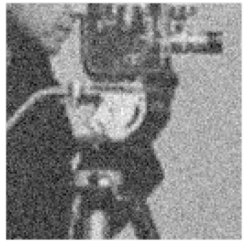


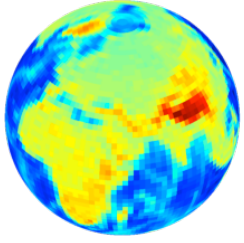


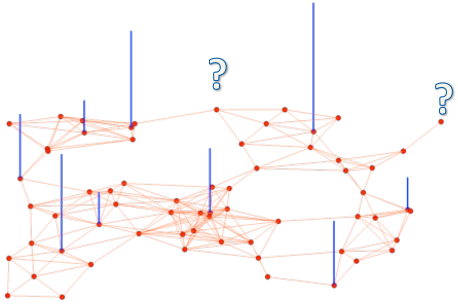
Source: MathWorks

*Different data domains*



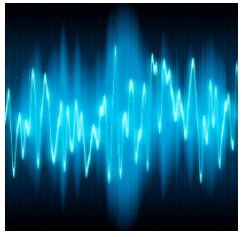


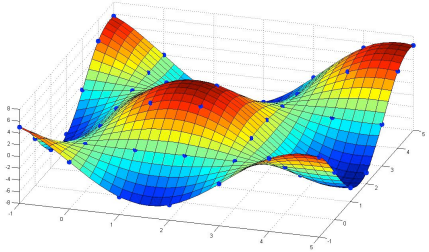




# Interpolation and Approximation

*Partial and/or noisy data due to sampling, experimentation, measurement error*



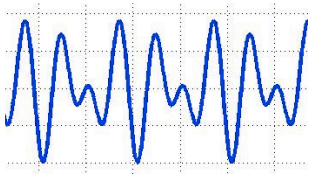


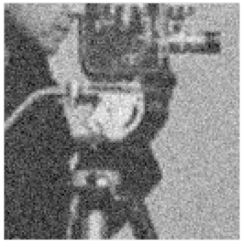
Source: MathWorks

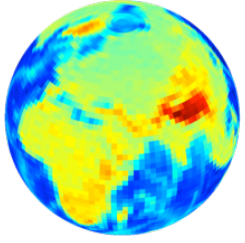
*Data processing and numerical analysis tasks*

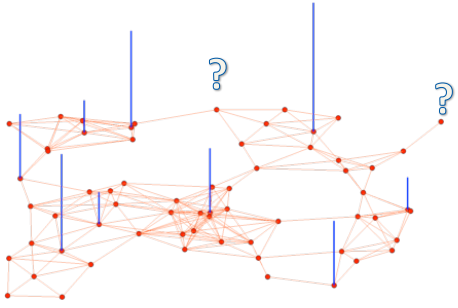
- Fill in missing data / make predictions
- Remove noise
- Compress data
- Explain observations
- Visualization
- Numerical integration
- Approximate a complicated function by a simple one
- …

*Different data domains*



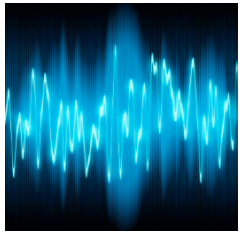


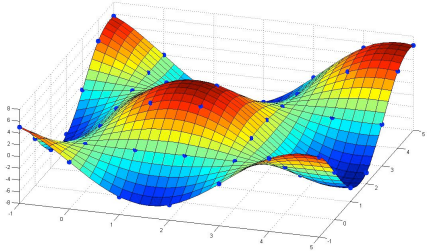




# Interpolation and Approximation

*Partial and/or noisy data due to sampling, experimentation, measurement error*



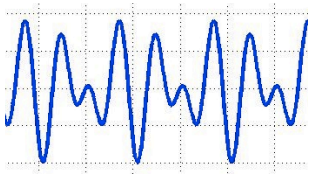


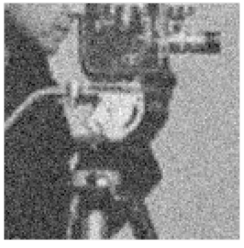
Source: MathWorks

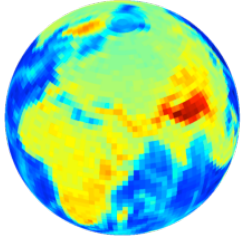
*Data processing and numerical analysis tasks*

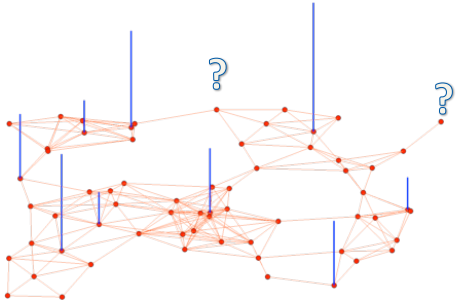
- Fill in missing data / make predictions
- Remove noise
- Compress data
- Explain observations
- Visualization
- Numerical integration
- Approximate a complicated function by a simple one
- …

*Different data domains*





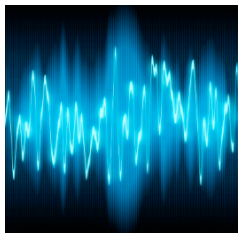


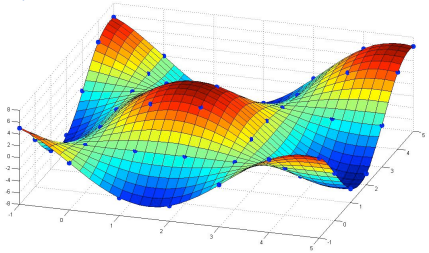


- We'll focus on 1D interpolation and approximation
  - Interpolation: Construct/estimate new data points within the range of a discrete set of known points
  - Approximation: Find best approximation to a given function or data set by a function from a predetermined class

# Interpolation and Approximation

*Partial and/or noisy data due to sampling, experimentation, measurement error*



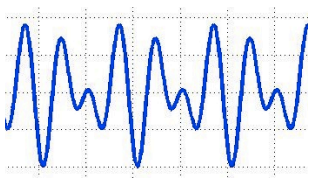


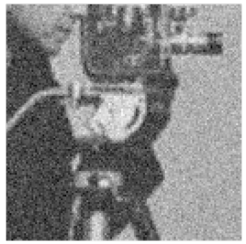
Source: MathWorks

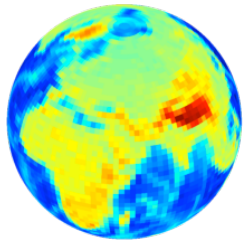
*Data processing and numerical analysis tasks*

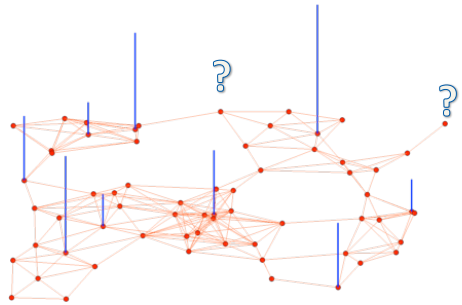
- Fill in missing data / make predictions
- Remove noise
- Compress data
- Explain observations
- Visualization
- Numerical integration
- Approximate a complicated function by a simple one
- …

*Different data domains*









- We'll focus on 1D interpolation and approximation
  - Interpolation: Construct/estimate new data points within the range of a discrete set of known points
  - Approximation: Find best approximation to a given function or data set by a function from a predetermined class
- Models and algorithms
  - Underlying all of these methods are models for relationships between data points (e.g., STAT 155, STAT 253, MATH 312, MATH 432, MATH 437, STAT 451, STAT 452, STAT 453, STAT 454)
  - Algorithms and numerical methods leverage those models and the structure of the data domain to perform data processing and numerical analysis tasks - our focus here

# Polynomial Interpolation

- Suppose that we are given $n$ data points

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

and we wish to find a polynomial of degree at most $n - 1$

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_{n-1} x^{n-1}$$

that passes through these points *exactly* — not a least squares fit

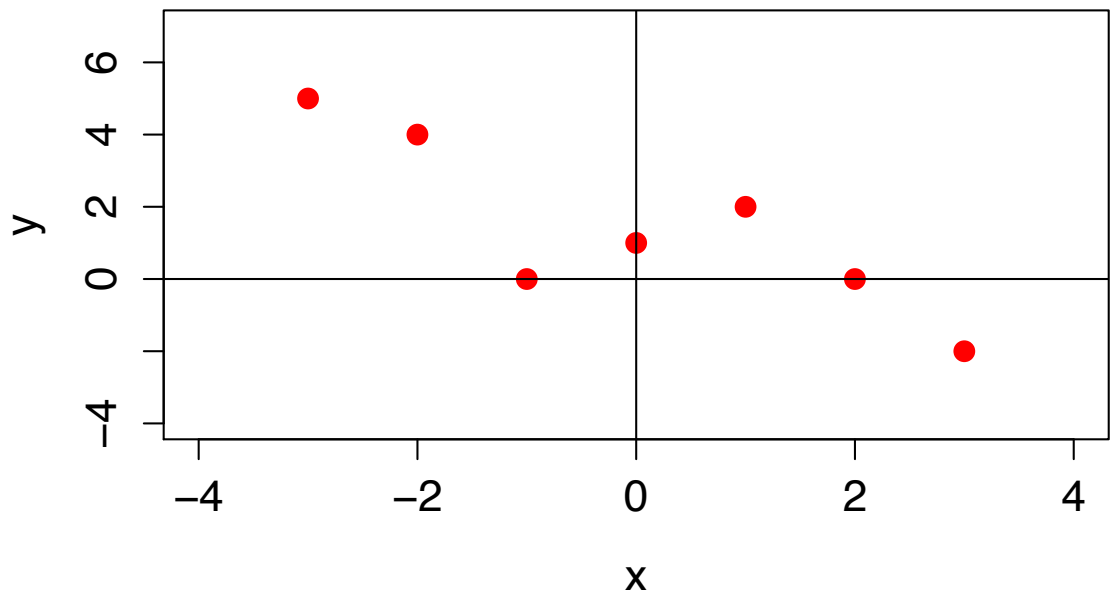
$$p(x_1) = y_1 \quad p(x_2) = y_2 \quad \cdots \quad p(x_n) = y_n$$

- This is called polynomial interpolation, and $p(x)$ is a polynomial interpolant

# Polynomial Interpolation

## Example

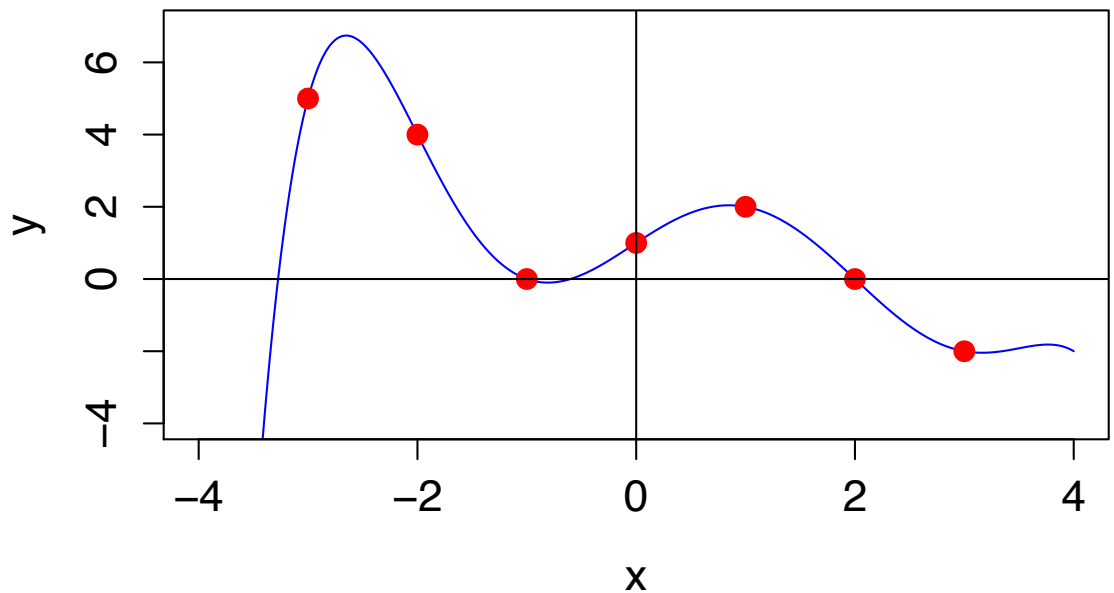
7 points



# Polynomial Interpolation

## Example

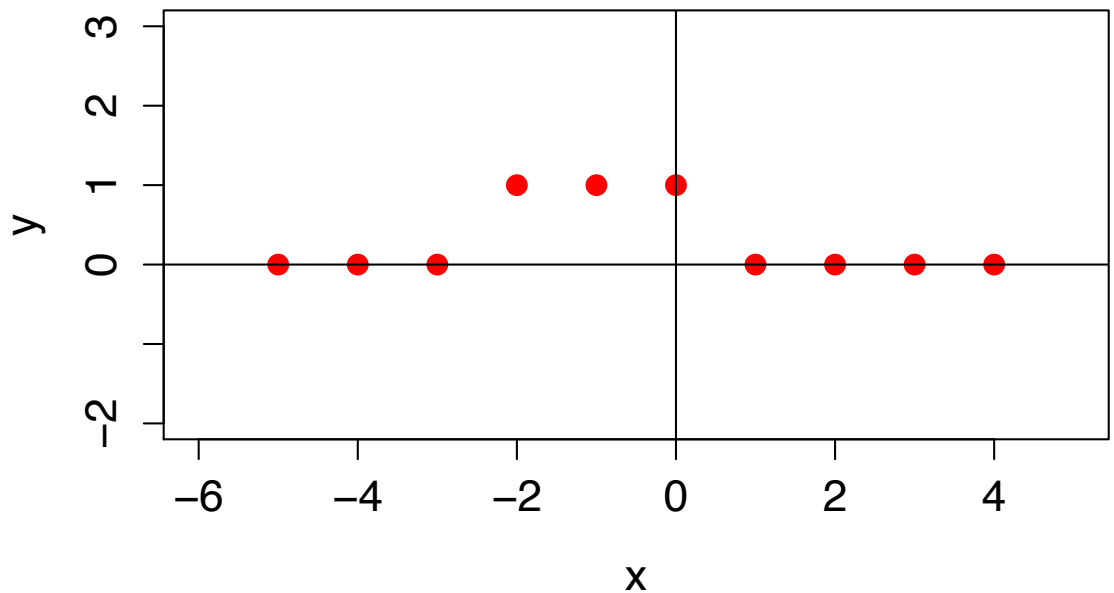
7 points



# Polynomial Interpolation

## Example

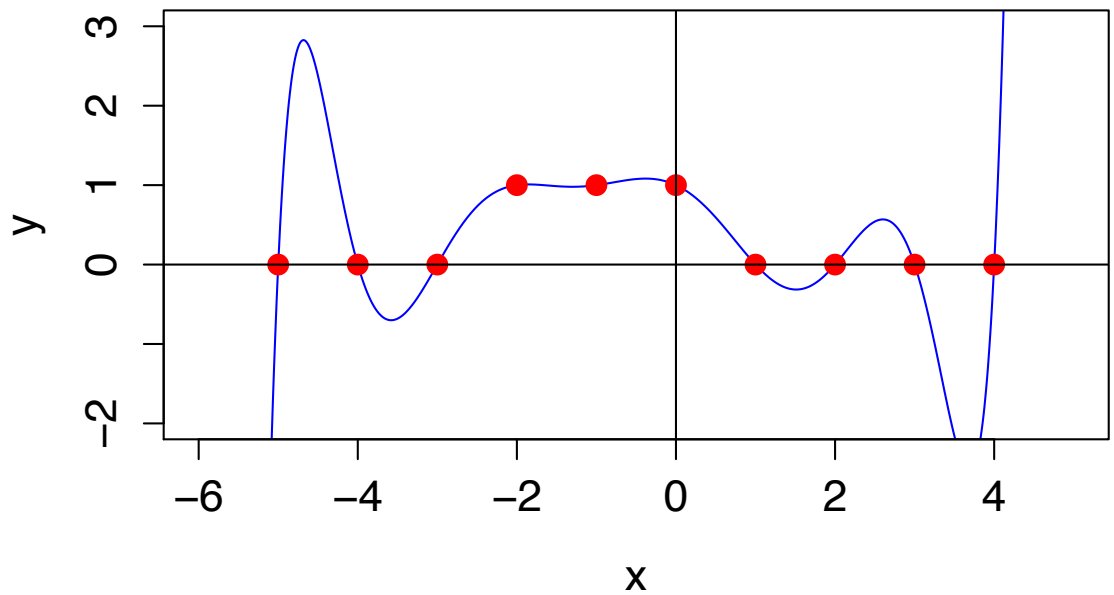
10 points



# Polynomial Interpolation

## Example

10 points



# Polynomial Interpolation

- Why polynomials?
  - Efficiently computed with additions and multiplications (recall Horner's Method)
- Does an interpolating polynomial of degree $n-1$ always exist?
- If so, is it unique?

# I. Vandermonde Interpolation

The relationship of data points $x_1, , \ldots, x_n$ and $y_1, \ldots, y_n$ to the coefficients $c_1, \ldots, c_n$ in the polynomial

$$p(x) = c_0 + c_1 x + \cdots + c_{n-1} x^{n-1}$$

is given by the Vandermonde matrix:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & & x_3^{n-1} \\ & \vdots & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

- Its determinant is $\det(V) = \prod_{i<j} (x_j - x_i)$
- Example: $n = 3 \Longrightarrow \det(V) = (x_2 - x_1)(x_3 - x_1)(x_3 - x_2)$
- If the $x_i$'s are distinct, the determinant is nonzero, the matrix is invertible, and there is a unique solution!

# I. Vandermonde Interpolation

$$\begin{bmatrix} 1 & x_1 & x_1^2 & & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & & x_3^{n-1} \\ & \vdots & & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

## Theorem (Polynomial Interpolation)

*If $x_1, x_2, \ldots, x_n$ are distinct, then there is a unique polynomial $p(x)$ of degree $n-1$ or less which interpolates the points $(x_1, y_1)$, $(x_2, y_2)$, $\ldots$, $(x_n, y_n)$.*

- Proof comes from the Vandermonde matrix!
- Vandermonde is ill-conditioned

# Basis Functions for Interpolation: Monomials

- The monomials $1, x, x^2, \ldots, x^{n-1}$ are a basis for polynomials of degree $n-1$
  - Any such function can be written as the linear combination

$$p_{n-1}(x) = \sum_{i=1}^{n} a_i g_i(x), \text{ where } g_i(x) = x^{i-1} \text{ for } i = 1, 2, \ldots, n$$

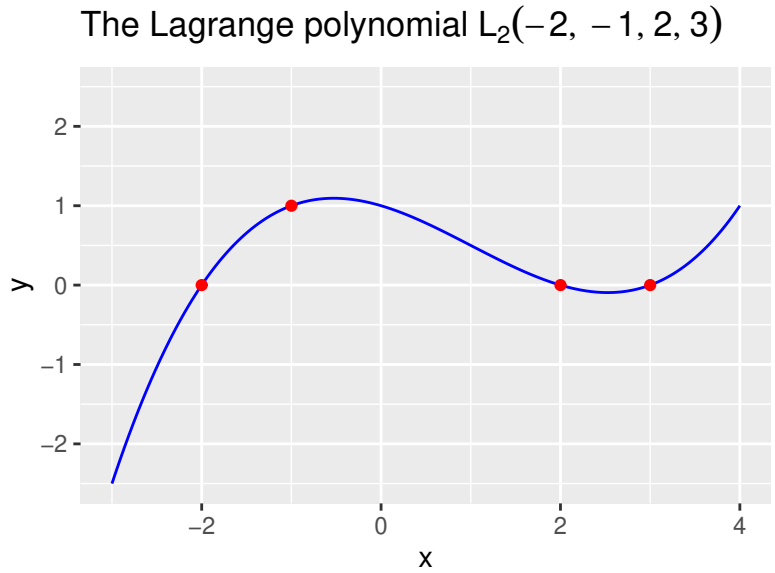
- Interpolation: Find $\{a_i\}_{i=1,2,\ldots,n}$ such that $p_{n-1}(x_j) = y_j$ for all $j$
- The columns of the Vandermonde matrix are the basis functions evaluated at the different interpolation points ($V_{kl} = g_l(x_k)$)

# Basis Functions for Interpolation: Lagrange Polynomials

- Another set of basis functions are the Lagrange polynomials (also called fundamental polynomials or cardinal functions):

$$L_i(x) = \prod_{j \neq i} \left( \frac{x - x_j}{x_i - x_j} \right), \quad i = 1, 2, \ldots, n$$

- Each basis function $L_i(x)$ is a polynomial of degree $n - 1$ that is equal to 1 at $x = x_i$ and equal to 0 at $x = x_j$ for all other interpolation points $x_j$ (besides $x_i$):




- With this choice of basis functions, the coefficients are easy to identify ($a_i = y_i$), but the interpolating polynomial is hard to compute

# II. Lagrange Interpolation

For points $(1, 10), (2, 6), (3, 4), (4, 10)$, consider the polynomial

$$\begin{aligned} p(x) =& 10\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} + 6\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} \\ &+ 4\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} + 10\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} \end{aligned}$$

$$\begin{aligned} p(x) &= \frac{-10}{6}(x^3 - 9x^2 + 26x - 24) + \frac{6}{2}(x^3 - 8x^2 + 19x - 12) \\ &\quad + \frac{4}{-2}(x^3 - 7x^2 + 14x - 8) + \frac{10}{6}(x^3 - 6x^2 + 11x - 6) \\ &= x^3 - 5x^2 + 4x + 10. \end{aligned}$$

In general,

$$p(x) = \sum_{i=1}^{n} y_i \prod_{j \neq i} \frac{(x - x_j)}{(x_i - x_j)} \qquad \text{(i.e., } a_i = y_i\text{)}$$

Note: this is *a* polynomial through the given points, and by uniqueness it is *the* interpolating polynomial.

# II. Lagrange Interpolation: Barycentric Formula

An equivalent, but more numerically efficient and stable way to write the Lagrange interpolant is the following, which is referred to as the (first) barycentric interpolation formula (see Ch. 5 of Trefethen for details):

$$p(x) = \frac{\sum_{i=1}^{n} \frac{\lambda_i y_i}{x - x_i}}{\sum_{i=1}^{n} \frac{\lambda_i}{x - x_i}},$$

where (i) each $\lambda_i$ is a constant given by $\lambda_i = \frac{1}{\prod_{j \neq i}(x_i - x_j)}$, and (ii) as special cases, $p(x) = y_i$ if $x = x_i$.

# III. Newton's Divided Differences (Algorithmically)

For points $(1, 10), (2, 6), (3, 4), (4, 10)$, consider the recursive triangle of divided differences

$$\begin{array}{c|cccc} 1 & 10 & & & \\ & & -4 & & \\ 2 & 6 & & 1 & \\ & & -2 & & 1 \\ 3 & 4 & & 4 & \\ & & 6 & & \\ 4 & 10 & & & \end{array}$$

and the polynomial

$$\begin{aligned} p(x) &= 10(1) + (-4)(x-1) + (1)(x-1)(x-2) + (1)(x-1)(x-2)(x-3) \\ &= 10 + (x-1)\Big[(-4) + (x-2)\Big[1 + (x-3)[1]\Big]\Big] \\ &= x^3 - 5x^2 + 4x + 10 \end{aligned}$$

(Recall Horner's method of polynomial evaluation)

# III. Newton's Divided Differences (Conceptually)

- Recall:

$$p(x) = 10(1)+(-4)(x-1)+(1)(x-1)(x-2)+(1)(x-1)(x-2)(x-3)$$

- A new set of basis functions:

$$\pi_i(x) = \begin{cases} 1, & i = 1 \\ (x - x_1)(x - x_2)\cdots(x - x_{i-1}), & i = 2, 3, \ldots, n \end{cases}$$

- Write the interpolation problem as $Ax = b$ again:

$$\begin{bmatrix} \pi_1(x_1) & \pi_2(x_1) & \pi_3(x_1) & \ldots & \pi_n(x_1) \\ \pi_1(x_2) & \pi_2(x_2) & \pi_3(x_2) & \ldots & \pi_n(x_2) \\ \pi_1(x_3) & \pi_2(x_3) & \pi_3(x_3) & \ldots & \pi_n(x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \pi_1(x_n) & \pi_2(x_n) & \pi_3(x_n) & \ldots & \pi_n(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

- But, $i > j$ implies $\pi_i(x_j) = 0$ !!!!!

# III. Newton's Divided Differences (Conceptually)

$$\begin{bmatrix} \pi_1(x_1) & \pi_2(x_1) & \pi_3(x_1) & \dots & \pi_n(x_1) \\ \pi_1(x_2) & \pi_2(x_2) & \pi_3(x_2) & \dots & \pi_n(x_2) \\ \pi_1(x_3) & \pi_2(x_3) & \pi_3(x_3) & \dots & \pi_n(x_3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \pi_1(x_n) & \pi_2(x_n) & \pi_3(x_n) & \dots & \pi_n(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

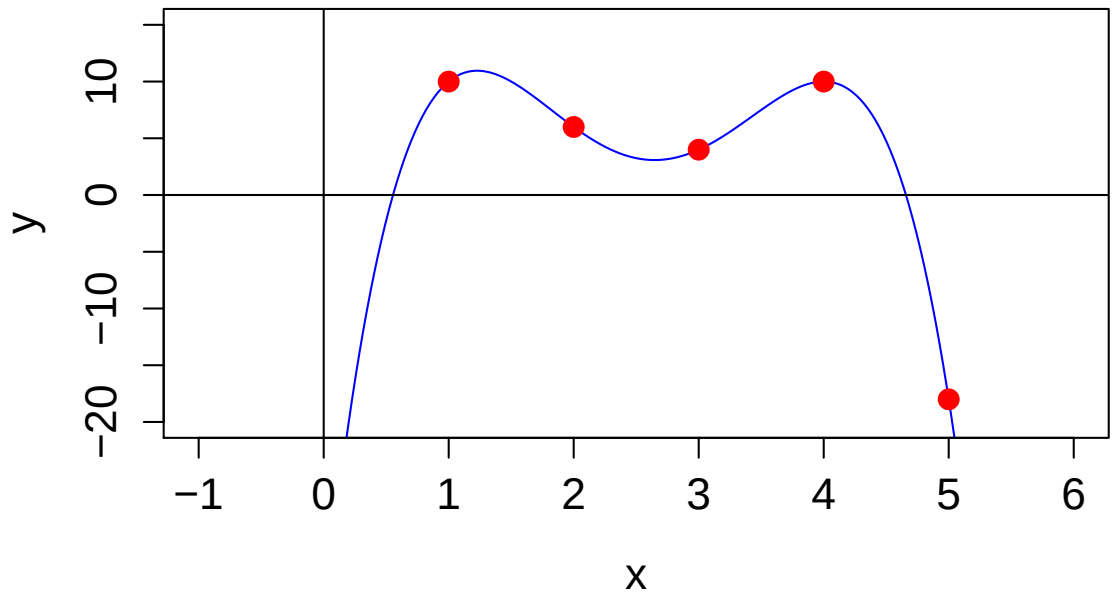
becomes

$$\begin{bmatrix} \pi_1(x_1) & 0 & 0 & \dots & 0 \\ \pi_1(x_2) & \pi_2(x_2) & 0 & \dots & 0 \\ \pi_1(x_3) & \pi_2(x_3) & \pi_3(x_3) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \pi_1(x_n) & \pi_2(x_n) & \pi_3(x_n) & \dots & \pi_n(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

- This is the forward substitution problem we have come to know and love

# Add one more point

We interpolated $(1, 10), (2, 6), (3, 4), (4, 10)$ and we add $(5, -18)$



# Add one more point

We interpolated $(1, 10), (2, 6), (3, 4), (4, 10)$ and we add $(5, -18)$

Lagrange:

$$p(x) = 10\frac{(x-2)(x-3)(x-4)(x-5)}{(1-2)(1-3)(1-4)(1-5)} + 6\frac{(x-1)(x-3)(x-4)(x-5)}{(2-1)(2-3)(2-4)(2-5)}$$
$$+ 4\frac{(x-1)(x-2)(x-4)(x-5)}{(3-1)(3-2)(3-4)(3-5)} + 10\frac{(x-1)(x-2)(x-3)(x-5)}{(4-1)(4-2)(4-3)(4-5)}$$
$$-18\ \frac{(x-1)(x-2)(x-3)(x-4)}{(5-1)(5-2)(5-3)(5-4)}$$

# Add one more point

We interpolated $(1, 10), (2, 6), (3, 4), (4, 10)$ and we add $(5, -18)$

| | | | | | |
|---|---|---|---|---|---|
| 1 | 10 | | | | |
| | | $-4$ | | | |
| 2 | 6 | | 1 | | |
| | | $-2$ | | 1 | |
| 3 | 4 | | 4 | | $-2$ |
| | | 6 | | $-7$ | |
| 4 | 10 | | $-17$ | | |
| | | $-28$ | | | |
| 5 | $-18$ | | | | |

$$p(x) = 10(1) + (-4)(x-1) + (1)(x-1)(x-2)$$
$$+ (1)(x-1)(x-2)(x-3) + (-2)(x-1)(x-2)(x-3)(x-4)$$
$$= 10 + (x-1)\Big[(-4) + (x-2)\Big[1 + (x-3)[1 + (x-4)[-2]]\Big]\Big]$$

So it is easier to add points with Newton's divided difference method

# One Way to Compress A Function

- Suppose that we are interested in computing with a function like $f(x) = \sin(x)$ on an interval $[a, b]$

- We could do the following:
  - sample some points in the interval $[a, b]$
  - find the polynomial $p(x)$ that interprets through these points
  - plot $f(x)$ and $p(x)$ together to see how we did
  - find the error between them on the interval $[a, b]$

- This is a form of compression: we only need to store the coefficients of the interpolating polynomial instead of a gigantic table of sin values

# Function Approximation

$f(x) = \sin(x)$: few interpolating points yields a good approximation