15. Lagrange interpolation

Last time

- Absolute and relative errors
- Condition number of a problem
- Stability of an algorithm

Goals for today

- Interpolation
- Piecewise interpolation
- Global Lagrange interpolation

Goals for today

- Interpolation
- Piecewise interpolation
- Global Lagrange interpolation

- Heading towards **global function approximation**

Representing data

- Suppose we have some **discrete data**
- E.g. measurements of a physical or economic system

Representing data

- Suppose we have some **discrete data**
- E.g. measurements of a physical or economic system

- These could be discrete **samples** coming from a system with *continuous* output

Representing data

- Suppose we have some **discrete data**
- E.g. measurements of a physical or economic system

- These could be discrete **samples** coming from a system with *continuous* output
- How can we **re-construct** a function from a set of discrete samples?

Re-constructing functions from data

- (At least) two different methods to re-construct functions:
  1. Fit a “best approximation” to the data from within some class of functions: **approximation theory**
  2. A function that *passes through* the data: **interpolation**

Re-constructing functions from data

- (At least) two different methods to re-construct functions:
  1. Fit a “best approximation” to the data from within some class of functions: **approximation theory**
  2. A function that *passes through* the data: **interpolation**
- If in fact the data came from sampling a function, we can compare the original and re-constructed functions

Collaboration I

Interpolation

Suppose we are given data (x_i, y_i) in 2D for $i = 0, \dots, n$ and we want to find a function $f(x)$ that passes through the points.

1. Which type of functions f should we try?
2. Write down the conditions on f that we want to satisfy.
3. What kind of mathematical problem do you get?
4. Is this problem solvable?

Interpolation

- Given data points (x_i, y_i) for $i = 0, \ldots, n$
- We want to find a function $f(x)$ passing through the data:

$$f(x_i) = y_i \quad \forall i$$

Interpolation

- Given data points (x_i, y_i) for $i = 0, \dots, n$
- We want to find a function $f(x)$ passing through the data:

$$f(x_i) = y_i \quad \forall i$$

- The x_i are **nodes** or **knots**
- We will assume that they are distinct and ordered:

$$a = x_0 < x_1 < \cdots < x_n = b$$

Polynomial interpolation

- The first class of functions f to try are **polynomials**

Polynomial interpolation

- The first class of functions f to try are **polynomials**

- Suppose $f(x) = a_0 + a_1 x + \cdots + a_n x^n$
- Then $f(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i \quad \forall i$

Polynomial interpolation

- The first class of functions f to try are **polynomials**

- Suppose $f(x) = a_0 + a_1 x + \cdots + a_n x^n$
- Then $f(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i \quad \forall i$

- Find the a_i that solve this system of equations

Polynomial interpolation

- The first class of functions f to try are **polynomials**

- Suppose $f(x) = a_0 + a_1 x + \cdots + a_n x^n$
- Then $f(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i \quad \forall i$

- Find the a_i that solve this system of equations
- What kind of system is it?

Polynomial interpolation

- The first class of functions f to try are **polynomials**

- Suppose $f(x) = a_0 + a_1 x + \cdots + a_n x^n$
- Then $f(x_i) = a_0 + a_1 x_i + \cdots + a_n x_i^n = y_i \quad \forall i$

- Find the a_i that solve this system of equations
- What kind of system is it?
- When can we expect to be able to solve it?

Polynomial interpolation II

- Each equation can be written

$$\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}^{\mathsf{T}} \mathbf{a} = y_i$$

- $\mathbf{a} = (a_1, a_2, \dots, a_n)^{\mathsf{T}}$ is a vector of the unknown a_is

Polynomial interpolation II

- Each equation can be written

$$\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}^{\mathsf{T}} \mathbf{a} = y_i$$

- $\mathbf{a} = (a_1, a_2, \ldots, a_n)^{\mathsf{T}}$ is a vector of the unknown a_is

- Hence we get an equation of form $\mathsf{V} \cdot \mathbf{a} = \mathbf{y}$

Polynomial interpolation II

- Each equation can be written

$$\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}^{\mathsf{T}} \mathbf{a} = y_i$$

- $\mathbf{a} = (a_1, a_2, \ldots, a_n)^{\mathsf{T}}$ is a vector of the unknown a_is

- Hence we get an equation of form $\mathsf{V} \cdot \mathbf{a} = \mathbf{y}$
- V is **Vandermonde matrix** with ith row $\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}$

Polynomial interpolation II

- Each equation can be written

$$\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}^{\mathsf{T}} \mathbf{a} = y_i$$

- $\mathbf{a} = (a_1, a_2, \ldots, a_n)^{\mathsf{T}}$ is a vector of the unknown a_is

- Hence we get an equation of form $\mathsf{V} \cdot \mathbf{a} = \mathbf{y}$
- V is **Vandermonde matrix** with ith row $\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}$
- **In principle** we can solve polynomial interpolation by solving this linear system

Polynomial interpolation II

- Each equation can be written

$$\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}^{\mathsf{T}} \mathbf{a} = y_i$$

- $\mathbf{a} = (a_1, a_2, \ldots, a_n)^{\mathsf{T}}$ is a vector of the unknown a_is

- Hence we get an equation of form $\mathsf{V} \cdot \mathbf{a} = \mathbf{y}$
- V is **Vandermonde matrix** with ith row $\begin{pmatrix} 1 & x_i & x_i^2 & \cdots & x_i^n \end{pmatrix}$
- **In principle** we can solve polynomial interpolation by solving this linear system
- V is ill-conditioned; algorithm is expensive and unstable

Lagrange interpolation

- Alternative method for polynomial interpolation

Lagrange interpolation

- Alternative method for polynomial interpolation
- We will build the polynomial interpolant as a sum

Lagrange interpolation

- Alternative method for polynomial interpolation
- We will build the polynomial interpolant as a sum

- Look for **cardinal basis** functions:
- Functions ℓ_k that are 1 at a single node and 0 elsewhere:

$$\ell_k(x_i) = \delta_{i,k} = [i = k]$$

Lagrange interpolation

- Alternative method for polynomial interpolation
- We will build the polynomial interpolant as a sum

- Look for **cardinal basis** functions:
- Functions ℓ_k that are 1 at a single node and 0 elsewhere:

$$\ell_k(x_i) = \delta_{i,k} = [i = k]$$

- **Iverson bracket notation** (indicator function):

$$[\mathcal{S}] = \begin{cases} 1, & \text{if statement } \mathcal{S} \text{ is correct} \\ 0, & \text{if not} \end{cases}$$

Collaboration II

Line joining two points

Given two points (x_0, y_0) and (x_1, y_1).

1. What degree polynomial interpolates them?
2. Find cardinal basis functions.
3. Hence find the polynomial interpolating (x_0, y_0) and (x_1, y_1).

Two points

- Simplest case: Find Line joining (x_0, y_0) and (x_1, y_1)
- We need a degree-1 polynomial ℓ_1 such that $\ell_1(x_1) = 1$ and $\ell_1(x_2) = 0$

Two points

- Simplest case: Find Line joining (x_0, y_0) and (x_1, y_1)
- We need a degree-1 polynomial ℓ_1 such that $\ell_1(x_1) = 1$ and $\ell_1(x_2) = 0$
- We could use $\ell_1(x) = ax + b$ and substitute
- Instead, since x_2 is a root, $p(x) = c(x - x_2)$

Two points

- Simplest case: Find Line joining (x_0, y_0) and (x_1, y_1)
- We need a degree-1 polynomial ℓ_1 such that $\ell_1(x_1) = 1$ and $\ell_1(x_2) = 0$

- We could use $\ell_1(x) = ax + b$ and substitute
- Instead, since x_2 is a root, $p(x) = c(x - x_2)$

- So $p(x_1) = c(x_1 - x_2) = 1$
- Hence $c = \frac{1}{x - x_1}$, giving $\ell_1(x) = \frac{x - x_2}{x_1 - x_2}$
- Symmetry gives ℓ_2

Lagrange interpolant

- The **Lagrange interpolant** or **Lagrange polynomial** satisfies

$$L(x_1) = y_1 \quad \text{and} \quad L(x_2) = y_2$$

- Since ℓ_1 and ℓ_2 are cardinal basis functions,

$$L(x) = y_1 \ell_1(x) + y_2 \ell_2(x)$$

Lagrange interpolant

- The **Lagrange interpolant** or **Lagrange polynomial** satisfies

$$L(x_1) = y_1 \quad \text{and} \quad L(x_2) = y_2$$

- Since ℓ_1 and ℓ_2 are cardinal basis functions,

$$L(x) = y_1\ell_1(x) + y_2\ell_2(x)$$

- *Any* linear polynomial $ax + b$ can be written in this way

Lagrange interpolant

- The **Lagrange interpolant** or **Lagrange polynomial** satisfies

$$L(x_1) = y_1 \quad \text{and} \quad L(x_2) = y_2$$

- Since ℓ_1 and ℓ_2 are cardinal basis functions,

$$L(x) = y_1\ell_1(x) + y_2\ell_2(x)$$

- *Any* linear polynomial $ax + b$ can be written in this way
- Hence $\{\ell_1, \ell_2\}$ forms a new **basis** of linear polynomials

Piecewise-linear interpolation

- A possible interpolant would be to use separate polynomials on each sub-interval $[x_i, x_{i+1}]$

Piecewise-linear interpolation

- A possible interpolant would be to use separate polynomials on each sub-interval $[x_i, x_{i+1}]$
- E.g. a piecewise-linear function satisfying cardinality conditions:

Piecewise-linear interpolation

- A possible interpolant would be to use separate polynomials on each sub-interval $[x_i, x_{i+1}]$
- E.g. a piecewise-linear function satisfying cardinality conditions:
- Piecewise-linear "hat" function with value 1 at x_k and zero for all other x_i
- Then a piecewise-linear interpolant ("join the dots") is a linear combination of hat **basis functions**

Piecewise-linear interpolation

- A possible interpolant would be to use separate polynomials on each sub-interval $[x_i, x_{i+1}]$
- E.g. a piecewise-linear function satisfying cardinality conditions:
- Piecewise-linear “hat” function with value 1 at x_k and zero for all other x_i
- Then a piecewise-linear interpolant (“join the dots”) is a linear combination of hat **basis functions**
- *Any* piecewise-linear function can be written like this

Piecewise-linear interpolation

- A possible interpolant would be to use separate polynomials on each sub-interval $[x_i, x_{i+1}]$
- E.g. a piecewise-linear function satisfying cardinality conditions:
- Piecewise-linear "hat" function with value 1 at x_k and zero for all other x_i
- Then a piecewise-linear interpolant ("join the dots") is a linear combination of hat **basis functions**
- *Any* piecewise-linear function can be written like this
- This points towards **finite-element** methods for solving differential equations

Piecewise polynomial interpolation

- Piecewise-linear interpolation gives a *non-smooth* result

Piecewise polynomial interpolation

- Piecewise-linear interpolation gives a *non-smooth* result
- We can make the result smoother by replacing linear pieces with higher-degree polynomials

Piecewise polynomial interpolation

- Piecewise-linear interpolation gives a *non-smooth* result
- We can make the result smoother by replacing linear pieces with higher-degree polynomials
- This gives **splines**, e.g. cubic splines

Piecewise polynomial interpolation

- Piecewise-linear interpolation gives a *non-smooth* result
- We can make the result smoother by replacing linear pieces with higher-degree polynomials

- This gives **splines**, e.g. cubic splines
- Two continuous derivatives

Piecewise polynomial interpolation

- Piecewise-linear interpolation gives a *non-smooth* result
- We can make the result smoother by replacing linear pieces with higher-degree polynomials

- This gives **splines**, e.g. cubic splines
- Two continuous derivatives

- Impose conditions at each node
- Requires solving a linear system

Global Lagrange interpolation in n points

- Instead we can look for a *single* polynomial that interpolates *all* $n+1$ points x_i *simultaneously*

Global Lagrange interpolation in n points

- Instead we can look for a *single* polynomial that interpolates *all* $n+1$ points x_i *simultaneously*
- By the Vandermonde argument we know this is possible

Global Lagrange interpolation in n points

- Instead we can look for a *single* polynomial that interpolates *all* $n+1$ points x_i *simultaneously*
- By the Vandermonde argument we know this is possible
- We can solve this by extending the argument from 2 points

Global Lagrange interpolation in n points

- Instead we can look for a *single* polynomial that interpolates *all* $n + 1$ points x_i *simultaneously*
- By the Vandermonde argument we know this is possible
- We can solve this by extending the argument from 2 points
- Construct a cardinal basis

Collaboration III

Polynomial interpolation

1. How can we make a cardinal basis function that is 1 at x_k and 0 at all other x_j?
2. How can we then make an interpolant of the data?

Global Lagrange interpolation II

- Generalise from 2 to n points:

$$\ell_k(x) = c_k(x - x_1)\cdots\widehat{(x - x_k)}\cdots(x - x_n)$$

where $\hat{\cdot}$ indicates a *missing* term

- We want $\ell_k(x_k) = 1$, so

$$c_k = \frac{1}{(x_k - x_1)\cdots\widehat{(x_k - x_k)}\cdots(x - x_n)}$$

Global Lagrange interpolation II

- Generalise from 2 to n points:

$$\ell_k(x) = c_k(x - x_1) \cdots \widehat{(x - x_k)} \cdots (x - x_n)$$

where $\hat{\cdot}$ indicates a *missing* term

- We want $\ell_k(x_k) = 1$, so

$$c_k = \frac{1}{(x_k - x_1) \cdots \widehat{(x_k - x_k)} \cdots (x - x_n)}$$

- Thus $\ell_k(x) = \prod_{i \neq k} \frac{x - x_i}{x_k - x_i}$
- And $L(x) = \sum_{k=0}^{n} y_k \ell_k(x)$

Global Lagrange interpolation III

- Uniqueness: Suppose they are not unique and subtract

Global Lagrange interpolation III

- Uniqueness: Suppose they are not unique and subtract
- We have constructed a *new basis* for the (vector) space of degree-n polynomials

Global Lagrange interpolation III

- Uniqueness: Suppose they are not unique and subtract
- We have constructed a *new basis* for the (vector) space of degree-n polynomials
- What can go wrong?

Global Lagrange interpolation III

- Uniqueness: Suppose they are not unique and subtract
- We have constructed a *new basis* for the (vector) space of degree-n polynomials

- What can go wrong?
- We will see that global Lagrange interpolation can go *very badly wrong* if we use equally-spaced points

Global Lagrange interpolation III

- Uniqueness: Suppose they are not unique and subtract
- We have constructed a *new basis* for the (vector) space of degree-n polynomials

- What can go wrong?
- We will see that global Lagrange interpolation can go *very badly wrong* if we use equally-spaced points
- It turns out to be much better to use points that *cluster* near the endpoints of interval

Summary of Lagrange interpolation

- Given data $(n+1)$ data points $(x_i, y_i)_{i=0}^{n}$, the Lagrange interpolant of degree n is

$$p(x) = \sum_{j=0}^{n} y_j \ell_j(x)$$

- Where

$$\ell_j(x) := \frac{\prod_{k \neq j}(x - x_k)}{\prod_{k \neq j}(x_j - x_k)}$$

Issues with Lagrange interpolation

- Evaluating this requires $\mathcal{O}(n^2)$ operations

Issues with Lagrange interpolation

- Evaluating this requires $\mathcal{O}(n^2)$ operations
- If we add a new node, we must recalculate

Issues with Lagrange interpolation

- Evaluating this requires $\mathcal{O}(n^2)$ operations
- If we add a new node, we must recalculate
- It also turns out to be numerically unstable

Issues with Lagrange interpolation

- Evaluating this requires $\mathcal{O}(n^2)$ operations
- If we add a new node, we must recalculate
- It also turns out to be numerically unstable

- These issues can be solved by reformulating it into **barycentric** Lagrange interpolation

Barycentric Lagrange interpolation

- Let's define the product

$$\ell(x) := (x - x_0)(x - x_1) \cdots (x - x_n)$$

Barycentric Lagrange interpolation

- Let's define the product

$$\ell(x) := (x - x_0)(x - x_1)\cdots(x - x_n)$$

- Define the **barycentric weights**

$$w_j := \frac{1}{\prod_{k \neq j}(x_j - x_k)}$$

Barycentric Lagrange interpolation

- Let's define the product

$$\ell(x) := (x - x_0)(x - x_1)\cdots(x - x_n)$$

- Define the **barycentric weights**

$$w_j := \frac{1}{\prod_{k \neq j}(x_j - x_k)}$$

- Then $\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}$

Barycentric Lagrange interpolation

- Let's define the product

$$\ell(x) := (x - x_0)(x - x_1)\cdots(x - x_n)$$

- Define the **barycentric weights**

$$w_j := \frac{1}{\prod_{k \neq j}(x_j - x_k)}$$

- Then $\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}$
- So $p(x) = \ell(x)\sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j$

Barycentric Lagrange interpolation

- Let's define the product

$$\ell(x) := (x - x_0)(x - x_1) \cdots (x - x_n)$$

- Define the **barycentric weights**

$$w_j := \frac{1}{\prod_{k \neq j}(x_j - x_k)}$$

- Then $\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}$
- So $p(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j} y_j$
- Also $w_j = 1/\ell'(x_j)$

Barycentric Lagrange interpolation II

- Now suppose we interpolate the constant function
  $\mathbf{1}(x) := 1 \quad \forall x$

Barycentric Lagrange interpolation II

- Now suppose we interpolate the constant function $\mathbf{1}(x) := 1 \quad \forall x$
- Then we get the following, for *all* x:

$$1 = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j}$$

Barycentric Lagrange interpolation II

- Now suppose we interpolate the constant function $\mathbf{1}(x) := 1 \quad \forall x$
- Then we get the following, for *all* x:

$$1 = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j}$$

- Dividing $p(x)$ by this we get

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x-t_j} y_j}{\sum_{j=0}^{n} \frac{w_j}{x-t_j}}$$

Barycentric Lagrange interpolation II

- Now suppose we interpolate the constant function $\mathbf{1}(x) := 1 \quad \forall x$
- Then we get the following, for *all* x:

$$1 = \ell(x) \sum_{j=0}^{n} \frac{w_j}{x - x_j}$$

- Dividing $p(x)$ by this we get

$$p(x) = \frac{\sum_{j=0}^{n} \frac{w_j}{x-t_j} y_j}{\sum_{j=0}^{n} \frac{w_j}{x-t_j}}$$

- This is the **barycentric form** of Lagrange interpolation

Barycentric Lagrange interpolation III

- For a given set of nodes x_j:

Barycentric Lagrange interpolation III

- For a given set of nodes x_j:
- Calculate the weights w_j *once*: $\mathcal{O}(N^2)$ operations

Barycentric Lagrange interpolation III

- For a given set of nodes x_j:
- Calculate the weights w_j *once*: $\mathcal{O}(N^2)$ operations
- Evaluate the interpolant $p(x)$ at x: $\mathcal{O}(N)$ operations

Barycentric Lagrange interpolation III

- For a given set of nodes x_j:
- Calculate the weights w_j *once*: $\mathcal{O}(N^2)$ operations
- Evaluate the interpolant $p(x)$ at x: $\mathcal{O}(N)$ operations

- This algorthm is numerically stable (despite the divisions)

Summary

- Degree-n polynomial **interpolates** $(n+1)$ data points
- Can construct Lagrange polynomial that **interpolates**
- Given in terms of a new cardinal basis

- The barycentric form gives a practical algorithm