Lecture 5: Lagrange Basis

Today:

- A useful choice of interpolation points — ***Chebyshev points***
- ***Lagrange basis*** for (global) polynomial interpolation

Reminder from last week:
Where are we with approximating functions?

We obtained a protocol for ***function interpolation…***

$$f_a(x) = c_0 b_0(x) + c_1 b_1(x) + \cdots + c_n b_n(x)$$

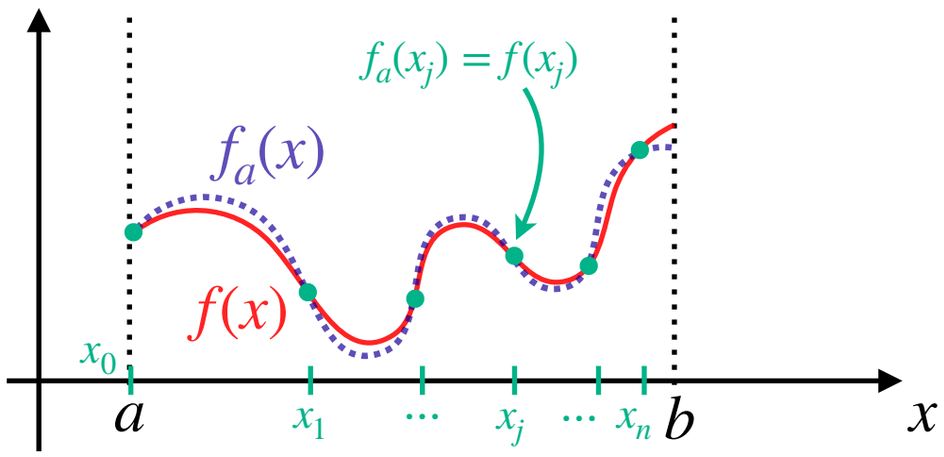
(A) Pick a vector space, $f(x) \in \mathscr{V}$, and a subspace, $f_a(x) \in \mathscr{W}$.

(B) Pick a basis for that subspace, $\mathscr{B} = \{b_0(x), b_1(x), \ldots, b_n(x)\}$.

(C) Break domain $[a, b]$ into $n + 1$ interpolation points

(D) Require: $f_a(x_j) = f(x_j)$ at the interpolation points x_j

$n + 1$ equations and $n + 1$ unknowns



$$f_a(x_j) = f(x_j), \quad j = 0, \ldots, n$$

$$\Longrightarrow \sum_{k=0}^{n} c_k b_k(x_j) = f(x_j), \quad j = 0, \ldots, n$$

$$\Longrightarrow \begin{bmatrix} b_0(x_0) & b_1(x_0) & \cdots & b_{n-1}(x_0) & b_n(x_0) \\ b_0(x_1) & b_1(x_1) & \cdots & b_{n-1}(x_1) & b_n(x_1) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ b_0(x_{n-1}) & b_1(x_{n-1}) & \cdots & b_{n-1}(x_{n-1}) & b_n(x_{n-1}) \\ b_0(x_n) & b_1(x_n) & \cdots & b_{n-1}(x_n) & b_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix}$$

Write as $\mathbf{Ac} = \mathbf{f}$

(E) Solve linear system for the coefficients $c_0, \ldots, c_n$

(F) We now have our interpolant, $f_a(x)$!

Questions for today:

- What interpolation points should we use?
- What basis should we use?

Interpolation points

Choice of interpolation points

The intuitive choice: we might naturally want to use equispaced points

$$x_j = a + j\frac{(b-a)}{n}, \quad j = 0,\ldots,n$$

This choice works OK for small n

Can lead to ***catastrophic failures*** for high n (HW)

A much better choice: Chebyshev point distribution

$$x_j = \frac{1}{2}(a+b) + \frac{1}{2}(b-a)\cos\left(\frac{j\pi}{n}\right), \quad j = 0,\ldots,n$$

Rationale for ***why*** this point distribution is better: subtle, not covered in class

But, we will see that it is better in a HW

Important! This result is specific to *polynomial interpolation*!

Lagrange Basis

What basis to use for ***polynomial*** interpolation

We motivated polynomial interpolation using the ***monomial basis*** $\mathcal{B}_m = \{1, x, x^2, \ldots, x^n\}$

We will find in HW 1 that this basis is not well suited to numerical computations

A better basis is the ***Lagrange basis***, $\mathcal{B}_l = \{L_0(x), L_1(x), \ldots, L_n(x)\}$, where

$$L_i(x) = \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{j=0}^{n} (x_i - x_j)}$$

That's a mouthful! Let's look at a concrete example…

Take $n = 2$ and $i = 0$. Then we have

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

Important! This result is specific to *polynomial interpolation*!

Activity: Lagrange basis functions

$$L_i(x) = \frac{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j)}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}$$

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

(A) How many Lagrange basis polynomials are there for $n = 2$? Write them out.

There are three basis polynomials. The basis for $\mathscr{P}^2[a, b]$ is $\mathscr{B} = \{L_0(x), L_1(x), L_2(x)\}$

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}, L_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}, L_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

(B) What type of polynomial is each $L_i(x)$?

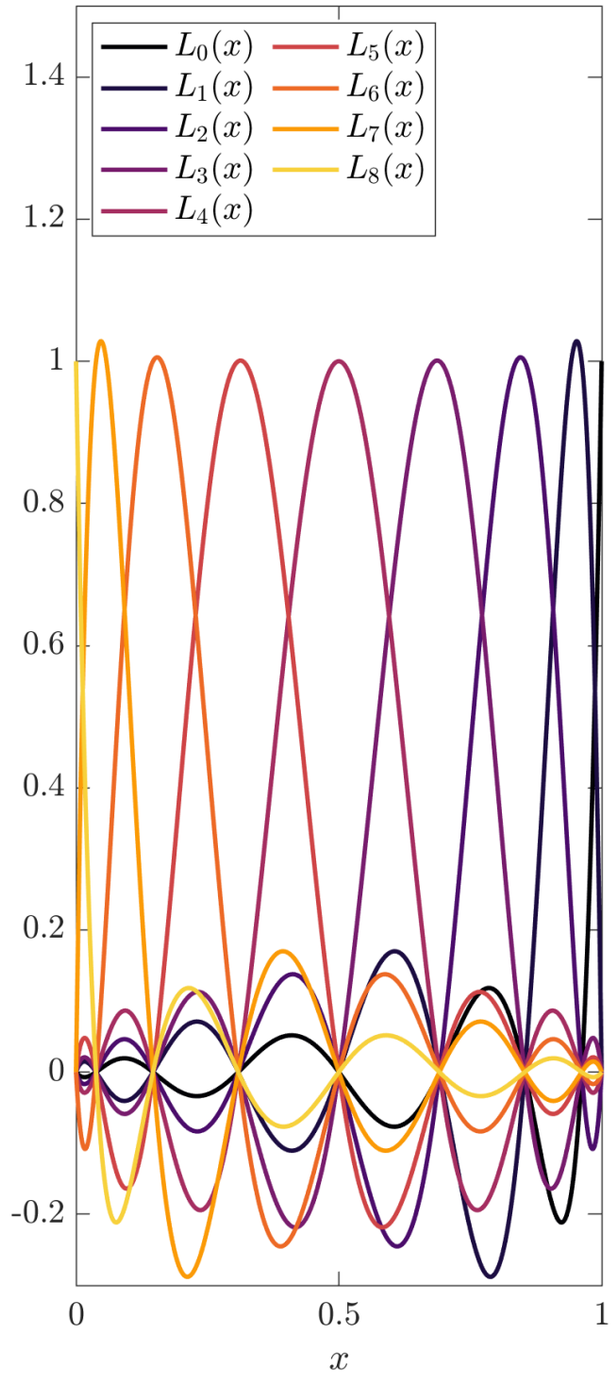
Each polynomial is a degree 2 polynomial. (More generally, for $\mathscr{P}^n[a, b]$, each $L_i(x)$ is a degree-n polynomial)

But why is this a better basis?

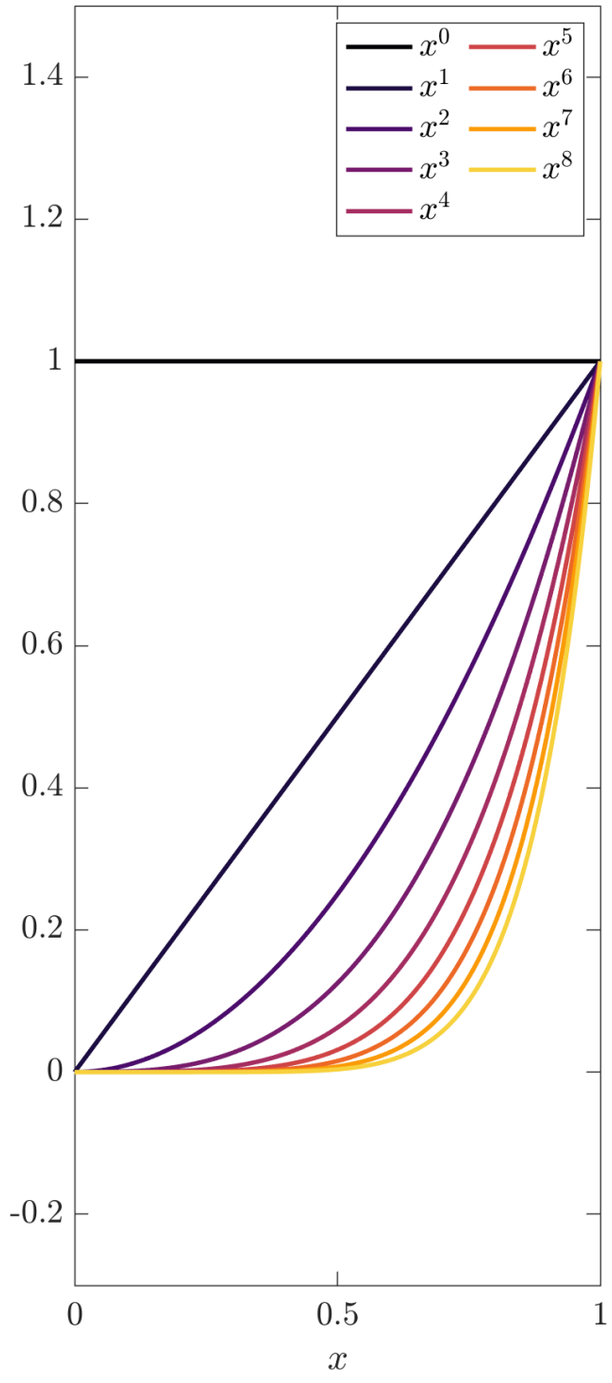
Why is the Lagrange basis superior to the monomial basis?

Activity. Consider the images containing the first nine monomial and Lagrange basis functions. From these images, hypothesize why you think the Lagrange basis is superior for doing computations.

Lagrange basis



Monomial basis

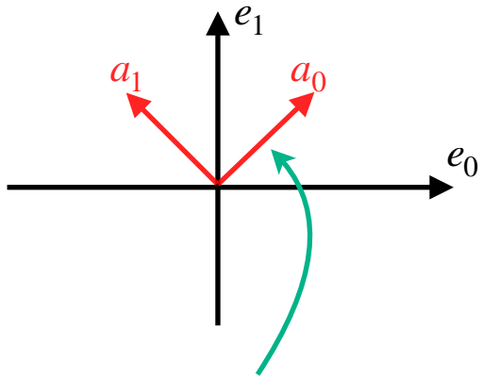


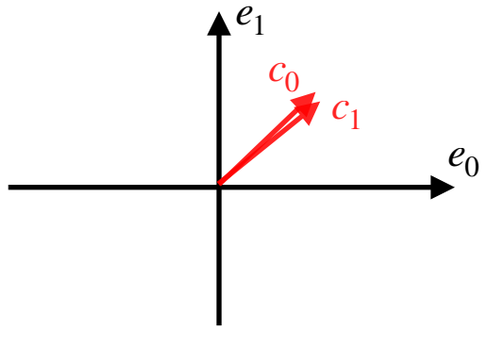
Discussion about superiority of Lagrange basis

The Lagrange basis functions are ***more*** linearly independent than the monomials.

$\Longrightarrow$ better for numerical computation

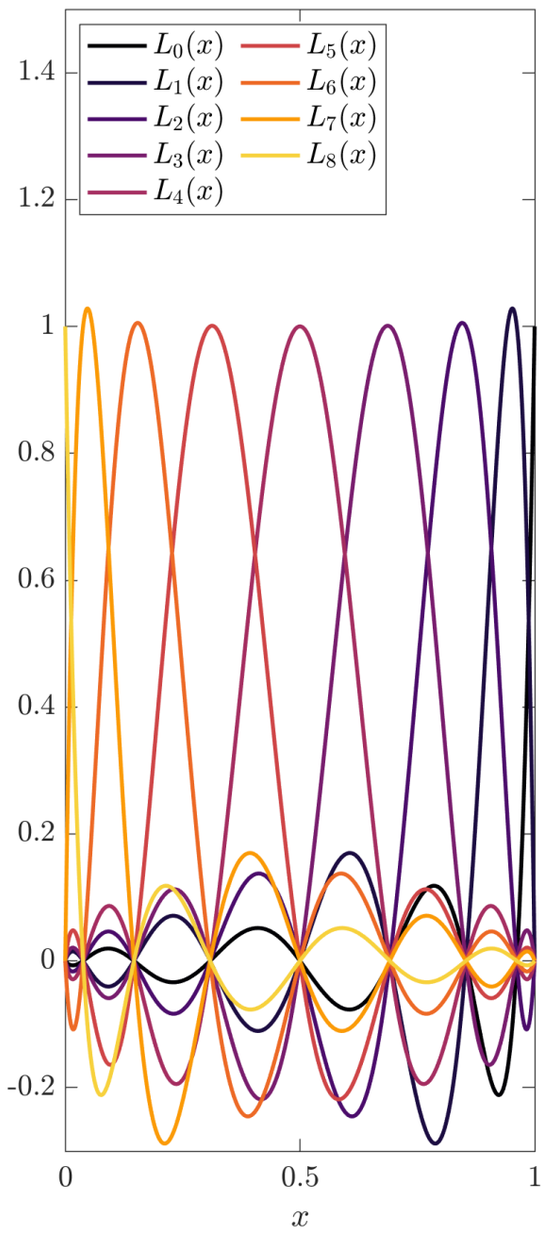
This is a perfect analogy to basis vectors in $\mathbb{R}^2$



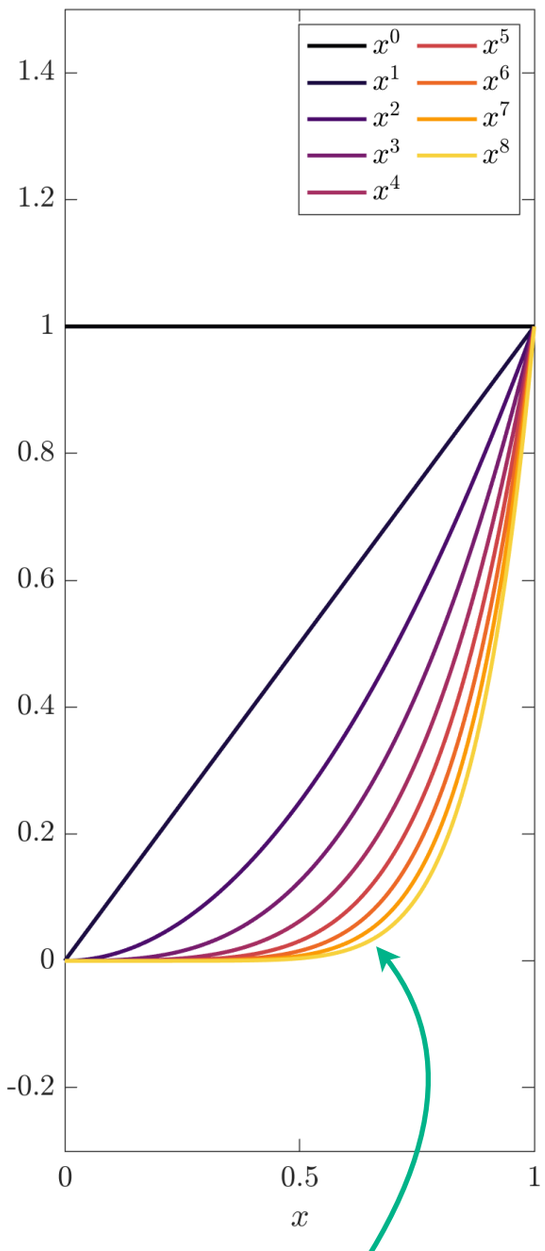


This is a better basis to work with!

Lagrange basis



Monomial basis



With increasing power, these functions look more and more similar!

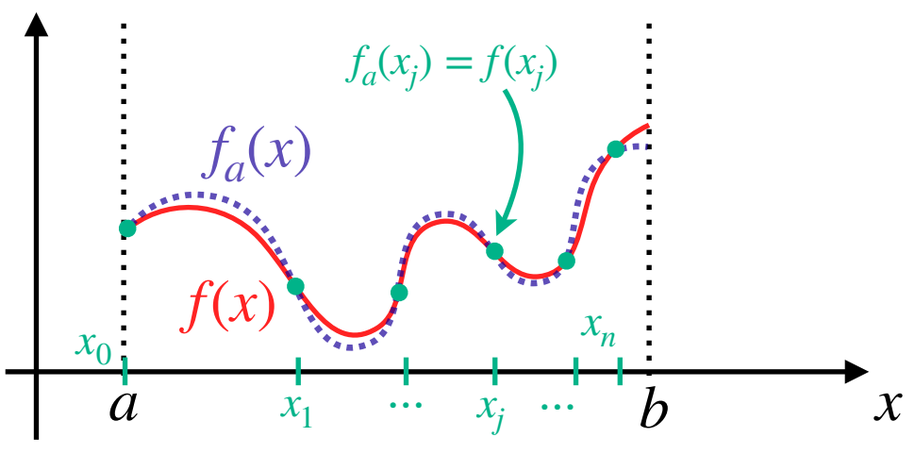
Using Lagrange basis functions in interpolation

We have defined Lagrange basis functions and given intuition for why they are a good basis to use.

But how do we construct our approximation $f_a(x)$ using this basis?

The procedure is the same as before!

(A) Pick a vector space, $\mathscr{V} = C[a,b]$, and a subspace, $\mathscr{W} = \mathscr{P}^n[a,b]$.
(B) Pick a basis for that subspace, $\mathscr{B}_l = \{L_0(x), L_1(x), \ldots, L_n(x)\}$.
(C) Break domain $[a,b]$ into $n+1$ interpolation points
(D) Require: $f_a(x_j) = f(x_j)$ at the interpolation points x_j



$$f_a(x_j) = f(x_j), \quad j = 0,\ldots,n$$

$$\Longrightarrow \sum_{k=0}^{n} c_k L_k(x_j) = f(x_j), \quad j = 0,\ldots,n$$

$$\Longrightarrow \begin{bmatrix} L_0(x_0) & L_1(x_0) & \cdots & L_{n-1}(x_0) & L_n(x_0) \\ L_0(x_1) & L_1(x_1) & \cdots & L_{n-1}(x_1) & L_n(x_n) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ L_0(x_{n-1}) & L_1(x_{n-1}) & \cdots & L_{n-1}(x_{n-1}) & L_n(x_{n-1}) \\ L_0(x_n) & L_1(x_n) & \cdots & L_{n-1}(x_n) & L_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix}$$

We can say more about these entries in $\mathbf{A}$!

How to interpolate with the Lagrange basis

So to interpolate with the Lagrange basis we need to solve for the coefficients from

$$\begin{bmatrix} L_0(x_0) & L_1(x_0) & \cdots & L_{n-1}(x_0) & L_n(x_0) \\ L_0(x_1) & L_1(x_1) & \cdots & L_{n-1}(x_1) & L_n(x_1) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ L_0(x_{n-1}) & L_1(x_{n-1}) & \cdots & L_{n-1}(x_{n-1}) & L_n(x_{n-1}) \\ L_0(x_n) & L_1(x_n) & \cdots & L_{n-1}(x_n) & L_n(x_n) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix}$$

But the entries in the matrix are just $L_i(x_j)$ where i is the column and j is the row

Magic: $L_i(x_j) = \begin{cases} 1 & i = j \\ 0 & else \end{cases}$

We can build intuition for this being true by returning to the case where $n = 2$ and $i = 1$. Then we have

$$L_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$= 0$ when $x = x_1$, $x = x_2$
$= 1$ when $x = x_0$

Identity matrix

$$\implies \mathbf{A} = \mathbf{I} \implies \mathbf{Ac} = \mathbf{f} \implies \mathbf{c} = \mathbf{f} \implies c_j = f(x_j)$$

Important distinction!

There are two ***separate*** issues that we addressed today that affect accuracy of polynomial interpolation. The accuracy of our ***polynomial interpolant***, $f_a(x)$, is influenced by:

(A) The choice of point distribution. This deals with the accuracy of the polynomial interpolant itself. If a uniform distribution is used, the interpolant will (often) be an inaccurate approximation of the true $f(x)$.

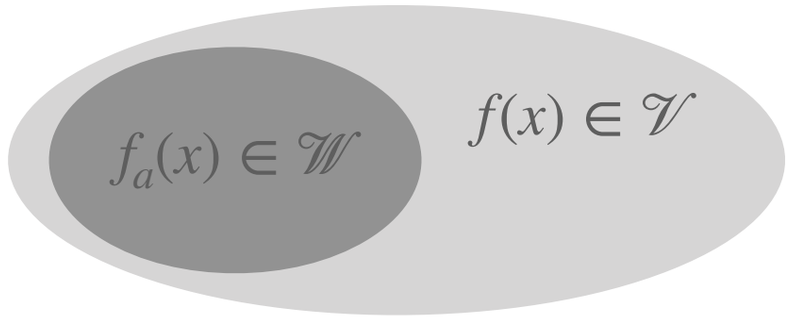
(B) The choice of basis. This does not, in principle, change the polynomial interpolant. ***In infinite precision, we would get exactly the same interpolant irrespective of the basis used!*** The issue is that to compute the interpolant we solve the system $\mathbf{Ac} = \mathbf{f}$ on a computer.

Why?

- Certain bases (like the monomial basis) lead to an $\mathbf{A}$ that is nearly not invertible, so that when a computer rounds off its answers, there are ***large*** errors in the computed coefficients.
- Other bases (like the Lagrange basis) lead to an $\mathbf{A}$ that is trivially invertible so that the coefficients can. Be computed to very high accuracy!

Summary of function approximation concepts

Function approximation concepts…



Vector space — used to represent $f(x)$

Subspace — used to restrict representation of $f_a(x)$

Inner product — used to define a ***norm*** — used to measure approximation error

Basis functions — used to represent $f_a(x)$ such that we can solve for coefficients

$$||e|| = \sqrt{(e,e)} \qquad f_a(x) = c_0 b_0(x) + c_1 b_1(x) + \cdots + c_n b_n(x)$$

Function approximation methods…

How do we choose $b_0(x), b_1(x), \ldots, b_n(x)$ and solve for $c_0, c_1, \ldots, c_n$?

$n+1$ equations and $n+1$ unknowns

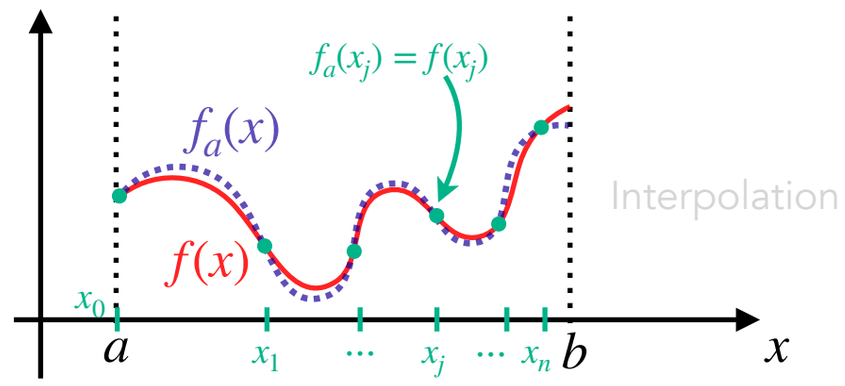
Interpolation

Polynomial interpolation, $f_a(x) \in \mathscr{P}^n[a,b]$

Global: monomial basis, Lagrange basis

Local: cubic spline

Today!



Interpolation

Trigonometric interpolation, $f_a(x) \in \mathscr{T}^n[a,b]$

Least Squares

$f(x) \in C[a,b] \qquad f_a(x) \in \mathscr{P}^n[a,b]$

$$f_a(x) = c_0 L_0(x) + c_1 L_1(x) + \cdots + c_n L_n(x)$$