Modes and Nodes (aka Functions and Points)

Both function basis and point set are under our control. What do we pick?

Ideas for basis functions:
- Monomials $1, x, x^2, x^3, x^4, \ldots$
- Functions that make $V = I \rightarrow \text{‘Lagrange basis’}$
- Functions that make $V$ triangular $\rightarrow \text{‘Newton basis’}$
- Splines (piecewise polynomials)
- Orthogonal polynomials
- Sines and cosines
- ‘Bumps’ (‘Radial Basis Functions’)

Ideas for points:
- Equispaced
- ‘Edge-Clustered’ (so-called Chebyshev/Gauss/… nodes)

Specific issues:
- Why not monomials on equispaced points?
  **Demo:** Monomial interpolation
  [cleared]
- Why not equispaced?
  **Demo:** Choice of Nodes for Polynomial Interpolation
  [cleared]
Lagrange Interpolation

Find a basis so that $V = I$, i.e.

$$f(x) \approx \sum_{i=1}^{m} \alpha_i \varphi_i(x)$$

$$\varphi_j(x_i) = \begin{cases} 1 & i = j, \\ 0 & \text{otherwise.} \end{cases}$$

1. $\varphi_j(x_i) = 0$ if $i \neq j$ (m-1 conditions)

   $$\Rightarrow (x-x_i) \mid \varphi_j(x), \ i \neq j$$

   $$\varphi_j(x) = \prod_{j \neq i} (x-x_i) \text{ up to a constant}$$

2. $\varphi_j(x_j) = 1$

   $$\Rightarrow \varphi_j(x_i) = \frac{\prod_{j \neq i} (x-x_i)}{\prod_{j \neq i} (x_j-x_i)}$$
Lagrange Polynomials: General Form

\[ \varphi_j(x) = \prod_{k=1, k \neq j}^m \frac{(x - x_k)}{(x_j - x_k)} \]

\[ \text{interpolant } P_m(x) = \sum_{j=1}^m f(x_j) \varphi_j(x) \]

\[ \Lambda_m = \max_x \sum_{i=1}^m |L_i(x)| \]
Newton Interpolation

Find a basis so that $V$ is triangular.

$$\varphi_j(\mathbf{x}) = \frac{1}{i!} \prod_{k=1}^{i-1} (x - x_k)$$

- Divided difference

$O(m^2)$

Why not Lagrange/Newton?

Both have $O(m^2)$ ops, expensive to do calculus.
Better conditioning: Orthogonal polynomials

What caused monomials to have a terribly conditioned Vandermonde?

- Close to linear dep.

What’s a way to make sure two vectors are not like that?

- Orthogonality

But polynomials are functions!
Orthogonality of Functions

How can functions be orthogonal?

\[ \sum_{i=1}^{n} \int f_i(x) g_i(x) \, dx = 0 \]
Constructing Orthogonal Polynomials

How can we find an orthogonal basis?

**Demo:** Orthogonal Polynomials [cleared] — Got: Legendre polynomials.
But how can I practically compute the Legendre polynomials?

\[
(f, g) = \int_{-1}^{1} w(x) f(x) g(x) \, dx
\]

\[
\Rightarrow w(x) = \frac{1}{\sqrt{1-x^2}}
\]
Chebyshev Polynomials: Definitions

Three equivalent definitions:

- Result of Gram-Schmidt with weight $1/\sqrt{1-x^2}$. What is that weight?

  ![Graph: half-circle](attachment://half_circle.png)

  \[ y = \sqrt{1-x^2} \quad (x^2+y^2=1) \]

  (Like for Legendre, you won’t exactly get the standard normalization if you do this.)

- \[ T_k(x) = \cos(k \cos^{-1}(x)) \]

- \[ T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \] plus \( T_0 = 1, \ T_0 = x \)

**Demo:** Chebyshev Interpolation  [cleared]  (Part 1)
Chebyshev Interpolation

What is the Vandermonde matrix for Chebyshev polynomials?

\[
V_{ij} = T_j(x_i) = \cos \left( j \cdot \cos^{-1}(x_i) \right)
\]

\[
x_i = \cos(i/n), \quad x_i = \cos \left( \frac{i}{N} \pi \right)
\]

\[
\Rightarrow V_{ij} = \cos \left( j \cdot \frac{i}{N} \pi \right)
\]

- matvec of $V$ is Discrete Cosine Transform (DCT)
- $\text{FFT } \Rightarrow O(N \log N)$
- $x_i$ are extrema of $T_n$
Chebyshev Nodes

Might also consider roots (instead of extrema) of $T_k$:

$$x_i = \cos \left( \frac{2i - 1}{2k} \pi \right) \quad (i = 1 \ldots, k).$$

Vandermonde for these (with $T_k$) can be applied in $O(N \log N)$ time, too.

It turns out that we were still looking for a good set of interpolation nodes. We came up with the criterion that the nodes should bunch towards the ends. Do these do that?

Yes.

Demo: Chebyshev Interpolation  [cleared] (Part 2)
Chebyshev Interpolation: Summary

- Chebyshev interpolation is fast and works extremely well
- In 1D, they’re a very good answer to the interpolation question
- But sometimes a piecewise approximation (with a specifiable level of smoothness) is more suited to the application
In-Class Activity: Interpolation

In-class activity: Interpolation
Interpolation Error

If $f$ is $n$ times continuously differentiable on a closed interval $I$ and $p_{n-1}(x)$ is a polynomial of degree at most $n$ that interpolates $f$ at $n$ distinct points $\{x_i\}$ ($i = 1, \ldots, n$) in that interval, then for each $x$ in the interval there exists $\xi$ in that interval such that

$$f(x) - p_{n-1}(x) = \frac{f^{(n)}(\xi)}{n!}(x - x_1)(x - x_2)\cdots(x - x_n).$$