Modes and Nodes (aka Functions and Points)

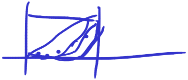
Both function basis and point set are under our control. What do we pick?

Ideas for basis functions:

- Monomials $1, x, x^2, x^3, x^4, \ldots$
- Functions that make $V = I \rightarrow$ 'Lagrange basis'
- Functions that make V triangular $\rightarrow$ '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:

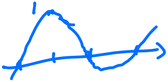
- Why *not* monomials on equispaced points? **Demo:** Monomial interpolation [cleared]
- Why not equispaced? **Demo:** Choice of Nodes for Polynomial Interpolation [cleared]

Lagrange Interpolation

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

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

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



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

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

$\varphi_j(x) = \prod_{j \neq i} (x - x_i)$ 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

Lagrange func.

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

interpolant $P_m(x) = \sum_{j=1}^{m} f(x_j)\,\varphi_j(x)$

$$\Lambda_m = \max_x \sum_{i=1}^{m} |\ell_i(x)|$$

Newton Interpolation

Find a basis so that V is triangular.

$$\varphi_j(x) = \prod_{k=1}^{j-1} (x - x_k)$$

= 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?

$$\text{orth} \Leftrightarrow 0 \overset{!}{=} \vec{f} \cdot \vec{g} = \sum_{i=1}^{n} f_i g_i = (\vec{f}, \vec{g})$$

$$= \int_{-1}^{1} f(x) \cdot g(x) = (f, g) = 0$$

Constructing Orthogonal Polynomials

How can we find an orthogonal basis?

Gram - Schmidt

Demo: Orthogonal Polynomials [cleared] — Got: Legendre polynomials.

But how can I practically compute the Legendre polynomials?

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

$$\hookrightarrow \; w(x) = \frac{1}{\sqrt{1-x^2}}$$

Chebyshev Polynomials: Definitions

Three equivalent definitions:

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

  $\frac{1}{\ }$: "half circle" $y = \sqrt{1-x^2}$ $(x^2+y^2=1)$

  (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)

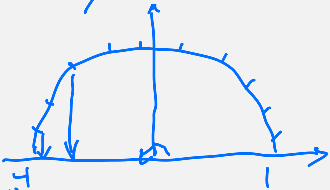
Chebyshev Interpolation

What is the Vandermonde matrix for Chebyshev polynomials?

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

  $x_i = \cos(*)$ $\quad x_i = \cos\left(\frac{i}{k}\pi\right)$

  $\Rightarrow V_{ij} = \cos\left(j \frac{i}{k}\pi\right)$



- matvec of V is

  Discrete Cosine Transform. (DCT)

  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:

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

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
- http://www.chebfun.org/ and: ATAP
- 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 $\{x_i\}$ $(i = 1, ..., n)$ in that interval, then for each x in the interval there exists ξ in that interval such that

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