

Interpolation

Accuracy

Monomial Basis Problems

Orthogonal Bases

Chebyshev

choice of nodes

choice of fun's

Outline

Python, Numpy, and Matplotlib
Making Models with Polynomials
Making Models with Monte Carlo
Error, Accuracy and Convergence
Floating Point
Modeling the World with Arrays
 The World in a Vector
 What can Matrices Do?
 Graphs
 Sparsity
Norms and Errors
The 'Undo' Button for Linear Operations: LU
Repeating Linear Operations:
Eigenvalues and Steady States
Eigenvalues: Applications

Approximate Undo: SVD and Least Squares

SVD: Applications

 Solving Funny-Shaped Linear Systems
 Data Fitting
 Norms and Condition Numbers
 Low-Rank Approximation

Interpolation

Iteration and Convergence

Solving One Equation

Solving Many Equations

Finding the Best: Optimization in 1D

Optimization in n Dimensions

Recap: Interpolation

Starting point: Looking for a linear combination of functions φ_i to hit given data points (x_i, y_i) .

Interpolation becomes solving the linear system:

$$y_i = f(\underbrace{x_i}_{\text{nodes}}) = \sum_{j=0}^{N_{\text{func}}} \underbrace{\alpha_j}_{\text{coeffs}} \underbrace{\varphi_j(x_i)}_{V_{ij}} \quad \Leftrightarrow \quad V\alpha = \mathbf{y}.$$

Want unique answer: Pick $N_{\text{func}} = N \rightarrow V$ square.

V is called the (generalized) Vandermonde matrix.

Main lesson:

$$V (\text{coefficients}) = (\text{values at nodes}).$$

Rethinking Interpolation

We have so far always used monomials $(1, x, x^2, x^3, \dots)$ and equispaced points for interpolation. It turns out that this has *significant problems*.

Demo: Monomial interpolation

Demo: Choice of Nodes for Polynomial Interpolation

Interpolation: Choosing Basis Function and Nodes

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

Ideas for basis functions:

- ▶ Monomials $1, x, x^2, x^3, x^4, \dots$
- ▶ 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 nodes:

- ▶ Equispaced
- ▶ 'Edge-Clustered' (so-called Chebyshev/Gauss/... nodes)

$$\begin{array}{l} (x-x_1)(x-x_2) \\ (x-x_2)(x-x_3) \\ (x-x_1)(x-x_3) \end{array}$$

Better Conditioning: Orthogonal Polynomials $x^{50} \approx \frac{1}{2}x^{49} + \frac{1}{2}x^{51}$

What caused monomials to have a terribly conditioned Vandermonde?

functions are almost linearly dependent

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

orthogonality

But polynomials are functions!

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

$$\langle f, g \rangle_w = \int w(x) f(x) g(x) dx$$

Legendre Polynomials

Start! $[1, x, x^2, x^3]$

orthogonalize f w.r.t. g

$$\bar{f}(x) = f(x) - \langle f, g \rangle \cdot g(x)$$

orthogonalize monomial w.r.t. previous

Better Conditioning: Orthogonal Polynomials (II)

But how can I practically compute the Legendre polynomials?

look them up!

3-term recurrence

$$L_i = f(L_{i-1}, L_{i-2}, L_{i-3})$$


Another Family of Orthogonal Polynomials: Chebyshev

Three equivalent definitions:

$$\omega(x) \quad \langle f, g \rangle_\omega = \int_{-1}^1 fg \omega$$

- ▶ Result of Gram-Schmidt with weight $1/\sqrt{1-x^2} = y$

What is that weight?

$$y^2 = 1 - x^2 \Rightarrow x^2 + y^2 = 1$$


- ▶ $T_k(x) = \cos(k \cos^{-1}(x))$
- ▶ $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$

Demo: Chebyshev interpolation part I

What are good nodes to use with Chebyshev polynomials?

$$x_i = \cos\left(\frac{i}{n}\pi\right)$$
$$T_k(x_i) = \cos\left(\frac{ik}{n}\pi\right) \leftarrow \text{equispaced}$$

Chebyshev Nodes

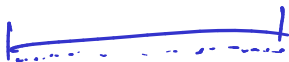
Might also consider zeros (instead of roots) of T_k :

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

The 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?



Demo: Chebyshev interpolation part II

$$\begin{array}{c}
 \swarrow \\
 \downarrow \\
 \checkmark \\
 z = y \\
 \uparrow \\
 \text{solve}
 \end{array}
 \begin{bmatrix}
 f_1(x_1) \dots f_{n-1}(x_n) \\
 \vdots \\
 f_0(x_{n-1})
 \end{bmatrix}$$

given \bar{x} (some point)

$$\begin{aligned}
 \tilde{f}(\bar{x}) &= \sum_i z_i \cdot f_i(\bar{x}) \\
 &= z^T \cdot \begin{bmatrix} f_0(\bar{x}) \\ \vdots \\ f_{n-1}(\bar{x}) \end{bmatrix}
 \end{aligned}$$

Calculus on Interpolants

Suppose we have an interpolant $\tilde{f}(x)$ with $f(x_i) = \tilde{f}(x_i)$ for $i = 1, \dots, n$:

$$\tilde{f}(x) = \alpha_1 \varphi_1(x) + \dots + \alpha_n \varphi_n(x)$$

How do we compute the derivative of \tilde{f} ?

$$\tilde{f}'(x) = \alpha_1 \varphi_1'(x) + \dots + \alpha_n \varphi_n'(x)$$
$$f'(x) \approx \tilde{f}'(x)$$

Suppose we have function values at nodes $(x_i, f(x_i))$ for $i = 1, \dots, n$ for a function f . If we want $f'(x_i)$, what can we do?

exact is hard

$$z = V(x)^{-1} f(x) \Rightarrow \tilde{f}(x)$$

differentiate $\tilde{f}(x)$