CS 450: Numerical Anlaysis[1]

Interpolation

University of Illinois at Urbana-Champaign

[1] *These slides have been drafted by Edgar Solomonik as lecture templates and supplementary material for the book "Scientific Computing: An Introductory Survey" by Michael T. Heath (slides).*

Interpolation

- Given $(t_1, y_1), \ldots, (t_m, y_m)$ with *nodes* $t_1 < \cdots < t_m$ an *interpolant* f satisfies:

- Interpolant is usually constructed as linear combinations of *basis functions* $\{\phi_j\}_{j=1}^n = \phi_1, \ldots, \phi_n$ so $f(t) = \sum_j x_j \phi_j(t)$.

Polynomial Interpolation

- The choice of *monomials* as basis functions, $\phi_j(t) = t^{j-1}$ yields a degree $n-1$ polynomial interpolant:

- Polynomial interpolants are easy to evaluate and do calculus on:

Demo: Monomial interpolation

Conditioning of Interpolation

- Conditioning of interpolation matrix $\boldsymbol{A}$ depends on basis functions and coordinates $t_1, \ldots, t_m$:

- The Vandermonde matrix tends to be ill-conditioned:

Lagrange Basis

- n-points fully define the unique $(n-1)$-degree polynomial interpolant in the *Lagrange basis*:

- Lagrange polynomials yield an ideal Vandermonde system, but the basis functions are hard to evaluate and do calculus on:

Newton Basis

- The *Newton basis* functions $\phi_j(t) = \prod_{k=1}^{j-1}(t - t_k)$ with $\phi_1(t) = 1$ seek the best of monomial and Lagrange bases:
- The Newton basis yields a triangular Vandermonde system:

Orthogonal Polynomials

- Recall that good conditioning for interpolation is achieved by constructing a well-conditioned Vandermonde matrix, which is the case when the columns (corresponding to each basis function) are orthonormal. To construct robust basis sets, we introduce a notion of *orthonormal functions*:

Legendre Polynomials

- The Gram-Schmidt orthogonalization procedure can be used to obtain an orthonormal basis with the same span as any given arbitrary basis:

$\langle f, g \rangle_w = \int_{-1}^{1} f(t) g(t) w(t)\, dt$

$p = \{1, t, t^2, \ldots\}$

$\hat{e}_i = p_i - \sum_{j=0}^{i-1} \frac{\langle p_i, e_j \rangle_w e_j}{\langle e_j, e_j \rangle_w}$

can choose

$e_i = \frac{\hat{e}_i}{\|e_i\|}$

standard

$e_i = \hat{e}_i / \hat{e}_i(1)$

- The *Legendre polynomials* are obtained by Gram-Schmidt on the monomial basis, with $w(t) = \begin{cases} 1 : -1 \le t \le 1 \\ 0 : \text{otherwise} \end{cases}$ and normalized so $\hat{\phi}_i(1) = 1$.

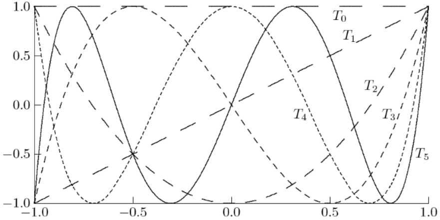
Chebyshev Basis

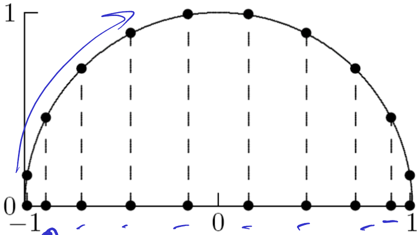
Demo: *Chebyshev interpolation*
Activity: *Chebyshev Interpolation*

- *Chebyshev polynomials* $\phi_j(t) = \cos((j-1)\arccos(t))$ and *Chebyshev nodes* $t_i = \cos\left(\frac{2i-1}{2n}\pi\right)$ provide a way to pick *nodes* $t_1, \ldots, t_n$ along with a basis, to yield perfect conditioning:

Demo: *Jump with Chebyshev Nodes*

Chebyshev Nodes Intuition





more nodes near the endpoints

- Note *equi-oscillation* property, successive extrema of $T_k = \phi_k$ have the same magnitude but opposite sign.
- Set of k Chebyshev nodes of are given by zeros of T_k and are abscissas of points uniformly spaced on the unit circle.

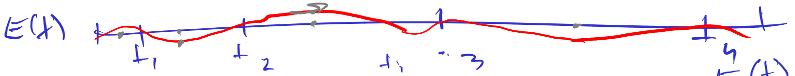
Error in Interpolation

We show by induction that given degree n polynomial interpolant $\tilde{f}$ of f the error $E(t) = f(t) - \tilde{f}(t)$ has n zeros $t_1, \ldots, t_n$ and there exist $y_1, \ldots, y_n$ so

true   interpolant

$t \in [t_1, t_n]$

$$E(t) = \int_{t_1}^{t} \int_{y_1}^{w_0} \cdots \int_{y_n}^{w_{n-1}} f^{(n+1)}(w_n) dw_n \cdots dw_0 \quad (1)$$



$$E(t) = \int_{z}^{t} E'(t) dt$$

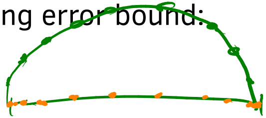
$$0 = E(t_i) - E(t_{i-1}) = \int_{t_{i-1}}^{t_i} E'(t) dt$$

By induction apply thm. to $E'(t)$ with $n-1$ zeros

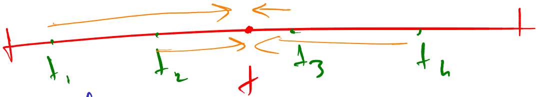
$$E'(t) = \int_{y_1}^{w_0} \cdots \int_{y_n}^{w_{n-1}} \underbrace{E^{(n+1)}(w_n)}_{f^{(n+1)}(w_n)} dw_n \cdots dw_1$$

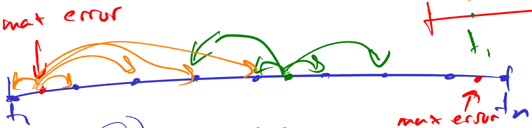
Interpolation Error Bounds

- Consequently, polynomial interpolation satisfies the following error bound:



$$\forall t \in [t_1, t_n] \quad \boxed{|E(t)| \le \max_{s \in [t_1, t_n]}} \frac{|f^{(n+1)}(s)|}{n!} \prod_{j=1}^{n} (t - t_j)$$





- Letting $h = t_n - t_1$ (often also achieve same for h as the node-spacing $t_{i+1} - t_i$), we obtain

$$|E(t)| = O(h^n)$$

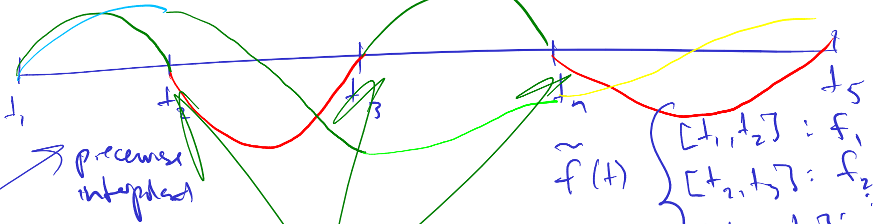
$$C = |f^{(n+1)}(s)| / n!$$

error is much less with smaller interpolation domain

piecewise interpolation

Piecewise Polynomial Interpolation

- The kth piece of the interpolant is a polynomial in $[t_i, t_{i+1}]$



$$\tilde{f}(t) \begin{cases} [t_1, t_2] : f_1(t) \\ [t_2, t_3] : f_2(t) \\ \vdots \\ [t_{n-1}, t_n] : f_n(t) \end{cases}$$

- *Hermite* interpolation ensures consecutive interpolant pieces have same derivative at each *knot* t_i:

$f_1(t_2) = f_2(t_2)$ ← $n-1$ conditions/equations

$f_1'(t_2) = f_2'(t_2)$ ← $n-1$ conditions/equations

cubic piecewise interpolant (each piece is a cubic polynomial)

$4(n-1)$ unknowns

$n + (n-1) + (n-1)$ equations with $4(n-1)$ unknowns

Spline Interpolation

- A *spline* is a $(k-1)$-time differentiable piecewise polynomial of degree k:

at each knot k equations, so overall

$k(n-1)$ equations for continuity/differentiability

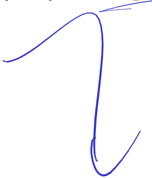
given n points/nodes, $n + k(n-1)$ equations, $(k+1)n$ unknowns

k equations to choose, $k=3$ (cubic), 2 equations left,

- The resulting interpolant coefficients are again determined by an appropriate *generalized Vandermonde system*:

and we choose $f''(t_1) = f''(t_n) = 0$

$n=2$ cubic natural spline



given $f(x_1)=y_1$, $x_1 \in [t_1, t_2]$ $\quad f(x_2)=y_2$ $\quad x \in [t_2, t_3]$, $\quad f(t_1)=y_1$, $f(t_2)=y_2$, $f(t_3)=y_3$

$$
\begin{bmatrix}
1 & t_1 & t_1^2 & t_1^3 & & & & \\
1 & t_2 & t_2^2 & t_2^3 & & & & \\
 & 1 & 2t_2 & 3t_2^2 & & -1 & -2t_2 & -3t_2^2 \\
 & & 2 & 6t_2 & & & -2 & -6t_2 \\
 & & & & 1 & t_2 & t_2^2 & t_2^3 \\
 & & 2 & 6t_1 & & & & \\
 & & & & & & 2 & 6t_3 \\
 & & & & 1 & t_3 & t_3^2 & t_3^3
\end{bmatrix}
\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{bmatrix}
\begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \text{coeff 1st pred} \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \text{coeff 2nd pred} \end{array}
=
\begin{bmatrix} y_1 \\ y_2 \\ 0 \\ 0 \\ y_2 \\ 0 \\ 0 \\ y_3 \end{bmatrix}
$$

$$
f(t) = \begin{cases} t \in [t_1, t_2] : \alpha_1 + \alpha_2 t + \alpha_3 t^2 + \alpha_4 t^3 \\ t \in [t_2, t_3] : \beta_1 + \beta_2 t + \beta_3 t^2 + \beta_4 t^3 \end{cases}
$$

$\boxed{f_2(t_3) = y_3}$

interpolat $\quad f(t_1) = y_1 \Rightarrow f_1(t_1) = y_1$, $\quad f(t_2) = y_2 \Rightarrow f_1(t_2) = f_2(t_2) = y_2$

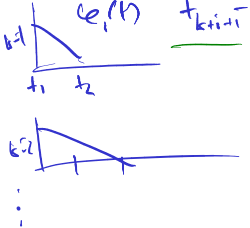
B-Splines

B-splines provide an effective way of constructing splines from a basis:

- The basis functions can be defined recursively with respect to degree.

$$\varphi_i^k(t) = \frac{t - t_i}{t_{k+i+1} - t}$$

$\varphi_i(t)$

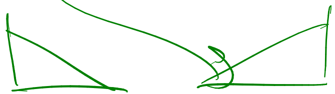


degree

$$f_i^k(t) = \varphi_i^k(t) f_i^{k-1}(t) - \varphi_{i+1}^k(t) f_{i+1}^{k-1}(t)$$

point

$$f_i^0(t) = \begin{cases} 1 & : [i, i+1] \\ 0 & : \text{elsewhere} \end{cases}$$



- The ith degree k polynomial piece is positive on $[t_i, t_{i+k+1}]$ and zero everywhere else
- All possible splines of degree k with nodes $\{t_i\}_{i=1}^n$ can be represented in the basis.
- The resulting interpolant coefficients are again determined by an appropriate generalized Vandermonde system.