APM 506 – Computational Methods
Finite difference formulas and polynomial interpolation

Rodrigo B. Platte

ASU School of Mathematical & Statistical Sciences

ARIZONA STATE UNIVERSITY

Spring 2017

Syllabus

Instructor
Dr. Rodrigo Platte
Office: Goldwater Center 634
E-mail: rbp@asu.edu

Course syllabus is posted on blackboard.
Please do not check email, browse the internet, ..., while in class.

MATLAB

MATLAB will be used extensively in class and will be required for HW.

Reference: Learning MATLAB by Tobin A Driscoll.

Reference: Crash course in MATLAB by Tobin A Driscoll (see blackboard)

Learn how to use MATLAB's PUBLISH. Used it to turn in your HW.

Taylor expansions

Recall that if f is analytic at a then

$$f(z) = \sum_{j=0}^{\infty} \frac{f^{(j)}(a)}{j!}(z-a)^j,$$

for $|z-a| < R$, where R is the radius of convergence of the series.

Now, let $z = x \pm h$, and $a = x$. That is, we will center the series at x and use it to approximate f at $x \pm h$. Then

$$f(x \pm h) = \sum_{j=0}^{\infty} (\pm 1)^j \frac{f^{(j)}(x)}{j!}(h)^j,$$

$$f(x+h) = f(x) + hf'(x) + h^2 f''(x)/2 + \dots$$

$$f(x-h) = f(x) - hf'(x) + h^2 f''(x)/2 - \dots$$

Taylor polynomial

If f is C^{k+1}, then

$$f(z) = \sum_{j=0}^{k} \frac{f^{(j)}(a)}{j!}(z-a)^j + \frac{f^{(k+1)}(\xi)}{(k+1)!}(z-a)^{k+1},$$

where ξ between a and z. Similarly,

$$f(x+h) = f(x) + hf'(x) + h^2 f''(x)/2 + \cdots + \frac{f^{(k+1)}(\xi)}{(k+1)!}(h)^{k+1}.$$

$$f(x-h) = f(x) - hf'(x) + h^2 f''(x)/2 - \cdots + (-1)^{(k+1)} \frac{f^{(k+1)}(\xi)}{(k+1)!}(h)^{k+1}.$$

Big-O notation

When we perform error analysis we often neglect high-order terms in Taylor expansions and use the big-0 notation. For example:

$$f(x-h) = f(x) - hf'(x) + h^2 f''(x)/2 + O(h^3).$$

In particular, the dependence is usually expressed in terms of the dominant term when h is small. For positive functions $f(h)$ and $g(h)$, we say $f(h) = O\big(g(h)\big)$ ("f is **big-O** of g") as $h \to 0$ if $f(h)/g(h)$ is bounded above for all sufficiently small h. We say $f(h) \sim g(h)$ ("f is **asymptotic** to g") as $h \to 0$ if $f(h)/g(h) \to 1$ as $h \to 0$. Clearly, $f \sim g$ implies $f = O(g)$; asymptotic notation is more specific than big-O notation.[1]

Examples:

$$2h^2 - h^3 \sim 2h^2 \quad \text{(best)}$$

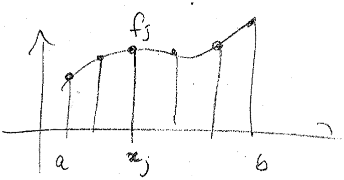
$$2h^2 - h^3 = O(h^2) \quad \text{(good)}$$

$$2h^2 - h^3 = O(h) \quad \text{(loose estimate)}$$

[1] Really, $O(g)$ and $\sim g$ are *sets* of functions, and $\sim g$ is a subset of $O(g)$. That we write $f = O(g)$ rather than $f \in O(g)$ is a quirk of tradition.

Finite difference formulas

We now want to approximate derivative values from function values.



$$f_j = f(x_j)$$

We now that

$$f'(x_j) = \lim_{h \to 0} \frac{f(x_j + h) - f(x_j)}{h} \approx \frac{f(x_{j+1}) - f(x_j)}{h} = \frac{f_{j+1} - f_j}{h}$$

Finite difference formulas

The error in this approximation can be found using Taylor expansions:

$$f(x_j + h) = f(x_j) + hf'(x_j) + h^2 f''(\xi)/2,$$

where $f \in C^2(a, b)$ and $\xi \in (x_j, x_j + h)$.

Hence,

$$\frac{f_{j+1} - f_j}{h} = f'(x_j) + hf''(\xi)/2$$

and

$$\left| \frac{f_{j+1} - f_j}{h} - f'(x_j) \right| = O(h).$$

Remark: This approximation is exact if f is a polynomial of degree at most 1.

Centered finite differences

Consider that approximation

$$f'(x_j) \approx \frac{f_{j+1} - f_{j-1}}{2h}.$$

Note that this is a 3-point formula (x_{j+1}, x_j, x_{j-1}).
Error analysis:

$$f(x_j + h) = f(x_j) + hf'(x_j) + h^2 f''(x_j)/2 + h^3 f'''(\xi_1)/6.$$

$$f(x_j - h) = f(x_j) - hf'(x_j) + h^2 f''(x_j)/2 - h^3 f'''(\xi_2)/6.$$

Hence

$$\frac{f_{j+1} - f_{j-1}}{2h} = f'(x_j) + \frac{h^2}{12}(f'''(\xi_1) + f'''(\xi_2)).$$

and

$$\left| \frac{f_{j+1} - f_{j-1}}{2h} - f'(x_j) \right| = O(h^2).$$

(exact for polynomials of degree at most 2)

High-order finite differences

Using more points we can get more accurate approximations:

$$f'(x_j) \approx \frac{a_0 f_{j+2} + a_1 f_{j+1} + a_2 f_j + a_3 f_{j-1} + a_4 f_{j-2}}{h}$$

We can use Taylor expansions to find coefficients such that the error is $O(h^4)$ and the formula is exact to polynomials of degree 4.

Work this out in class to get the formula below:

$$f'(x_j) \approx \frac{1}{h}\left(-\frac{5}{60} f_{j+2} + \frac{2}{3} f_{j+1} - \frac{2}{3} f_{j-1} + \frac{5}{60} f_{j-2}\right)$$

Remark: higher accuracy requires more function values (larger stencil size). In this case we are using a 5-point stencil.

Numerical experiments

Matlab: loglog and semilogy plots.

Second derivative formulas:

Finite difference formulas for higher derivatives can be derived the same way:

$$f''(x_j) \approx \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}, \quad O(h^2)$$

Sided finite differences:

$$f''(x_j) \approx \frac{a_j f_j + a_{j-1} f_{j-1} + a_{j-2} f_{j-2}}{h^2}$$

Find the weights in class ...

Second derivative formulas:

Finite difference formulas for higher derivatives can be derived the same way:

$$f''(x_j) \approx \frac{f_{j+1} - 2f_j + f_{j-1}}{h^2}, \quad O(h^2)$$

Sided finite differences:

$$f''(x_j) \approx \frac{a_j f_j + a_{j-1} f_{j-1} + a_{j-2} f_{j-2}}{h^2}$$

Find the weights in class ...

Remark: centered FD formulas are more accurate than sided FD formulas.

Polynomial interpolation

FD formulas can also be derived and analyzed using polynomial interpolation.

Example

Consider the line $y = mx + b$ through the points (x_j, f_j) and (x_{j+1}, f_{j+1})

$$m = \frac{f_{j+1} - f_j}{x_{j+1} - x_j}$$

$$y' = m = \frac{f_{j+1} - f_j}{h}$$

Polynomial interpolation

FD formulas can also be derived and analyzed using polynomial interpolation.

Example

Now consider the parabola $y = ax^2 + bx + c$ through the points $(-h, f_{-1})$, $(0, f_0)$, and (h, f_1).

y' at $x = 0$ is given by b.

Polynomial interpolation

FD formulas can also be derived and analyzed using polynomial interpolation.

Example

Now consider the parabola $y = ax^2 + bx + c$ through the points $(-h, f_{-1})$, $(0, f_0)$, and (h, f_1).

y' at $x = 0$ is given by b.

We have that $c = f_0$ and

$$
\begin{aligned}
f_1 &= ah^2 + bh + f_0 \qquad (1) \\
f_{-1} &= ah^2 - bh + f_0 \qquad (2)
\end{aligned}
$$

Polynomial interpolation

FD formulas can also be derived and analyzed using polynomial interpolation.

Example

Now consider the parabola $y = ax^2 + bx + c$ through the points $(-h, f_{-1})$, $(0, f_0)$, and (h, f_1).

y' at $x = 0$ is given by b.

We have that $c = f_0$ and

$$\begin{aligned} f_1 &= ah^2 + bh + f_0 && (1) \\ f_{-1} &= ah^2 - bh + f_0 && (2) \end{aligned}$$

Hence

$$b = \frac{f_1 - f_{-1}}{2h}$$

Polynomial interpolation

FD formulas can also be derived and analyzed using polynomial interpolation.

Example

Now consider the parabola $y = ax^2 + bx + c$ through the points $(-h, f_{-1})$, $(0, f_0)$, and (h, f_1).

We also have that y'' at $x = 0$ is given by $2a$.

We have that $c = f_0$ and

$$\begin{aligned} f_1 &= ah^2 + bh + f_0 && (3) \\ f_{-1} &= ah^2 - bh + f_0 && (4) \end{aligned}$$

Hence

$$2ah^2 = f_1 + f_{-1} - 2f_0$$

and

$$y''(0) = 2a = \frac{f_1 - 2f_0 + f_{-1}}{h^2}$$

High-order polynomial interpolation

We will now focus on the accuracy of high-order polynomial interpolants.

See `interp_demo.m`

The Vandermonde matrix

https://en.wikipedia.org/wiki/Vandermonde_matrix

Consider a polynomial written using the **monomial** basis $\{1, x, x^2, \dots\}$:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots a_n x^n.$$

The interpolation condition is

$$\begin{aligned}
f_0 &= a_0 + a_1 x_0 + a_2 x_0^2 + a_3 x_0^3 + \dots a_n x_0^n && (5)\\
f_1 &= a_0 + a_1 x_1 + a_2 x_1^2 + a_3 x_1^3 + \dots a_n x_1^n && (6)\\
f_2 &= a_0 + a_1 x_2 + a_2 x_2^2 + a_3 x_2^3 + \dots a_n x_2^n && (7)\\
\vdots &\quad \vdots \quad \vdots && (8)\\
f_n &= a_0 + a_1 x_n + a_2 x_n^2 + a_3 x_n^3 + \dots a_n x_n^n && (9)
\end{aligned}$$

The Vandermonde matrix

https://en.wikipedia.org/wiki/Vandermonde_matrix
Consider a polynomial written using the **monomial** basis $\{1, x, x^2, \dots\}$:

$$p_n(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \dots a_n x^n.$$

Or in matrix form

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}$$

This system is expensive to invert, that is $O(n^3)$, and ill-conditioned.

There are better ways to find the polynomial interpolant

Barycentric Lagrange Interpolation

Reference:
J-P Berrut, L.N. Trefethen, Barycentric Lagrange Interpolation, SIAM Review, Vol. 46, 2004.
`https://people.maths.ox.ac.uk/trefethen/barycentric.pdf`

The Lagrange form:

$$p_n(x) = \sum_{j=0}^{n} \ell_j(x) f_j,$$

where

$$\ell_j(x) = \left(\prod_{k=0, k\neq j}^{n} (x - x_k) \right) \Big/ \left(\prod_{k=0, k\neq j}^{n} (x_j - x_k) \right)$$

Barycentric Lagrange Interpolation

The Lagrange form:

$$p_n(x) = \sum_{j=0}^{n} \ell_j(x) f_j,$$

where

$$\ell_j(x) = \left(\prod_{k=0, k\neq j}^{n} (x - x_k) \right) \Big/ \left(\prod_{k=0, k\neq j}^{n} (x_j - x_k) \right)$$

ℓ_j are polynomials of degree n called **Cardinal or Lagrange functions**.

$$\ell_j(x_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} .$$

Hence, p_n is a polynomial of degree n and $p(x_j) = f_j$.

Barycentric Lagrange Interpolation

Notice that the Lagrange form requires $O(n^2)$ operations for each evaluation. In derive a better formula, let

$$\ell(x) = (x - x_0)(x - x_1)\dots(x - x_n).$$

and

$$w_j = 1/\left(\prod_{k=0,k\neq j}^{n}(x_j - x_k)\right).$$

Hence

$$\ell_j(x) = \ell(x)\frac{w_j}{(x - x_j)}$$

and

$$p_n(x) = \ell(x)\sum_{j=0}^{n}\frac{w_j}{(x - x_j)}f_j.$$

Barycentric Lagrange Interpolation

$$p_n(x) = \ell(x) \sum_{j=0}^{n} \frac{w_j}{(x - x_j)} f_j.$$

Notice that

$$1 = \ell(x) \sum_{j=0}^{n} \frac{w_j}{(x - x_j)}$$

and

$$\ell(x) = 1 \Big/ \sum_{j=0}^{n} \frac{w_j}{(x - x_j)}.$$

Therefore

$$p_n(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(x - x_j)} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(x - x_j)}}, \quad \text{(Barycentric formula).}$$

Barycentric Lagrange Interpolation

$$p_n(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(x - x_j)} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{(x - x_j)}}, \quad \text{(Barycentric formula)}.$$

Notice that the wieghts

$$w_j = 1 \Big/ \left(\prod_{k=0, k \neq j}^{n} (x_j - x_k) \right).$$

require $O(n^2)$ operations to compute. Once they are computed, or if they are known, the interpolant is evaluated with $O(n)$ operations.

Barycentric Weights

- Equidistant nodes on $[-1, 1]$, $h = 2/n$.

$$w_j = (-1)^j \binom{n}{j}$$

- Chebyshev points of the first kind:

$$x_j = \cos \frac{(2j+1)\pi}{2n+2}, \quad j = 0 \dots n.$$

$$w_j = (-1)^j \sin \frac{(2j+1)\pi}{2n+2}$$

- Chebyshev points of the second kind:

$$x_j = \cos \frac{j\pi}{n}, \quad j = 0 \dots n.$$

$$w_0 = \frac{1}{2}, \quad w_n = (-1)^n \frac{1}{2},$$

$$w_j = (-1)^j, \quad j = 1 \dots n-1.$$

The chebfun system

`http://www.chebfun.org`

- chebpts.m
- baryWeights.m
- bary.m

matlab experiments

Differentiation and high-order FD formulas

Recall that

$$p_n(x) = \sum_{j=0}^{n} \ell_j(x) f_j \Rightarrow p_n'(x_i) = \sum_{j=0}^{n} \ell_j'(x_i) f_j$$

Therefore, $\ell_j'(x_i)$ are the FD weights.
It is easy to show that (see paper by Berrut and Trefethen)

$$\ell_j'(x_i) = \frac{w_j}{w_i} \frac{1}{x_i - x_j}, \quad i \neq j.$$

and

$$\ell_i'(x_i) = -\sum_{i \neq j} \ell_j'(x_i).$$

Differentiation and high-order FD formulas

Equispaced nodes:

$$w_j = (-1)^j \binom{n}{j}$$

See Pascal's triangle:
`https://en.wikipedia.org/wiki/Pascal's_triangle`

Example

The 3-point centered FD weights:

$$w_0 = 1, \quad w_1 = -2, \quad w_2 = 1$$

Therefore

$$\ell_0'(x_1) = -\frac{1}{2h}, \quad \ell_2'(x_1) = \frac{1}{2h}, \quad \ell_1'(x_1) = 0.$$

$$p'(x_1) = \frac{f_2 - f_0}{2h}.$$

For larger stencils see `fdw.m`

High-order FD formulas

Reference for high-order FD weights:
"A review of pseudospectral methods", by Fornberg and Sloan (available on blackboard)
Remarks:

- In the limit $n \to \infty$, centered FD weights converge to

$$c_j = \begin{cases} \frac{(-1)^{j+1}}{j}, & j = \pm 1, \pm 2, \dots \\ 0, & j = 0. \end{cases}$$

- The situation is very different for one-sided approximations. The magnitude of some weights grow like

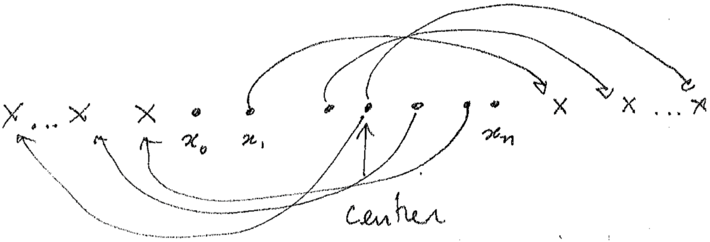
$$\sim \pi^{-1/2} n^{-3/2} 2^{n+3/2}$$

(see `bar_plot_fdweights.m`)

How to deal with boundary points?

- Assume periodic Boundary Conditions.
- Use clustered nodes near boundary (Chebyshev points).

Periodic BC



In this case we can use centered FD formulas at each point in the domain! See `Spectral_FD_periodic.m`

Chebyshev Polynomials

$$\begin{aligned} T_n(x) &= \cos(n \arccos\ x) \qquad -1 \leq x \leq 1. \\ T_0(x) &= 1 \\ T_1(x) &= x \end{aligned}$$

Chebyshev Polynomials

$$\begin{aligned} T_n(x) &= \cos(n \arccos\ x) \quad -1 \leq x \leq 1. \\ T_0(x) &= 1 \\ T_1(x) &= x \end{aligned}$$

Recurrence: Let $\theta = \arccos\ x$, $x = \cos\theta$.

$$\begin{aligned} \cos((n+1)\theta) &= \cos(n\theta)\cos(\theta) - \sin(n\theta)\sin(\theta) \\ \cos((n-1)\theta) &= \cos(n\theta)\cos(\theta) + \sin(n\theta)\sin(\theta) \end{aligned}$$

Hence:

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x).$$

or

Chebyshev Polynomials

$$\begin{aligned} T_n(x) &= \cos(n \arccos\ x) \quad -1 \leq x \leq 1. \\ T_0(x) &= 1 \\ T_1(x) &= x \end{aligned}$$

Recurrence: Let $\theta = \arccos\ x$, $x = \cos\theta$.

$$\begin{aligned} \cos((n+1)\theta) &= \cos(n\theta)\cos(\theta) - \sin(n\theta)\sin(\theta) \\ \cos((n-1)\theta) &= \cos(n\theta)\cos(\theta) + \sin(n\theta)\sin(\theta) \end{aligned}$$

Hence:

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x).$$

or

$$\begin{aligned} T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \\ T_0(x) &= 1, \\ T_1(x) &= x. \end{aligned}$$

Chebyshev Polynomials – properties

$$
\begin{aligned}
T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x), \\
T_0(x) &= 1, \\
T_1(x) &= x.
\end{aligned}
$$

Theorem

- *T_n is a polynomial of degree n.*
- *T_n is even (odd) if n is even (odd).*
- $T_n(x) = 2^{n-1}x^n + C_{n-1}x^{n-1} + \cdots + C_0, \quad n = 1, 2, \ldots.$

Proof by induction (use the recurrence formula).

(Plot polynomials with Matlab)

Chebyshev Polynomials – roots (first kind)

Roots:

$$x_j = \cos\left(\frac{2j+1}{2(n+1)}\pi\right), \quad j = 0, \dots n.$$

Notice that

$$T_{n+1}(x_j) = \cos((n+1)\arccos x_j) = \cos\left(\frac{2j+1}{2}\pi\right) = 0.$$

The nodes $\{x_j\}$ are called Chebyshev points of the <u>first kind</u>.

Chebyshev Polynomials – roots (second kind)

Chebyshev points of the second kind are roots of Chebyshev polynomials of the second kind:

$$U_n(x) = \frac{\sin((n+1)\arccos x)}{\sin(\arccos x)}, \quad -1 \leq x \leq 1.$$

Notice that

$$\frac{d}{dx} T_n(x) = U_{n-1}(x).$$

Hence U_n must be a polynomial as well.

Chebyshev points of the second kind:

$$x_j = cos\left(\frac{j\pi}{n}\right), \quad j = 0 \dots n.$$

In this case,

$$U_{n-1}(x_j) = 0, \quad j = 1, \dots, n-1.$$

Chebyshev points of the second kind:

Also, notice that

$$T_n(x_j) = \cos(j\pi) = \pm 1.$$

That is, the points x_j give that max/min values of T_n.

Second kind are preferable for solving PDEs because of boundary conditions!

Inner Products and Norms

Inner Product

$$\langle \cdot, \cdot \rangle : V \times V \to C$$

1. $\langle x, y \rangle = \overline{\langle y, x \rangle}$
   (Conjugate Symmetry)
2. $\langle ax, y \rangle = a\langle x, y \rangle$
   $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
   (Linearity)
3. $\langle x, x \rangle \geq 0$ and $\langle x, x \rangle = 0 \iff x = 0$
   (Positive-definiteness)

Examples of Inner Products

ℓ_2 inner product for vectors in $\mathbb{R}^n$

$$\mathbf{x} = [x_1, x_2, \dots x_n]^T, \quad \mathbf{y} = [y_1, y_2, \dots, y_n]^T,$$

$$\langle x, y \rangle = y^* x = x_1 \overline{y_1} + \cdots + x_n \overline{y_n}$$

Weighted ℓ_2 inner products for vectors in $\mathbb{R}^n$

$$\langle x, y \rangle_w = x_1 \overline{y_1} w_1 + \cdots + x_n \overline{y_n} w_n$$

$$w_k > 0$$

L_2 inner product for functions ($L_2[a, b]$)

$$< f, g >= \int_a^b f(x) \overline{g(x)} dx$$

$$< f, g >_w = \int_a^b f(x) \overline{g(x)} w(x) dx, \quad w(x) > 0.$$

Norms and Normed Vector Spaces

Requirements for a norm:

1. $\|f\| \geq 0$ if $f \neq 0$
2. $\|\alpha f\| = |\alpha| \|f\|$
3. $\|f + g\| \leq \|f\| + \|g\|$

Examples (here **x** is a vector):

$$\|\mathbf{x}\|_p = \left(\sum_k |x_k|^p \right)^{1/p}, \quad p \geq 1.$$

$$\|\mathbf{x}\|_1 = \sum_k |x_k|$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_k |x_k|^2}$$

$$\|\mathbf{x}\|_\infty = \max_k |x_k|$$

Norms and Normed Vector Spaces

Examples (here f is function defined on $[a, b]$):

$$\|f\|_p = \left(\int_a^b |f(x)|^p dx \right)^{1/p}, \quad p \geq 1.$$

$$\|f\|_1 = \int_a^b |f(x)| dx$$

$$\|f\|_2 = \sqrt{\int_a^b |f(x)|^2 dx}$$

$$\|f\|_\infty = \max_{x \in [a,b]} |f(x)|$$

Norms and angles induced by inner-products

Given an inner-product defined on a vector space V, define

$$\|x\| := \sqrt{\langle x, x \rangle}$$

and

$$\cos\theta := \frac{\langle x, y \rangle}{\|x\| \|y\|}$$

Examples of induced norms:

$$\ell_2 : \quad \|x\|^2 = \sum_k |x_k|^2 = \langle x, x \rangle.$$

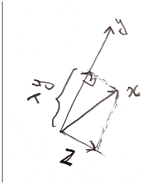
$$L_2[a, b] : \quad \|f\|^2 = \int_a^b |f(x)|^2 dx = \langle f, f \rangle.$$

Cauchy-Schwarz Inequality

$$|\langle x, y\rangle| \leq \|x\|\|y\|$$

Recall that $\langle x, y\rangle = \|x\|\|y\|\cos\theta$. But from our definition of θ, it is not obvious that $|\cos\theta| \leq 1$.

Proof:
Let $y \neq 0$, $\lambda = \frac{\langle x,y\rangle}{\|y\|^2}$, and $z = x - \lambda y$.



Then,

$$\langle z, y\rangle = \langle x - \lambda y, y\rangle = \langle x, y\rangle - \lambda\langle y, y\rangle = 0.$$

Cauchy-Schwarz Inequality

Therefore

$$\begin{aligned} \|x\|^2 &= \langle z+\lambda y, z+\lambda y\rangle = \langle z, z+\lambda y\rangle + \lambda\langle y, z+\lambda y\rangle \\ &= \langle z,z\rangle + \overline{\lambda}\langle z,y\rangle + \lambda\langle y,z\rangle + |\lambda|^2\langle y,y\rangle \\ &= \|z\|^2 + |\lambda|^2\|y\|^2 \geq |\lambda|^2\|y\|^2 = \frac{|\langle x,y\rangle|^2}{\|y\|^2}. \end{aligned}$$

Hence,

$$\|x\|^2\|y\|^2 \geq |\langle x,y\rangle|^2$$

Remark: The equality holds if $z = 0$, that is, $x = \alpha y$ (y is parallel to x).

The definition of angle using $\cos\theta = \langle x,y\rangle/\|x\|\|y\|$ seems reasonable now!
That is,

$$\theta = \pi/2 \quad \text{if } \langle x,y\rangle = 0.$$

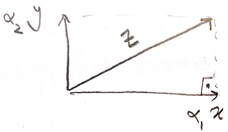
$$\theta = 0 \ \text{ or } \ \pi \quad \text{if } \ x = \alpha y.$$

Orthogonality (Pythagorean Theorem)

Suppose

$$\langle x, y \rangle = 0$$

and $z = \alpha_1 x + \alpha_2 y$.



Then,

$$\begin{aligned} \|z\|^2 &= \langle \alpha_1 x + \alpha_2 y, \alpha_1 x + \alpha_2 y \rangle \\ &= \alpha_1^2 \|x\|^2 + \alpha_2 \overline{\alpha_1} \langle y, x \rangle + \alpha_1 \overline{\alpha_2} \langle x, y \rangle + |\alpha_2|^2 \|y\|^2. \end{aligned}$$

Hence,

$$\|z\|^2 = |\alpha_1|^2 \|x\|^2 + |\alpha_2|^2 \|y\|^2$$

In particular, if $\|x\| = 1$ and $\|y\| = 1$,

$$\|z\|^2 = |\alpha_1|^2 + |\alpha_2|^2$$

Orthogonality (Pythagorean Theorem), Remarks

Recall from Linear Algebra that if Q is an unitary matrix and $z = Qx$, then $\|z\| = \|x\|$.

For functions, suppose

$$f(x) = \sum_k \lambda_k \phi_k(x), \quad \langle \phi_j, \phi_k \rangle = 0, \ \ j \neq k.$$

Then

$$\|f\|^2 = \sum_k |\lambda_k|^2 \|\phi_k\|^2.$$

If in addition, $\|\phi_k\| = 1$,

$$\|f\|^2 = \sum_k |\lambda_k|^2.$$

In particular,

$$|\lambda_k| \leq \|f\|. \quad \text{(expansion coefficients are well behaved)}$$

Change of Basis

Example in matlab of coefficients in monomial and Chebyshev expansions.

Change of Basis Let $\{a_1, a_2, \ldots, a_n\}$ and a set of linearly independent vectors or functions. Then there exists an orthogonal set $\{q_1, q_2, \ldots, q_n\}$ such that $span\{a_k\} = span\{q_k\}$.

Proof: Gram–Schmidt orthogonalization.

Example: Monomials to Legendre using QR orthogonalization in Matlab.

Examples of orthogonal bases

Fourier basis:

$$\{e^{ikx}\}, \quad \langle f, g \rangle = \int_0^{2\pi} f(x)\overline{g(x)}dx.$$

Legendre basis:

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

Chebyshev polynomials (first kind):

$$\{T_k(x)\}, \quad \langle f, g \rangle = \int_{-1}^{1} f(x)\overline{g(x)}\frac{1}{\sqrt{1-x^2}}dx.$$

Error in polynomial approximation

Theorem (Weierstrass)

If f is a given continuous function for $a \leq x \leq b$, and if ϵ is an arbitrary positive quantity, it is possible to construct an approximating polynomial P such that

$$|f(x) - P(x)| < \epsilon$$

for $a \leq x \leq b$.

Proof: `http://www.math.univ-toulouse.fr/~lassere/pdf/2012INPsuitesD1bis.pdf`

Computing polynomial approximations

- Polynomial interpolation
- Polynomial least-squares (discrete or continuous)
  (optimal w.r.t the L_2 norm)
- Remez algorithm (not used in practice)
  (optimal w.r.t the L_∞ norm)
- etc

Remark: Both continuous least-squares and the Remez algorithm require knowledge of f everywhere on the interval $[a, b]$.

Interpolation and discrete least-squares approximate f from $\{(x_j, f(x_j))\}$, $j = 0, \ldots, n$.

Cauchy Interpolation Error

Theorem

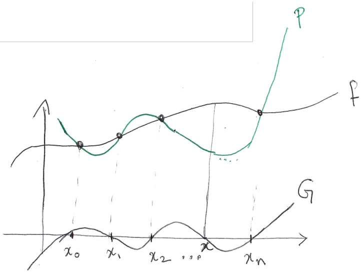
Let x_i, $i = 0, \dots, n$ be distinct points and let p be the Lagrange interpolant of degree at most n of some function f at the points x_i. Let $x \in \mathbb{R}$ and suppose that $f \in C^{n+1}(J)$, for some interval J containing x_i and x. Then there exists a point ξ in the interior of J such that

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{n+1}(\xi)\ell(x),$$

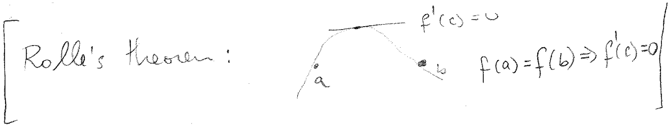
where $\ell(x) = (x - x_0)(x - x_1)\dots(x - x_n)$.

Cauchy Interpolation Error – Proof

Proof: Let $G(t) = [f(x) - p(x)]\ell(t) - [f(t) - p(t)]\ell(x)$.
G has (at least) $n+2$ distinct zeros: x_i and x.



Recall that if $f(a) = f(b)$ then $f'(c) = 0$ for some $c \in (a, b)$.



Hence $G^{(n+1)}(\xi) = 0$ for some $\xi \in (a, b)$. Morever, $P^{(n+1)} = 0$, and $\ell^{(n+1)} = (n+1)!$. Therefore

$$0 = [f(x) - p(x)](n+1)! - [f^{(n+1)}(\xi) - 0]\ell(x),$$

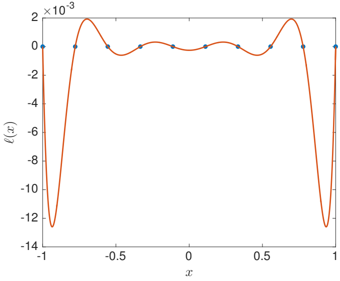
Cauchy Interpolation Error – Proof

$$0 = [f(x) - p(x)](n+1)! - [f^{(n+1)}(\xi) - 0]\ell(x),$$

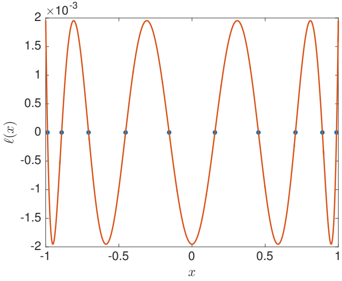
$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)\ell(x).$$

How small can we make $|\ell(x)|$ by choosing good interpolation points x_j?

equispaced points



Chebyshev points



Optimality of Chebyshev points

On the interval $[-1, 1]$, we have the following:

- If $x_0, x_1, \ldots, x_n$ are Chebyshev points of the first kind,

$$\max_{-1 \le x \le 1} |\ell(x)| = 2^{-n}.$$

- For any other set of interpolation points,

$$\max_{-1 \le x \le 1} |\ell(x)| \ge 2^{-n}.$$

Proof: (first part)
Recall that $T_n(x) = 2^{n-1}x^n + C_{n-1}x^{n-1} + \cdots + C_0$. Hence

$$\cos((n+1)\arccos x) = 2^n(x - x_0)(x - x_1)\ldots(x - x_n).$$

or

$$|(x - x_0)(x - x_1)\ldots(x - x_n)| = 2^{-n}|\cos((n+1)\arccos x)|.$$

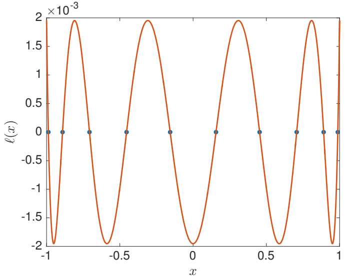
Optimality of Chebyshev points

- For any other set of interpolation points, $\max_{-1\le x\le 1} |\ell(x)| \ge 2^{-n}$.

Proof: (second part by contradiction)
Assume $\max_{-1\le x\le 1} |\tilde{\ell}(x)| < 2^{-n}$.

$$\begin{aligned} \ell(x) &= (x-x_0)(x-x_1)\dots(x-x_n) && \text{Chebyshev nodes} \\ \tilde{\ell}(x) &= (x-\tilde{x}_0)(x-\tilde{x}_1)\dots(x-\tilde{x}_n) && \text{other nodes} \end{aligned}$$

Hence $\ell(x)-\tilde{\ell}(x)$ is a polynomial of degree n with $n+1$ roots ?!



Example of error bound in interpolation at Cehbyshev points

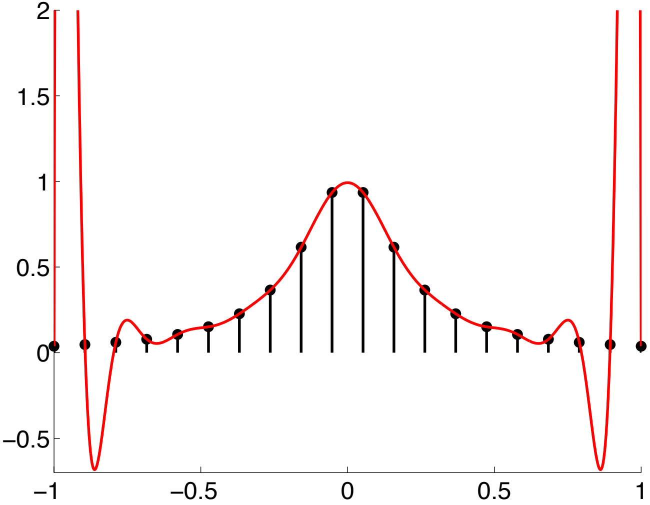
Let $f(x) = e^x$, then

$$|f(x) - p_n(x)| \leq \frac{1}{(n+1)!}\ e\ 2^{-n}$$

```
>> n = 14; 1/factorial(n+1)*exp(1)*2^-n
ans =
     1.268746717007866e-16
```

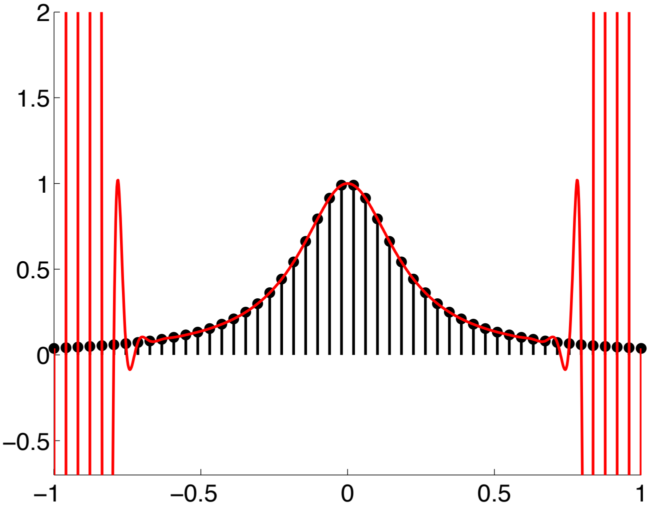
By contrast, using linear splines to approximate this function to the same accuracy we need about 10^8 data points. And cubic splines would require about 10^5 data points.

Polynomial interpolation on equispaced nodes



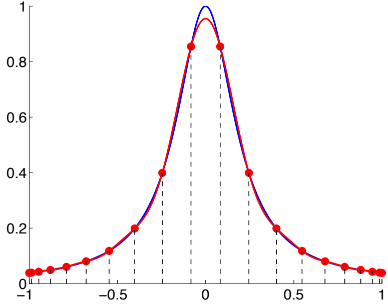
Analytic approximation but looks bad!

The Runge phenomenon – Polynomial interpolation



Analytic approximation but looks bad! (approximation diverges)

Polynomial interpolation on Chebyshev nodes

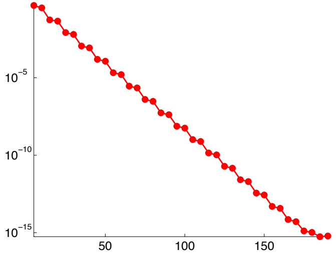


Polynomial interpolation on Chebyshev notes

Error decays exponentially fast

$$O(\exp(-aN))$$

Geometric convergence



Convergence of Chebyshev polynomial interpolation

Theorem

Let f be a continuous function on $[-1,1]$, p_n its degree n polynomial interpolant in Chebyshev points (first or second kind). Then

- *if f has a kth derivative in $[-1,1]$ of bounded variation for some $k \geq 1$,*

$$\|f - p_n\|_\infty = O(n^{-k}), \quad \text{as} \quad n \to \infty$$

- *if f is analytic in a neighborhood of $[-1,1]$,*

$$\|f - p_n\|_\infty = O(C^n), \quad \text{as} \quad n \to \infty$$

for some $C < 1$; in particular we may take $C = 1/(M+m)$ is f is analytic in the closed ellipse with foci ± 1 and semimajor and seminor axis lengths $M \geq 1$ and $m \geq 0$.

Convergence of Chebyshev polynomial interpolation

Reference: Z. Battles and L.N. Trefethen, An extension of Matlab to continuous functions and operators, SIAM J. Sci. Comp. 25 (2004), 1743-1770.

Proof: take APM524 (Spectral Methods).

Examples:

$$f(x) = |x|^3$$

Here $f'''(x) = \pm 6$ and $f^{(4)}(x) = 6\delta(x)$. Hence f''' has a derivative of bounded variation. That is,

$$\int_{-1}^{1} |f^{(4)}(x)| dx < M$$

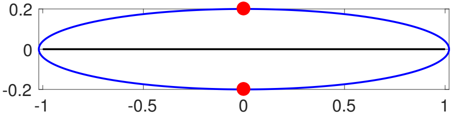
Hence,

$$\|p_n - f\|_\infty = O(n^{-3})$$

Convergence of Chebyshev polynomial interpolation

Example:

$$f(x) = 1/(1+25x^2)$$



In this case we take

$$m = .2$$

and

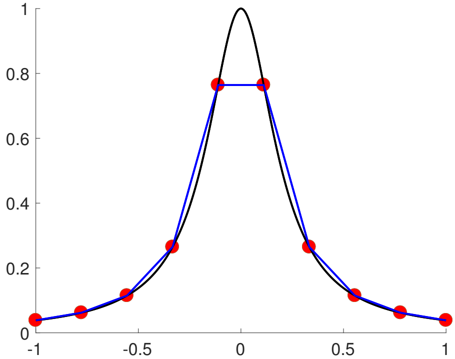
$$M = \sqrt{1+.2^2}$$

Finally

$$\|p_n - f\|_\infty = O\left(\frac{1}{(M+m)^n}\right)$$

Piecewise Polynomial Interpolation (splines)

Linear splines: piecewise linear interpolation

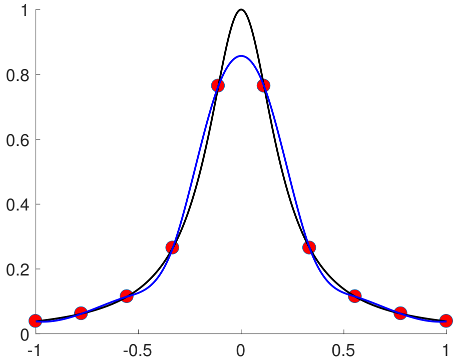


Using the Cauchy error formula, we can show that

$$\|f - s\|_\infty = O(h^2)$$

Piecewise Polynomial Interpolation (splines)

Cubic splines: piecewise cubic interpolation



Using the Cauchy error formula, we can show that

$$\|f - s\|_\infty = O(h^4)$$