FWAM Session B: Function Approximation and Differential Equations

Alex Barnett[1] and **Keaton Burns**[1,2]

Wed, 10/30/19

[1]Center for Computational Mathematics, Flatiron Institute
[2]Center for Computational Mathematics, Flatiron Institute, and Department of Mathematics, MIT

LECTURE 1: interpolation, integration, spectral methods

Motivations

exact func. $f(x)$ described by ∞ number of points

how handle approximately (but accurately) in computer, using least cost (bytes)?

- Interpolation: cheap but accurate look-up table for expensive $f(x)$
  data fitting: given non-noisy data $f(x_i)$ at some x_i, model $f(x)$ at other points x?

  Contrast: fit noisy data = learning (pdf for) params in model, via likelihood/prior
- (Numerical) integration:
  eg computing expectation values given a pdf

  Contrast: Monte Carlo (random, high-dim.) integration, Thurs am
- Differentiation:
  get gradient ∇f in order to optimize or
- Spectral (often Fourier) methods:
  If $f(x)$ is smooth, handle very accurately without much extra cost

Deterministic (non-random) methods.

Integr/diff crucial for numerical ODEs and PDEs topic of LECTURE II

Goals LECTURE I

TODO

teach range of practical methods focusing on 1D

pointers to dimensions $d > 1$

concepts:

convergence order

spectral methods

local vs global

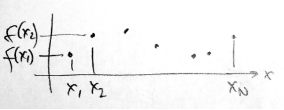
adaptivity

rounding error

interpolation $=$ func. representation, key to all else

Interpolation in 1D ($d = 1$)

Say $y_j = f(x_j)$ known at nodes $\{x_j\}$ N-pt "grid"
exact data, not noisy
want interpolant $\tilde{f}(x)$, s.t. $\tilde{f}(\mathbf{x}_j) = y_j$

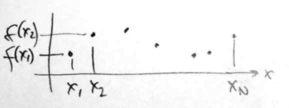


Interpolation in 1D ($d = 1$)

Say $y_j = f(x_j)$ known at nodes $\{x_j\}$ N-pt "grid"

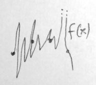
exact data, not noisy

want interpolant $\tilde{f}(x)$, s.t. $\tilde{f}(\mathbf{x}_j) = y_j$



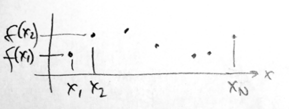
hopeless w/o assumptions on f, eg smoothness, otherwise...

- extra info helps, eg f periodic, or $f(x) = \text{smooth} \cdot |x|^{-1/2}$



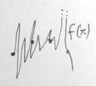
Interpolation in 1D ($d = 1$)

Say $y_j = f(x_j)$ known at nodes $\{x_j\}$ N-pt "grid"
exact data, not noisy
want interpolant $\tilde{f}(x)$, s.t. $\tilde{f}(\mathbf{x}_j) = y_j$



hopeless w/o assumptions on f, eg smoothness, otherwise...

- extra info helps, eg f periodic, or $f(x) = \text{smooth} \cdot |x|^{-1/2}$

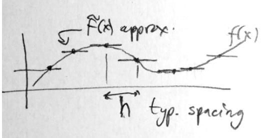


Simplest: use value at x_j nearest to x

"snap to grid"

Error $\max_x |\tilde{f}(x) - f(x)| = \mathcal{O}(h)$ as $h \to 0$

holds if f' bounded; can be nonsmooth but not crazy



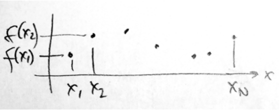
Recall notation "$\mathcal{O}(h)$": exists $C, h_0 > 0$ s.t. error $\leq Ch$ for all $h < h_0$

Interpolation in 1D ($d = 1$)

Say $y_j = f(x_j)$ known at nodes $\{x_j\}$ $\quad$ N-pt "grid"

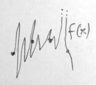
exact data, not noisy

want interpolant $\tilde{f}(x)$, s.t. $\tilde{f}(\mathbf{x}_j) = y_j$



hopeless w/o assumptions on f, eg smoothness, otherwise...

- extra info helps, eg f periodic, or $f(x) = \text{smooth} \cdot |x|^{-1/2}$

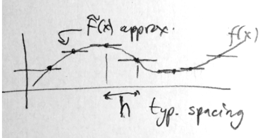


Simplest: use value at x_j nearest to x

"snap to grid"

Error $\max_x |\tilde{f}(x) - f(x)| = \mathcal{O}(h)$ as $h \to 0$

holds if f' bounded; can be nonsmooth but not crazy



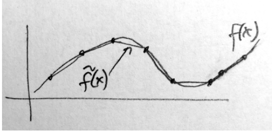
Recall notation "$\mathcal{O}(h)$": exists $C, h_0 > 0$ s.t. error $\leq Ch$ for all $h < h_0$

Piecewise linear:

"connect the dots"

max error $= \mathcal{O}(h^2)$ as $h \to 0$

needs f'' bounded, ie smoother than before



Message: a higher order method is only higher order if f smooth enough

Interlude: convergence rates

Should know or measure convergence rate of any method you use

- "effort" parameter N eg # grid-points $= 1/h^d$ where $h =$ grid spacing, $d =$ dim

We just saw algebraic conv. error $= \mathcal{O}(N^{-p})$, for order $p = 1, 2$

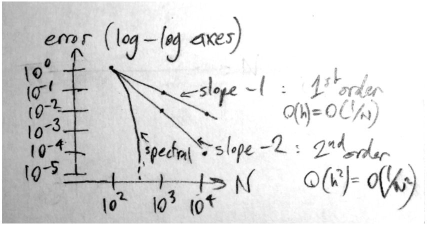
Interlude: convergence rates

Should know or measure convergence rate of any method you use

- "effort" parameter N eg # grid-points $= 1/h^d$ where $h =$ grid spacing, $d =$ dim

We just saw algebraic conv. error $= \mathcal{O}(N^{-p})$, for order $p = 1, 2$

Is only one graph in numerical analysis: $\boxed{\text{"relative error vs effort"}}$



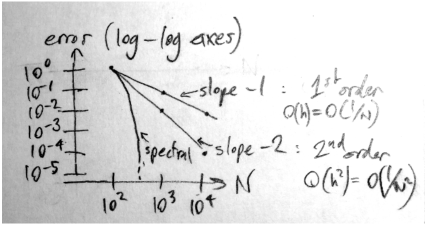
Interlude: convergence rates

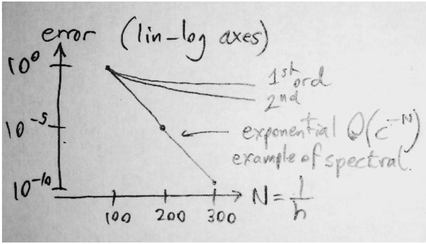
Should know or measure convergence rate of any method you use

- "effort" parameter N eg # grid-points $= 1/h^d$ where $h =$ grid spacing, $d =$ dim

We just saw algebraic conv. error $= \mathcal{O}(N^{-p})$, for order $p = 1, 2$

Is only one graph in numerical analysis: $\boxed{\text{"relative error vs effort"}}$





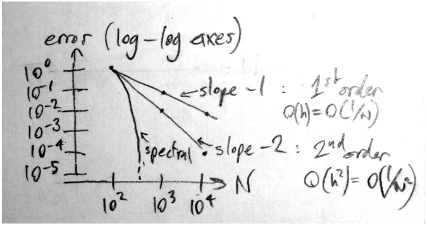
Interlude: convergence rates

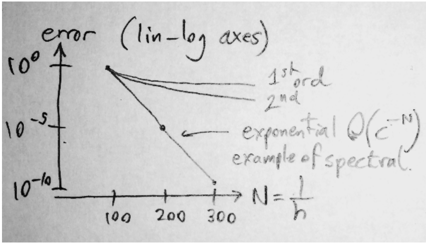
Should know or measure convergence rate of any method you use

- "effort" parameter N eg # grid-points $= 1/h^d$ where $h =$ grid spacing, $d =$ dim

We just saw algebraic conv. error $= \mathcal{O}(N^{-p})$, for order $p = 1, 2$

Is only one graph in numerical analysis: $\boxed{\text{“relative error vs effort”}}$





Note how spectral gets many digits for small N crucial for eg 3D prob.

"spectral" = "superalgebraic", $\mathcal{O}(N^{-k})$ for any k

- how many digits to you want? for 1-digit (10% error), low order ok, easier to code

<rant> test your code w/ *known exact soln* to check error conv. <\rant>

What is the prefactor C in error $\leq Ch^k$? Has asymp. rate even kicked in yet? :)

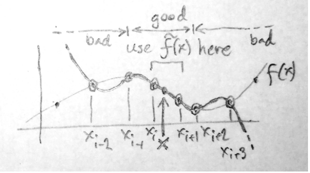
Higher-order interpolation for smooth f: the local idea

For any target x, use only set of nearest p nodes:

Exists unique degree-$(p-1)$ poly, $\sum_{k=0}^{p-1} c_k x^k$ which matches local data $(x_j, y_j)_{j=1}^{p}$

generalizes piecewise lin. idea

do not eval poly outside its central region!



- error $\mathcal{O}(h^k)$, ie high order, but $\tilde{f}$ *not* continuous ($\tilde{f} \notin C$) small jumps

  if must have cont, recommend splines, eg cubic $p = 3$: $\tilde{f} \in C^2$, meaning $\tilde{f}''$ is cont.

FLATIRON INSTITUTE Center for Computational Mathematics

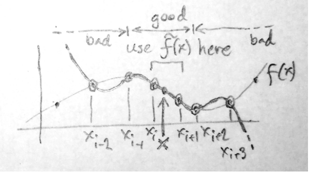
Higher-order interpolation for smooth f: the local idea

For any target x, use only set of nearest p nodes:

Exists unique degree-$(p-1)$ poly, $\sum_{k=0}^{p-1} c_k x^k$
which matches local data $(x_j, y_j)_{j=1}^{p}$

generalizes piecewise lin. idea

do not eval poly outside its central region!



- error $\mathcal{O}(h^k)$, ie high order, but $\tilde{f}$ *not* continuous ($\tilde{f} \notin C$) small jumps

  if must have cont, recommend splines, eg cubic $p = 3$: $\tilde{f} \in C^2$, meaning $\tilde{f}''$ is cont.

How to find the degree-$(k-1)$ poly?

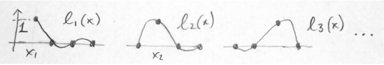
1) Crafty: solve square lin sys for coeffs $\sum_{k<p} x_j^k c_k = y_j$ $\quad j = 1, \dots, p$

   ie $V\mathbf{c} = \mathbf{y}$ $\quad V$="Vandermonde" matrix, is ill-cond. but works

2) Traditional: barycentric formula $\tilde{f}(x) = \dfrac{\sum_{j=1}^{p} \frac{y_j}{x-x_j} w_j}{\sum_{j=1}^{p} \frac{1}{x-x_j} w_j}$ $\quad w_j = \frac{1}{\prod_{i\neq j}(x_j - x_i)}$

   [Tre13, Ch. 5]

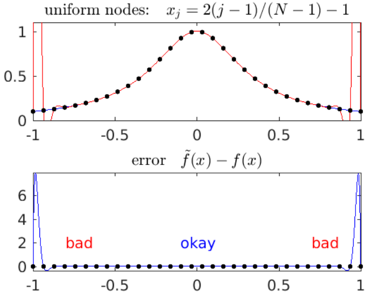
Either way, $\tilde{f}(x) = \sum_{j=1}^{p} y_j \ell_j(x)$ where $\ell_j(x)$ is ℓth Lagrange basis func:



Global polynomial (Lagrange) interpolation?

Want increase order p. Use *all* data, get single $\tilde{f}(x)$, so $p = N$? "global"

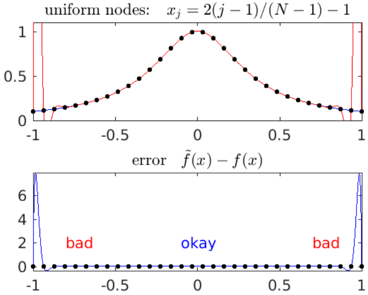
$p = N = 32$, smooth (analytic) $f(x) = \frac{1}{1+9x^2}$ on $[-1, 1]$: (Runge 1901)



Global polynomial (Lagrange) interpolation?

Want increase order p. Use *all* data, get single $\tilde{f}(x)$, so $p = N$? "global"

$p = N = 32$, smooth (analytic) $f(x) = \frac{1}{1+9x^2}$ on $[-1, 1]$: (Runge 1901)

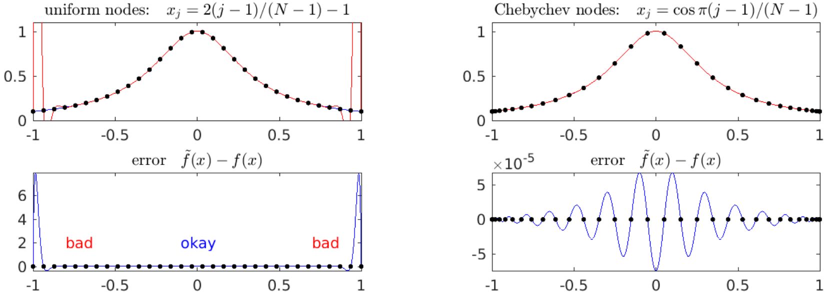


- warning: unif. grid, global interp. fails $\rightarrow$ only use locally in central region

Global polynomial (Lagrange) interpolation?

Want increase order p. Use *all* data, get single $\tilde{f}(x)$, so $p = N$? "global"

$p = N = 32$, smooth (analytic) $f(x) = \frac{1}{1+9x^2}$ on $[-1, 1]$: (Runge 1901)



- warning: unif. grid, global interp. fails → only use locally in central region

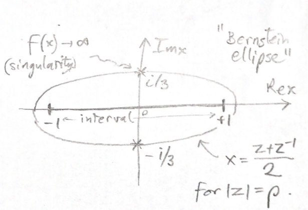
But exists good choice of nodes. . .

"Chebychev": means non-unif. grid density $\sim \frac{1}{\sqrt{1-x^2}}$

- our first spectral method

  max err $= \mathcal{O}(\rho^{-N})$ exponential conv!

  $\rho > 1$ "radius" of largest ellipse in which f analytic



Node choice and adaptivity

Recap poly approx. $f(x)$ on $[a, b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

Question: Do you get to *choose* the set of nodes at which f known?

- No: data fitting applications (or noisy variants: kriging, Gaussian processes, etc)

  use local poly (central region only!), or something stable (splines)
- Yes: almost all else, interp., quadrature, PDE solvers so pick good nodes!

Node choice and adaptivity

Recap poly approx. $f(x)$ on $[a, b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

Question: Do you get to *choose* the set of nodes at which f known?

- No: data fitting applications (or noisy variants: kriging, Gaussian processes, etc)

  use local poly (central region only!), or something stable (splines)

- Yes: almost all else, interp., quadrature, PDE solvers so pick good nodes!

Adaptivity idea global is inefficient if f smooth in most places, structured in a few

Node choice and adaptivity

Recap poly approx. $f(x)$ on $[a,b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

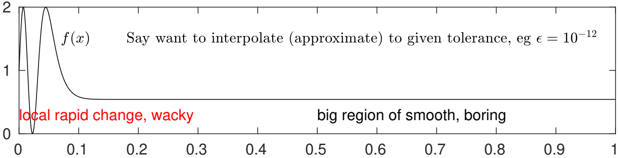
Question: Do you get to *choose* the set of nodes at which f known?

- No: data fitting applications (or noisy variants: kriging, Gaussian processes, etc)

  use local poly (central region only!), or something stable (splines)
- Yes: almost all else, interp., quadrature, PDE solvers so pick good nodes!

Adaptivity idea

global is inefficient if f smooth in most places, structured in a few



Node choice and adaptivity

Recap poly approx. $f(x)$ on $[a, b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

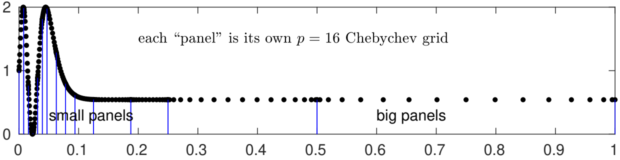
Question: Do you get to *choose* the set of nodes at which f known?

- No: data fitting applications (or noisy variants: kriging, Gaussian processes, etc)

  use local poly (central region only!), or something stable (splines)
- Yes: almost all else, interp., quadrature, PDE solvers so pick good nodes!

Adaptivity idea

global is inefficient if f smooth in most places, structured in a few



automatically split (recursively) panels until max err $\leq \epsilon$

via tests for local error

1D adaptive interpolator codes to try:

- `github/dbstein/function_generator` py+numba, fast (Stein '19)
- `chebfun` for MATLAB big-p Cheb. grids can exploit FFTs! (Trefethen et al.)

App.: replace nasty expensive $f(x)$ by cheap one! optimal "look-up table"

Interpolation of periodic functions

Just did f on intervals $[a, b]$. Interpolation (& integration, etc.) of *periodic* f not same!

Periodic: $f(x + 2\pi) = f(x)$ for all x, $f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

Interpolation of periodic functions

Just did f on intervals $[a, b]$. Interpolation (& integration, etc.) of *periodic* f not same!

Periodic: $f(x + 2\pi) = f(x)$ for all x, $f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

instead of poly's, use truncated series $\tilde{f}(x) = \sum_{|k|<N/2} c_k e^{ikx}$ "trig. poly"

What's best you can do? if $\hat{f}_n$ decays rapidly,

How get c_n? unif. grid now good! Use FFT of vals $f(2\pi j/N)$,

$j = 1, \ldots, N$

$c_n =$

exp conv if f analytic in strip.

what if f not as smooth as this?

relapse to conv. order given by smoothness of f

why? $\hat{f}_n$ decay

Messages: spectral conv usually requires understanding f off the real axis!

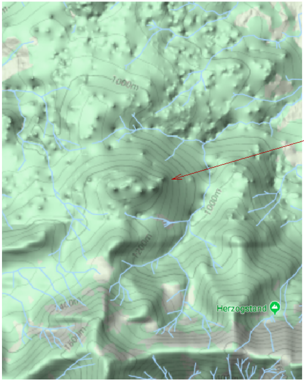
smooth = rapid Fourier series decay = rapid conv

Interpolation in $d > 1$

can you choose the points?
data fit: often no
PDE solve: usually yes - product grids, piecewise (FEM), etc

in $d > 1$ getting $f(\mathbf{x})$ from scattered data $\{\mathbf{x}_i\}$ is hard

Eg google terrain: $f(\mathbf{x})$ rough $\rightarrow$ v low ord



height $f(\mathbf{x})$ interp from unstructured points in 2D, kernel method

pock–marks!

interp from Cartesian grid, more accurate

unless you know f smooth, eg fit local multivariate polynomial
kriging

Numerical integration

Usually the user gets to choose the nodes x_j

Once have interpolant $\tilde{f}$ from data $f(x_j)$, can *integrate it exactly*

"intepolatory quadrature"

Eg: piecewise linear gives composite trap rule $\mathcal{O}(N^{-2})$

periodic spectral gives periodic trap rule $\mathcal{O}(c^{-N})$ if analytic

TO DO

extrapolation

Rounding error [GC12, Ch. 5–6]

LECTURE II: numerical differential equations

For now we start with “elliptic”: time-independent problems

Motivations

eg steady-state (equilibrium) diffusion of a chemical

eg what electric potential caused by bunch of charges surrounded by H_2O ? (protein electrostatics)

Find u solving $\Delta u = f$, $f =$ volume source term

Δ means Laplacian $\partial^2/\partial x^2 + \partial^2/\partial y^2 + \ldots$. Δu is curvature of u

plus some BCs on u

eg viscous fluid flow: $\mathbf{u}$ is velocity field, sat Stokes eqns

eg what is ground state of quantum system, solving $\Delta u = Eu$

Mike will in next talk overview this and 2 other flavors of PDE

References

- A Greenbaum and T P Chartier, *Numerical methods*, Princeton University Press, 2012.
- L. N. Trefethen, *Approximation theory and approximation practice*, SIAM, 2013, `http://www.maths.ox.ac.uk/chebfun/ATAP`.