FLATIRON INSTITUTE
Center for Computational Mathematics

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: for optimization, or get matrix which approx. an ODE/PDE
- 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 how does your accuracy improve vs number of discretization points

global (one expansion formula for the whole domain)

  vs local (different expansions for x in different regions)

spectral method global, often use FFT, converge faster than any fixed order

adaptivity automatically placing degrees of freedom only where they need to be

rounding error

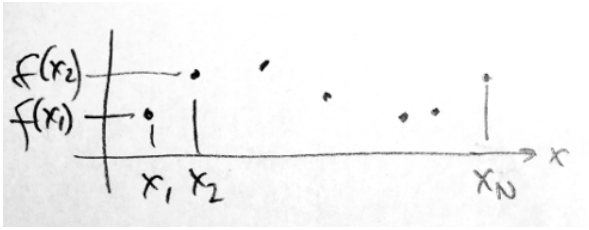
interpolation = func. representation, is key to all else

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. $\tilde{f}(\boldsymbol{x}_j) = y_j$

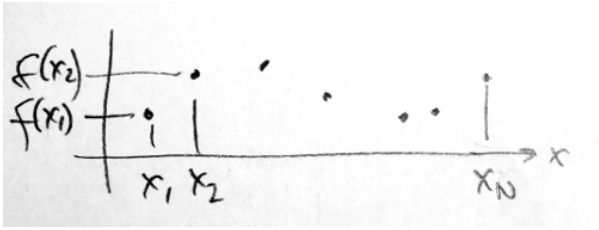


Interpolation in 1D ($d = 1$)

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

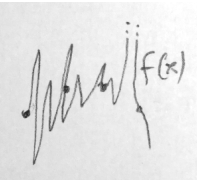
exact data, not noisy

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



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

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

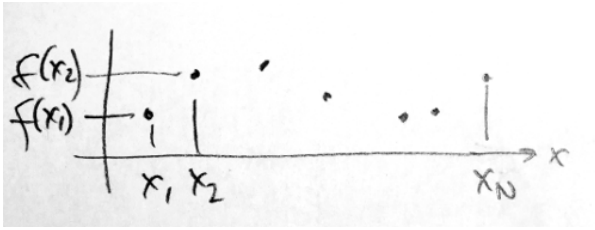


Interpolation in 1D ($d = 1$)

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

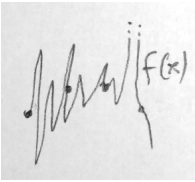
exact data, not noisy

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



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

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

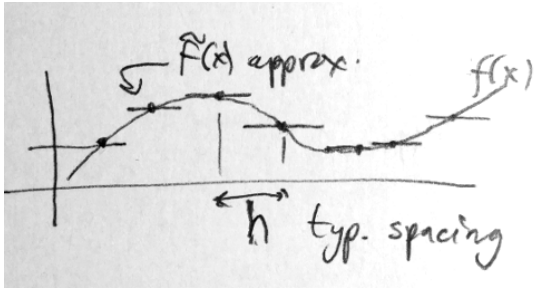


Simplest: use value at x_j nearest to x

"snap to grid"

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

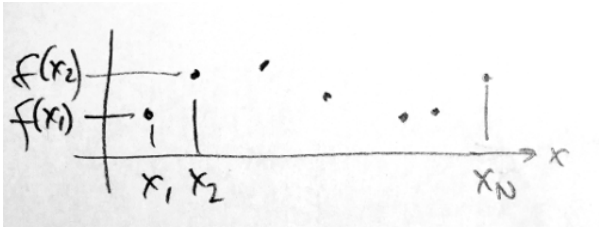
holds if f' bounded; can be nonsmooth but not crazy



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

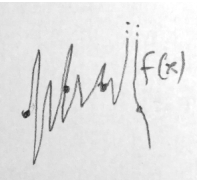
Interpolation in 1D ($d = 1$)

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



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

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

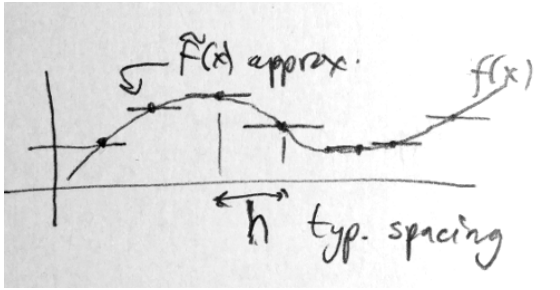


Simplest: use value at x_j nearest to x

"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



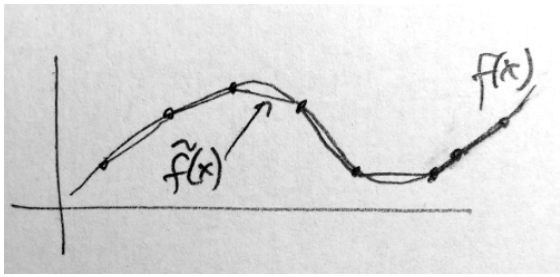
Recall notation "$\mathcal{O}(h)$": exists $C, h_0 > 0$ s.t. error $\leq Ch$ for all $h < h_0$

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 $p = 1, 2$

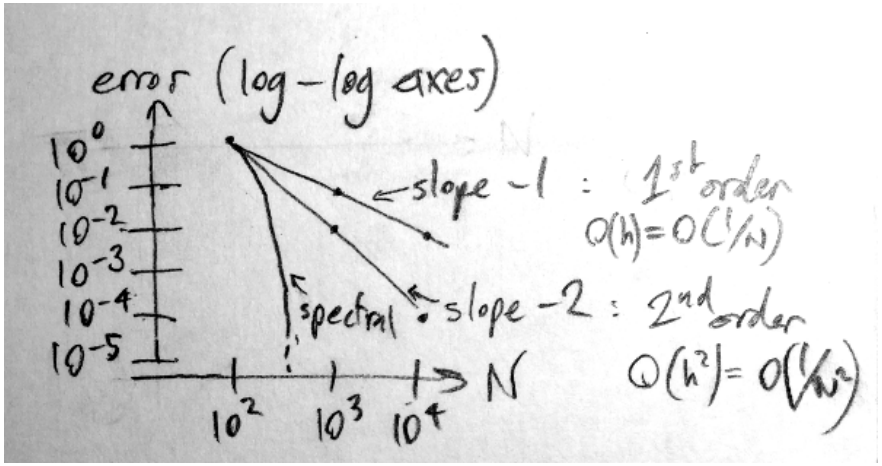
Interlude: convergence rates

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: "relative error vs effort"



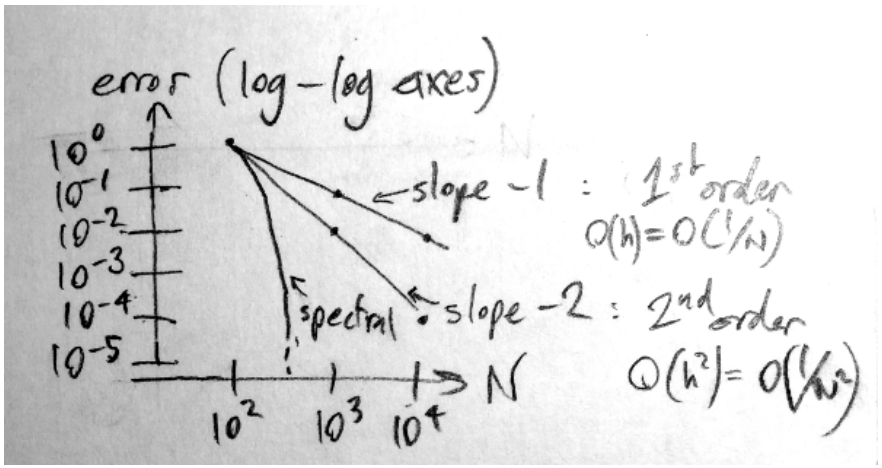
Interlude: convergence rates

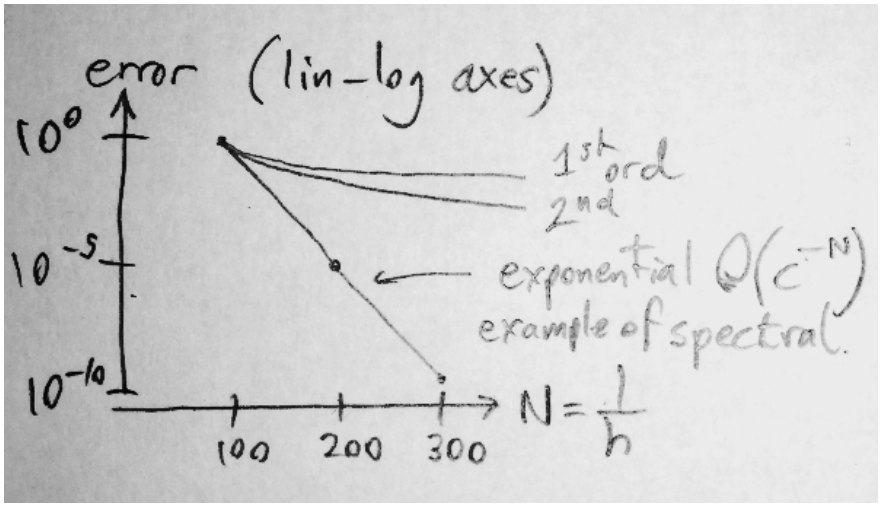
Should know or measure convergence rate of any method you use

- "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: "relative error vs effort"





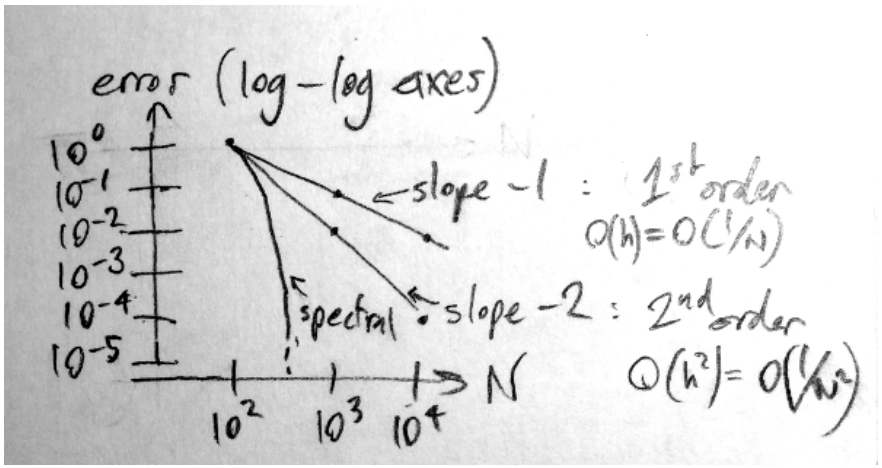
Interlude: convergence rates

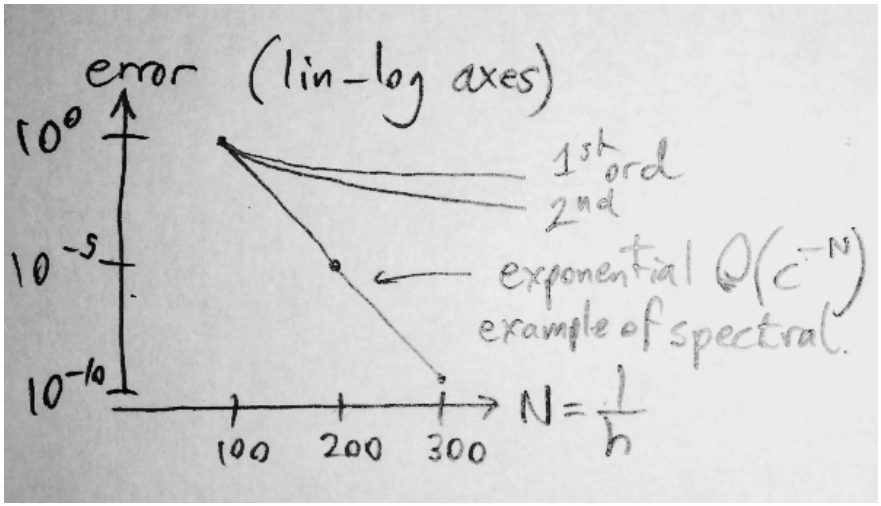
Should know or measure convergence rate of any method you use

- "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: "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? :)

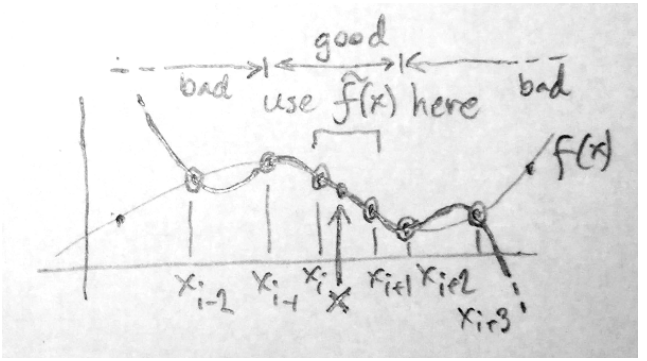
Higher-order interpolation for smooth f: the local idea

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.

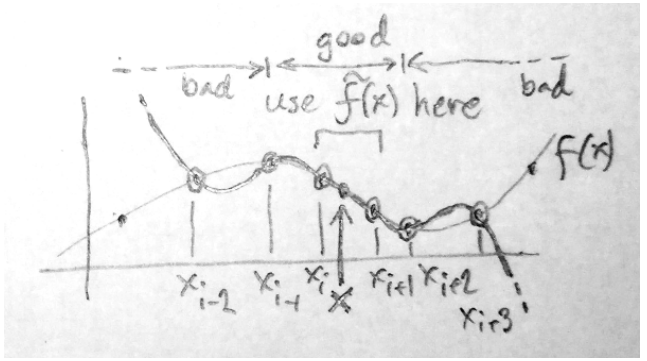
Higher-order interpolation for smooth f: the local idea

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?

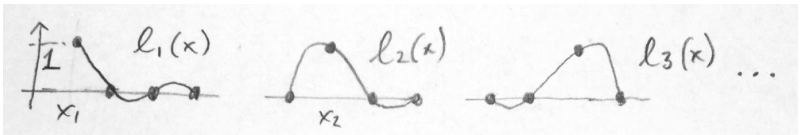
1) Crafty: solve square lin sys for coeffs $\sum_{k<p} x_j^k c_k = y_j \qquad j = 1, \ldots, p$

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

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

   [Tre13, Ch. 5]

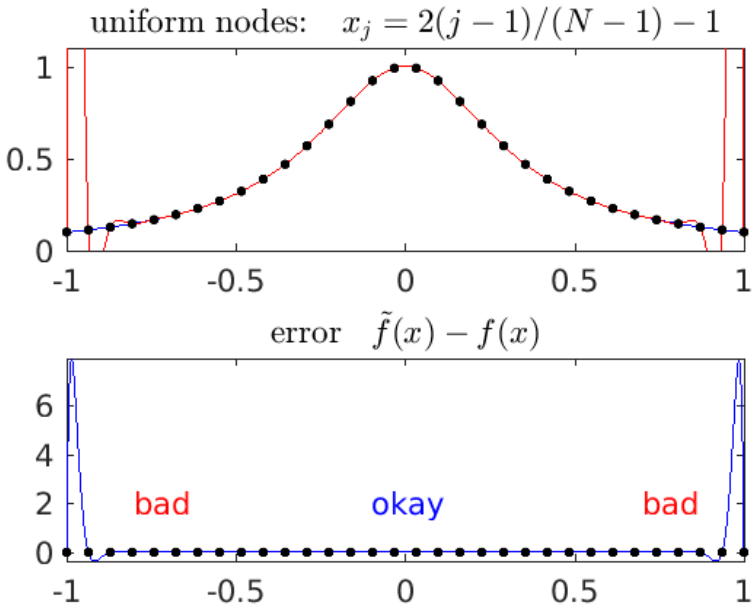
Either way, $\tilde{f}(x) = \sum_{j=1}^{p} y_j \ell_j(x)$ where $\ell_j(x)$ is ℓth Lagrange basis func:



Global polynomial (Lagrange) interpolation?

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

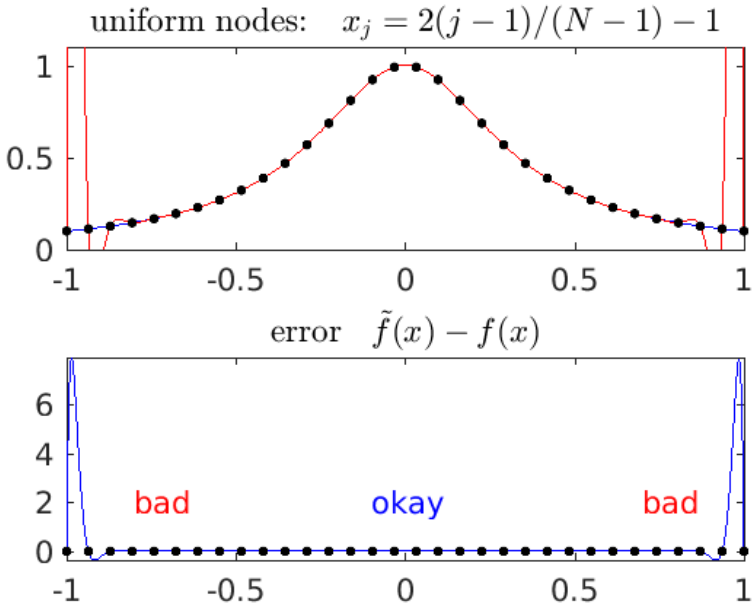
$p = N = 32$, smooth (analytic) $f(x) = \frac{1}{1+9x^2}$ on $[-1, 1]$: (Runge 1901)



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)

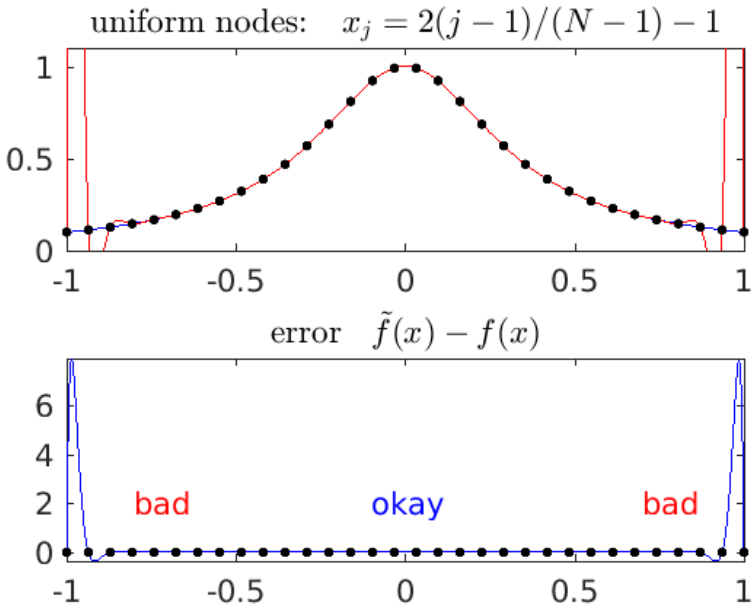


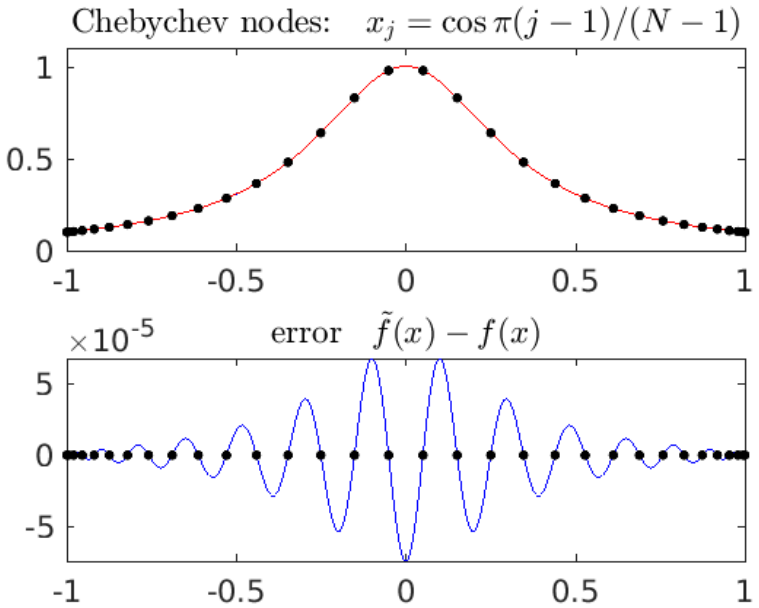
- warning: unif. grid, global interp. fails → only use locally in central region

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)





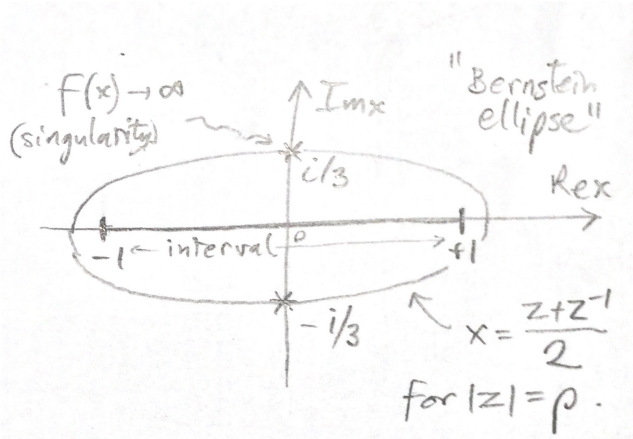
- warning: unif. grid, global interp. fails → only use locally in central region

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 (eg 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 (eg 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

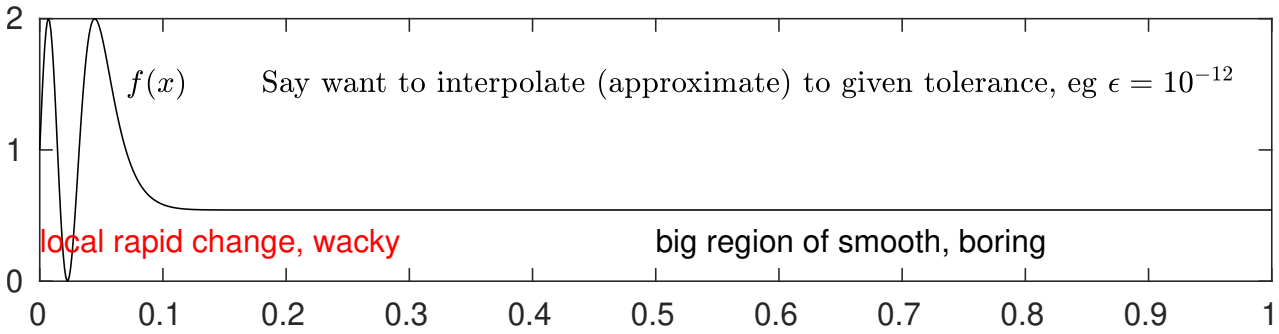
Recap poly approx. $f(x)$ on $[a,b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

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 (eg 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

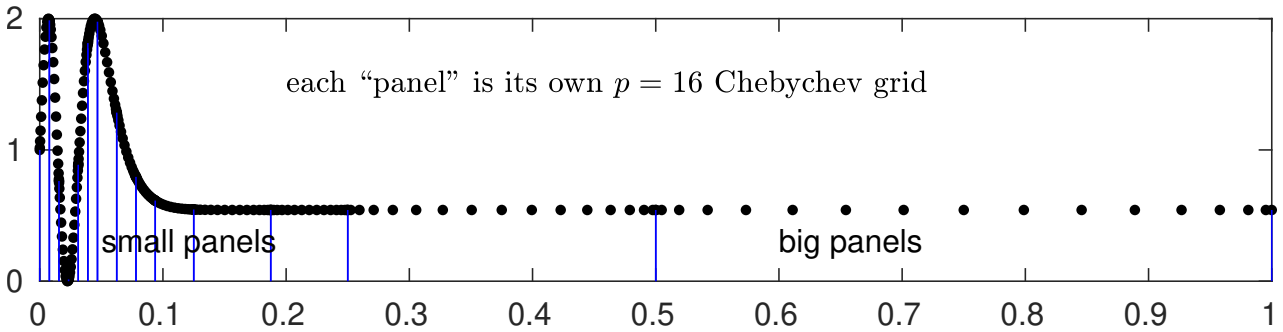
Recap poly approx. $f(x)$ on $[a,b]$: are good & bad node sets $\{x_j\}_{j=1}^N$

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 (eg 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"

Global interpolation of periodic functions I

Just did f on intervals $[a, b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

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

Global interpolation of periodic functions I

Just did f on intervals $[a, b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

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

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

Global interpolation of periodic functions I

Just did f on intervals $[a, b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

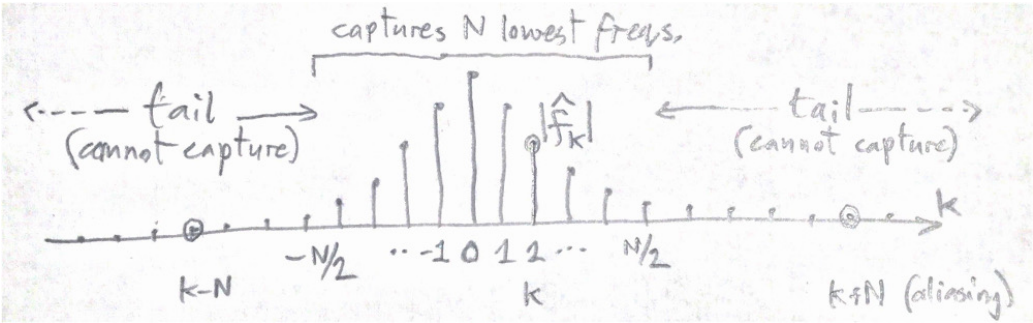
Periodic: $f(x + 2\pi) = f(x)$ for all x, $\quad f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

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

What's best you can do?

get N coeffs right $c_k = \hat{f}_k$

error $\sim$ size of tail $\{\hat{f}_k\}_{|k|\geq N/2}$



Global interpolation of periodic functions I

Just did f on intervals $[a,b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

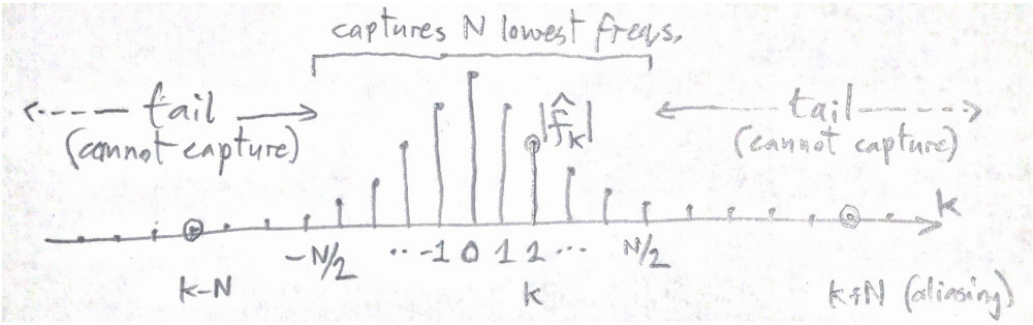
Periodic: $f(x+2\pi) = f(x)$ for all x, $\quad f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

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

What's best you can do?

get N coeffs right $c_k = \hat{f}_k$

error $\sim$ size of tail $\{\hat{f}_k\}_{|k|\geq N/2}$



How read off c_k from *samples* of f on a grid?

uniform grid best (unlike for poly's!); non-uniform needs linear solve, slow $\mathcal{O}(N^3)$ effort

Uniform grid $x_j = \frac{2\pi j}{N}$, $\quad$ set $c_k = \frac{1}{N}\sum_{j=1}^{N} e^{ikx_j} f(x_j)$ simply $\boldsymbol{c} = FFT[\boldsymbol{f}]$

Global interpolation of periodic functions I

Just did f on intervals $[a,b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

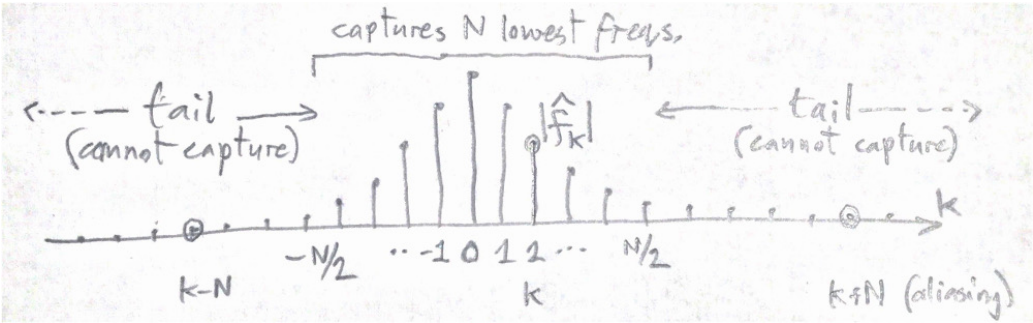
Periodic: $f(x+2\pi) = f(x)$ for all x, $\quad f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

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

What's best you can do?

get N coeffs right $c_k = \hat{f}_k$

error $\sim$ size of tail $\{\hat{f}_k\}_{|k|\geq N/2}$



How read off c_k from *samples* of f on a grid?

uniform grid best (unlike for poly's!); non-uniform needs linear solve, slow $\mathcal{O}(N^3)$ effort

Uniform grid $x_j = \frac{2\pi j}{N}$, $\quad$ set $c_k = \frac{1}{N}\sum_{j=1}^{N} e^{ikx_j} f(x_j)$ simply $\boldsymbol{c} = FFT[\boldsymbol{f}]$

easy to show $c_k = \cdots + \hat{f}_{k-N} + \hat{f}_k + \hat{f}_{k+N} + \hat{f}_{k+2N} + \ldots$

$= \hat{f}_k$ desired $+ \sum_{m\neq 0} \hat{f}_{k+mN}$ aliasing error, small if tail small

Global interpolation of periodic functions I

Just did f on intervals $[a,b]$. global interp. (& integr., etc.) of smooth *periodic* f differs!

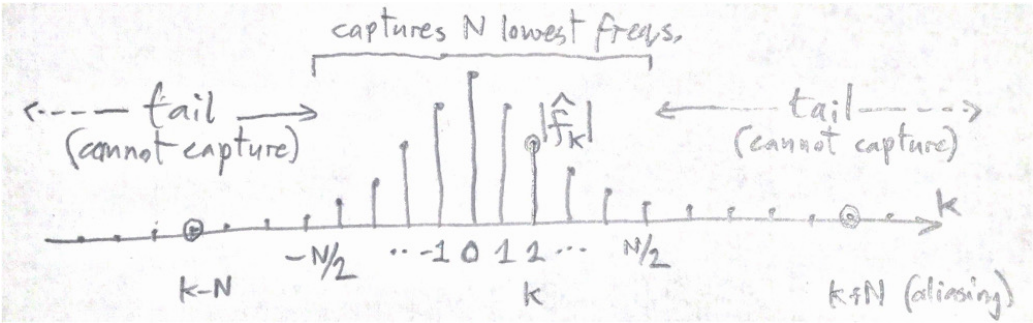
Periodic: $f(x+2\pi) = f(x)$ for all x, $\quad f(x) = \sum_{n\in\mathbb{Z}} \hat{f}_k e^{ikx}$ Fourier series

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

What's best you can do?

get N coeffs right $c_k = \hat{f}_k$

error $\sim$ size of tail $\{\hat{f}_k\}_{|k|\geq N/2}$



How read off c_k from *samples* of f on a grid?

uniform grid best (unlike for poly's!); non-uniform needs linear solve, slow $\mathcal{O}(N^3)$ effort

Uniform grid $x_j = \frac{2\pi j}{N}$, $\quad$ set $c_k = \frac{1}{N}\sum_{j=1}^{N} e^{ikx_j} f(x_j)$ simply $\boldsymbol{c} = FFT[\boldsymbol{f}]$

easy to show $c_k = \cdots + \hat{f}_{k-N} + \hat{f}_k + \hat{f}_{k+N} + \hat{f}_{k+2N} + \ldots$

$= \hat{f}_k$ desired $+ \sum_{m\neq 0} \hat{f}_{k+mN}$ aliasing error, small if tail small

Summary: given N samples $f(x_j)$, interp. error = truncation + aliasing

a crude bound is $\max_{x\in[0,2\pi)} |\tilde{f}(x) - f(x)| \leq 2 \sum_{|k|\geq N/2} |\hat{f}_k|$

ie error controlled by size of tail

Global interpolation of periodic functions II

As grow grid N, how accurate is it? just derived err $\sim$ sum of $|\hat{f}_k|$ in tail $|k| \geq N/2$

Now $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}dx = \frac{1}{2\pi}\int_0^{2\pi} f^{(p)}(x)\frac{e^{-ikx}}{(-ik)^p}dx$ integr. by parts p times

So for a periodic $f \in C^p$, recall means first p derivs of f bounded

$\hat{f}_k = \mathcal{O}(k^{-p})$, tail sum $\mathcal{O}(N^{1-p}) = \mathcal{O}(h^{p-1})$ at least $(p-1)$th order acc.

Global interpolation of periodic functions II

As grow grid N, how accurate is it? just derived err $\sim$ sum of $|\hat{f}_k|$ in tail $|k| \geq N/2$

Now $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}dx = \frac{1}{2\pi}\int_0^{2\pi} f^{(p)}(x)\frac{e^{-ikx}}{(-ik)^p}dx$ integr. by parts p times

So for a periodic $f \in C^p$, recall means first p derivs of f bounded

$\hat{f}_k = \mathcal{O}(k^{-p})$, tail sum $\mathcal{O}(N^{1-p}) = \mathcal{O}(h^{p-1})$ at least $(p-1)$th order acc.

Example of: f smoother $\leftrightarrow$ faster $\hat{f}_k$ tail decay $\leftrightarrow$ faster convergence

Global interpolation of periodic functions II

As grow grid N, how accurate is it? just derived err $\sim$ sum of $|\hat{f}_k|$ in tail $|k| \geq N/2$

Now $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}dx = \frac{1}{2\pi}\int_0^{2\pi} f^{(p)}(x)\frac{e^{-ikx}}{(-ik)^p}dx$ integr. by parts p times

So for a periodic $f \in C^p$, recall means first p derivs of f bounded

$\hat{f}_k = \mathcal{O}(k^{-p})$, tail sum $\mathcal{O}(N^{1-p}) = \mathcal{O}(h^{p-1})$ at least $(p-1)$th order acc.

Example of: f smoother $\leftrightarrow$ faster $\hat{f}_k$ tail decay $\leftrightarrow$ faster convergence

Even smoother case: f analytic, so $f(x)$ analytic in some complex strip $|\operatorname{Im} x| \leq \alpha$

then $\hat{f}_k = \mathcal{O}(e^{-\alpha|k|})$, exponential conv in N (fun proof: shift the contour)

as with Bernstein ellipse, to get exp. conv. *rate* need understand f *off its real axis* (wild!)

Global interpolation of periodic functions II

As grow grid N, how accurate is it? just derived err $\sim$ sum of $|\hat{f}_k|$ in tail $|k| \geq N/2$

Now $\hat{f}_k = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-ikx}dx = \frac{1}{2\pi}\int_0^{2\pi} f^{(p)}(x)\frac{e^{-ikx}}{(-ik)^p}dx$ integr. by parts p times

So for a periodic $f \in C^p$, recall means first p derivs of f bounded

$\hat{f}_k = \mathcal{O}(k^{-p})$, tail sum $\mathcal{O}(N^{1-p}) = \mathcal{O}(h^{p-1})$ at least $(p-1)$th order acc.

Example of: f smoother $\leftrightarrow$ faster $\hat{f}_k$ tail decay $\leftrightarrow$ faster convergence

Even smoother case: f analytic, so $f(x)$ analytic in some complex strip $|\operatorname{Im} x| \leq \alpha$

then $\hat{f}_k = \mathcal{O}(e^{-\alpha|k|})$, exponential conv in N (fun proof: shift the contour)

as with Bernstein ellipse, to get exp. conv. *rate* need understand f *off its real axis* (wild!)

Smoothest case: "band-limited" f with $\hat{f}_k = 0$, $|k| > k_{\max}$,

then interpolant *exact* once $N > 2k_{\max}$

That's theory. In real life you always *measure* your conv. order/rate!

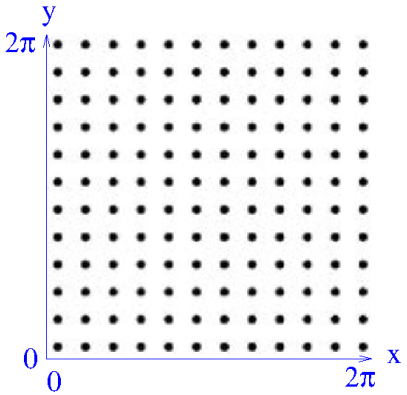
Take-home: for f smooth & periodic, unif. grid global spectral acc.

- use FFTs, cost $\mathcal{O}(N \log N)$, to go between $f(x_j)$ grid & $\hat{f}_k$ Fourier coeffs

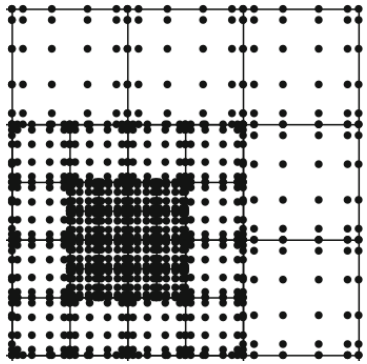
Flavor of interpolation in higher dims $d > 1$

If you can choose the nodes:

- tensor product of 1D interp.
- either global
- or adaptively refined boxes

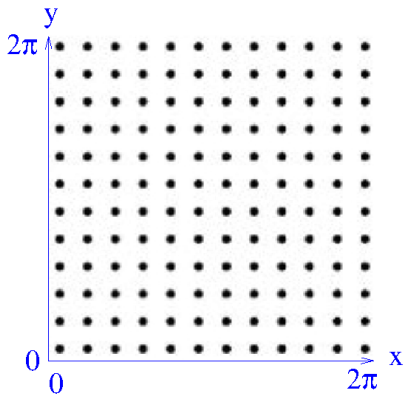


periodic, global

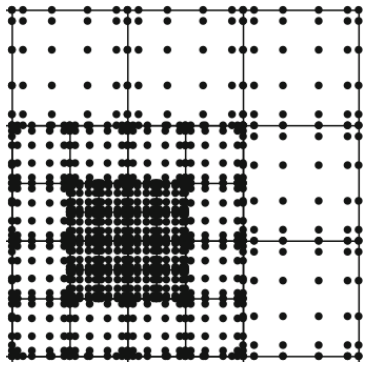

adaptive $p = 6 \times 6$ Cheby

Flavor of interpolation in higher dims $d > 1$

If you can choose the nodes:
- tensor product of 1D interp.
- either global
- or adaptively refined boxes



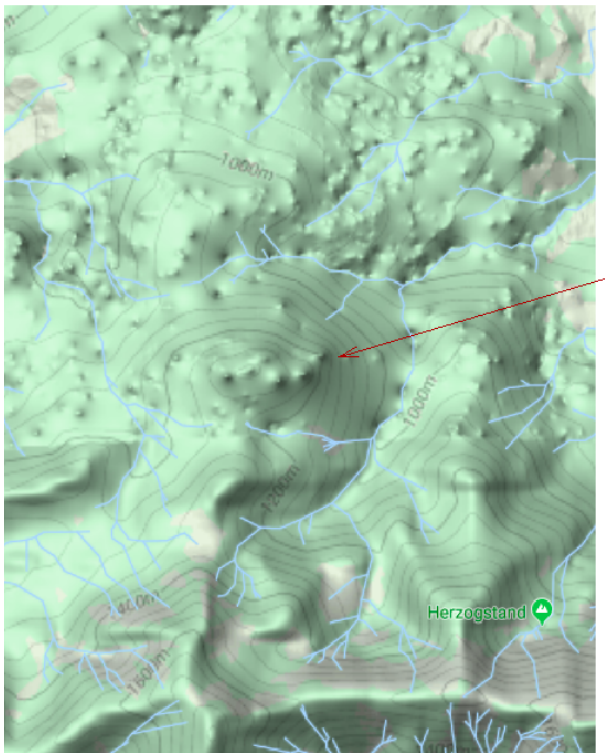
periodic, global


adaptive $p = 6 \times 6$ Cheby

If cannot choose the nodes:
interp. $f(\boldsymbol{x})$ from scattered data $\{\boldsymbol{x}_i\}$ is hard

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

are amusing jumps in node grids:



height f(x) interp from unstructured points in 2D, kernel method

pock–marks!

interp from Cartesian grid, more accurate

Or if know f smooth

fit local multivariate polynomial

If f noisy and smooth, many methods

kriging, kernels, ***

Numerical integration (back to $d = 1$)

Here, usually user gets to choose the nodes x_j so pick best ones!

Idea: get interpolant $\tilde{f}$ from data $f(x_j)$, then *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

Gaussian idea: there is a well-chosen node set giving twice the rate vs interp.

PTR exact out to $N-1$, not merely $N/2$ (detail)

same in interval for poly degree

Integration in dims $d > 1$

For d up to a few, products of 1D nodes exp. cost $N = n^d$

can also do in other coord sys, eg sphere $=$ $z \in [-1, 1]$ tensor prod. with periodic $\phi \in [0, 2\pi)$

Higher d

Monte Carlo methods: sum N values of $f(\boldsymbol{x}_j)$ for $\boldsymbol{x}_j$ random points

error $= \mathcal{O}(N^{-1/2})$ 1/2-order acc, indep of dim d

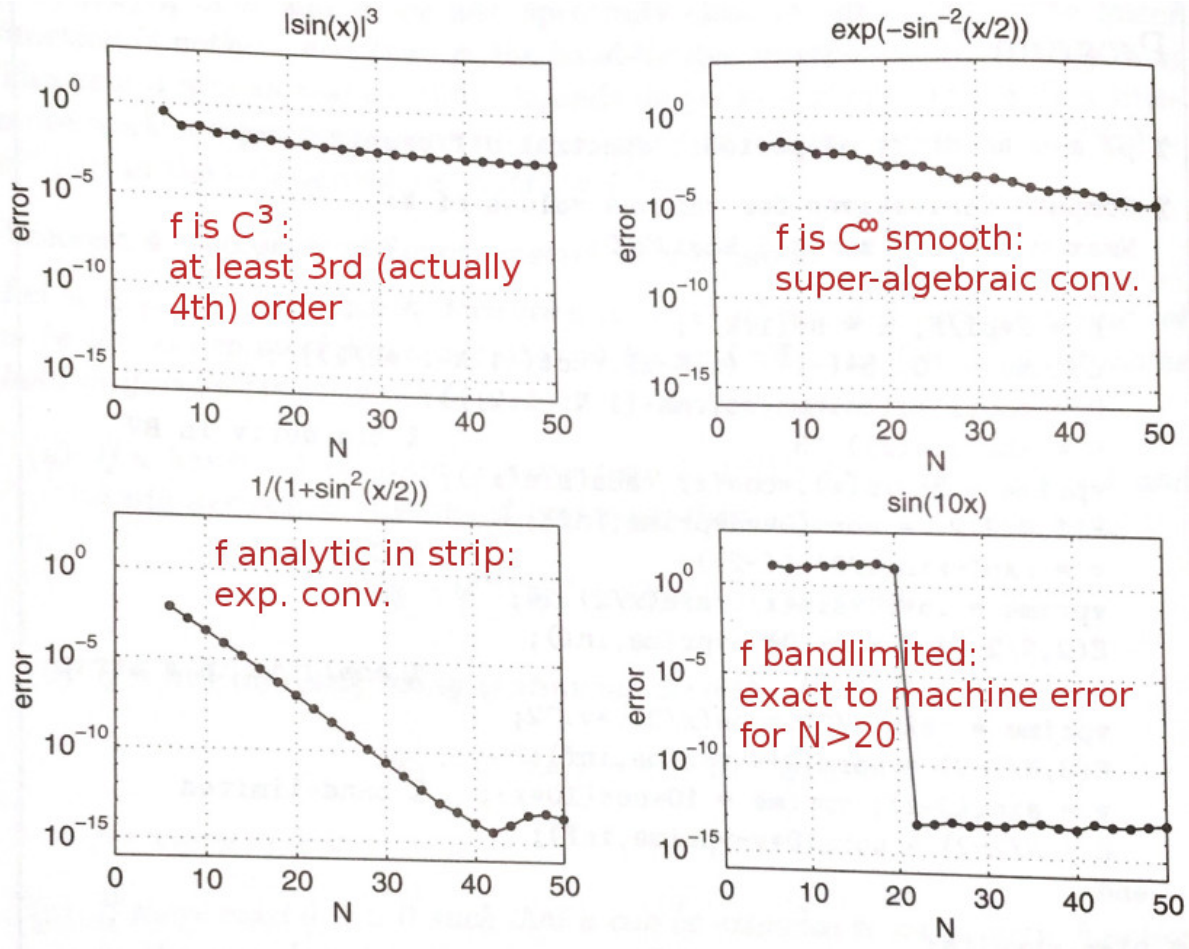
Note MCMC does not easily give $\int f(\boldsymbol{x}) d\boldsymbol{x}$, just ratios of such

Differentiation in $d = 1$

As w/ integration: once have interpolant, differentiate it exactly

D is $p \times p$ matrix acting on func. values $\{f(x_j)\}$ to give $\{f'(x_j)\}$

Examples: periodic grid in $[0, 2\pi)$, max error:



Topics not covered

extrapolation

Rounding error [GC12, Ch. 5–6] coming in Lec II

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: $\boldsymbol{u}$ 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

Finite difference idea

set up linear system for 1D ODE $u'' = g$ w/ some BCs

Uniform 1D grid in $[0, 1]$ $\quad h = 1/N$

$u''(x) = \frac{u(x-h)-2u(x)+u(x+h)}{h^2} + \mathcal{O}(h^4)$ via Taylor thm

show results as $h \to 0$ for known u

trade-off rounding error

Alex can do this, or move to diff lec I and do f' ?

IVPs to BVPs

Motivate focus on BVPs by mentioning:

- Most people familiar with method of lines
- Equivalent "propagator" approach: discretize time, then solve BVP
- Most timesteppers are FD methods (some examples)

Finite difference methods

Basic viewpoint:

- Construct Taylor-series approximation to value at neighboring points
- For N points, expand to (N-1) derivatives (error $\mathcal{O}(h^N)$)
- Eliminate system to get approximation to d-th derivative ($d < N$)

(error $\mathcal{O}(h^{N-d})$)

Alternate viewpoint:

- Differentiate the unique interpolating polynomial of deg N-1

Illustrate with centered forward 1st-order scheme, centered 2nd-order scheme (get extra degree for free).

Timestepping with finite differente methods

Consider temporal ODE $u'(t) = f(u(t))$

Explicit methods: simple, just require evaluating $f(u_n)$.

- E.g. Forward Euler: use 1st-order forward difference rule

Unstable for large timesteps

$$u'(t) = -\lambda u(t) \quad \lambda > 0 \tag{1}$$

$$\frac{u_{n+1} - u_n}{k} = -\lambda u_n \tag{2}$$

$$u_{n+1} = (1 - k\lambda)u_n \tag{3}$$

$$\boxed{k\lambda < 2} \tag{4}$$

Implicit methods: stable, but require solving $f(u^{n+1}) = ...$

- Backward Euler: use 1st-order backward difference rule

Stability, but potentially at high cost

$$u'(t) = -\lambda u(t) \quad \lambda > 0 \tag{5}$$

$$\frac{u_{n+1} - u_n}{} = -\lambda u_{n+1} \tag{6}$$

Advanced finite difference techniques

- Selecting stencils term by term (e.g. upwinding)
- Adaptive stencil selection (e.g. WENO)

Resources: ...
Codes: ...

Finite element methods

Partition domain into elements, represent unknown within each element using basis functions (usually polynomials)

Convert equations to weak form

- Give example
- Weak form advantages: lowers order, works if lower derivs are integrable (good for discontinuities, hyperbolic), can use tent functions

Replace weak form with Galerkin/weighted-residual form to get algebraic system

Traditional FEM methods use tent functions, produce continuous solutions. Fluxes at cell boundaries can be computed from internal representation, but can be discontinuous (non conservative).

Finite volume methods

Case of FEM taking functions to be piecewise constant inside elements
Resulting system is exactly conservative (good for hyperbolic), but requires Riemann solve

- Order increased by using neihgbors to reconstruct higher order internal representations or fluxes (slope/flux reconstruction)
- Reconstruction is nonlinear to control oscillations around discontinuities (TVD, ENO/WENO).

Very common in CFD, CCA

Advanced finite element methods

- Discontinuous Galerkin: allow discontinuities, need Riemann solvers again
- Spectral elements: move towards large p for better internal representations

Resources: ...

Codes: ...

Spectral methods

Limit of very few elements with very large p: exponential accuracy for smooth problems

Traditional techniques: Fourier spectral methods.

- Fast due to FFT: optimal complexity with exponential accuracy
- Limited to simple geometries / equations with symmetries

Polynomial spectral methods

- More flexible in terms of equations
- Still limited to simple geometries: cubes, spheres, cylinders, etc.
- Still a weak method, but don't integrate by parts. Apply Galerkin directly.

Modern research: sparse methods for arbitrary equations.

Resources: ...

Codes: ...

Boundary integral methods

Use knowledge of PDEs in constructing numerical solutions:
For linear PDEs dominated by boundary rather than bulk terms, compute solution by forming integral equation of the fundamental solution/Green's function. Reduced dimensionality. Improved conditioning. Low-rank interactions and fast methods. Reconstruct solution in the bulk.
Examples: Stokes flow, Helmholtz, Maxwell, typically with homogeneous media

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`.