

# FWAM Session B: Function Approximation and Differential Equations

**Alex Barnett<sup>1</sup>** and **Keaton Burns<sup>2</sup>**

Wed, 10/30/19

<sup>1</sup>Center for Computational Mathematics, Flatiron Institute

<sup>2</sup>Center for Computational Astrophysics, Flatiron Institute, and Department of Mathematics, MIT

# LECTURE 1: interpolation, integration, spectral methods

## Motivations

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

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

- Interpolation:

given exact data  $f(x_i)$  at some  $x_i$ , model  $f(x)$  at other points  $x$   
cheap but accurate look-up table for expensive  $f(x)$

Contrast: fit noisy data = learning (pdf for) params in model, via likelihood/prior

- (Numerical) integration:

eg computing expectation values given a pdf, or their quantum version

Contrast: Monte Carlo (random, high-dim.) integration, Thurs am

- Differentiation:

get gradient  $\nabla f$ : 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 11

# Goals for LECTURE I

## TODO

teach range of practical methods focusing on 1D  
pointers to dimensions  $d > 1$

## Concepts:

convergence order      how does accuracy improve w/ 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 very fast

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

rounding error & catastrophic cancellation      how not shoot self in the foot

interpolation = func. representation, is key to all else

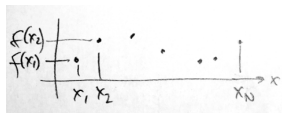
## Interpolation in 1D ( $d = 1$ )

Say  $y_j = f(x_j)$  known at nodes  $\{x_j\}$

exact data, not noisy

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

$N$ -pt "grid"

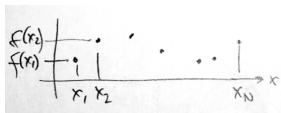


## Interpolation in 1D ( $d = 1$ )

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

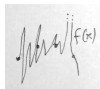
exact data, not noisy

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



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

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

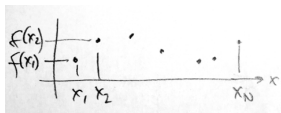


## Interpolation in 1D ( $d = 1$ )

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

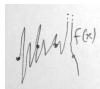
exact data, not noisy

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



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

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

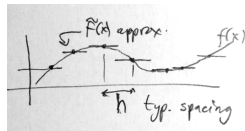


Simplest: use value at  $x_j$  nearest to  $x$

"snap to grid"

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

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



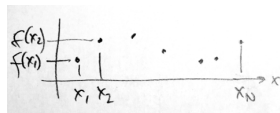
Recall notation " $\mathcal{O}(h)$ ": exists  $C, h_0 > 0$  s.t. error  $\leq Ch$  for all  $h < h_0$

# Interpolation in 1D ( $d = 1$ )

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

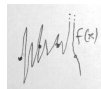
exact data, not noisy

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



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

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

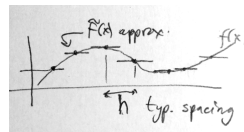


**Simplest:** use value at  $x_j$  nearest to  $x$

"snap to grid"

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

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



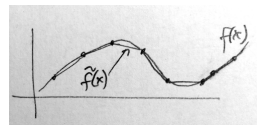
Recall notation " $\mathcal{O}(h)$ ": exists  $C, h_0 > 0$  s.t. error  $\leq Ch$  for all  $h < h_0$

**Piecewise linear:**

"connect the dots"

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

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  $p = 1, 2$



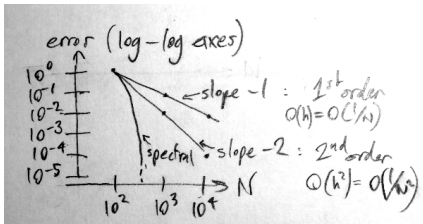
## Interlude: convergence rates

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  $p = 1, 2$

There's only one graph in numerical analysis: “relative error vs effort”



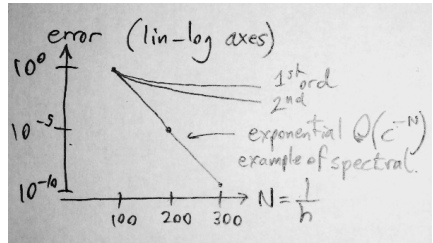
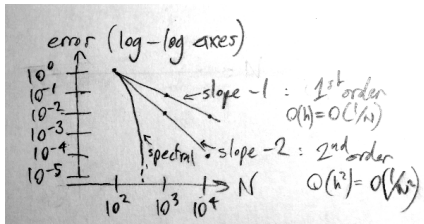
## Interlude: convergence rates

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  $p = 1, 2$

There's only one graph in numerical analysis: “relative error vs effort”



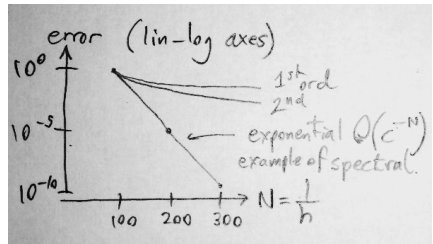
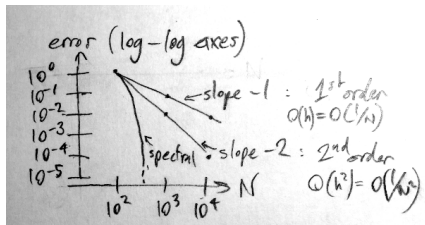
## Interlude: convergence rates

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  $p = 1, 2$

There's 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”, beats  $\mathcal{O}(N^{-p})$  for any  $p$

- 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>

How big is prefactor  $C$  in error  $\leq Ch^p$ ? Has asymp. rate even kicked in yet? :)

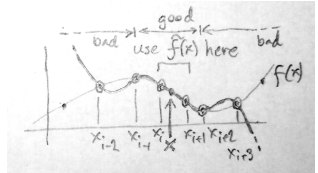
# Higher-order interpolation for smooth $f$ : the local idea

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!



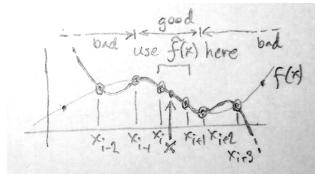
# Higher-order interpolation for smooth $f$ : the local idea

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!

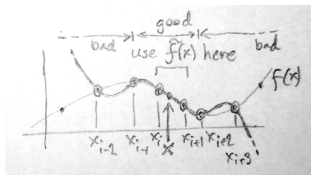


- error  $\mathcal{O}(h^p)$ , ie high order, but  $\tilde{f}$  not continuous ( $\tilde{f} \notin C$ ) has small jumps  
if must have cont, recommend splines, eg cubic  $p = 3$ :  $\tilde{f} \in C^2$ , meaning  $\tilde{f}''$  is cont.

# 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  $\mathcal{O}(h^p)$ , ie high order, but  $\tilde{f}$  not continuous ( $\tilde{f} \notin C$ ) has small jumps  
if must have cont, recommend splines, eg cubic  $p = 3$ :  $\tilde{f} \in C^2$ , meaning  $\tilde{f}''$  is cont.

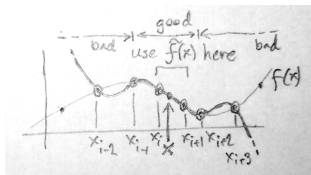
How to find this degree- $(p-1)$  poly?

- 1) crafty: solve square lin sys for coeffs  $\sum_{k < p} x_j^k c_k = y_j \quad j = 1, \dots, p$   
ie,  $V\mathbf{c} = \mathbf{y}$   $V$ ="Vandermonde" matrix, is ill-cond. but works

# 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!

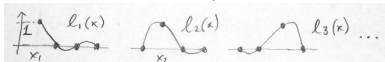
- error  $\mathcal{O}(h^p)$ , ie high order, but  $\tilde{f}$  not continuous ( $\tilde{f} \notin C$ ) has small jumps  
if must have cont, recommend splines, eg cubic  $p = 3$ :  $\tilde{f} \in C^2$ , meaning  $\tilde{f}''$  is cont.

How to find this degree- $(p-1)$  poly?

- crafty: solve square lin sys for coeffs  $\sum_{k < p} x_j^k c_k = y_j$   $j = 1, \dots, p$   
ie,  $V\mathbf{c} = \mathbf{y}$   $V = \text{"Vandermonde" matrix, is ill-cond. but works}$

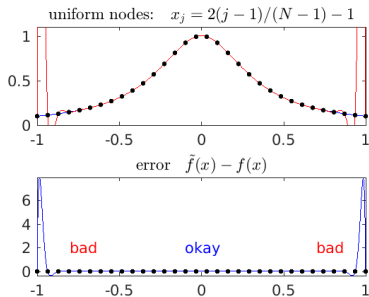
- traditional: barycentric formula  $\tilde{f}(x) = \frac{\sum_{j=1}^p \frac{y_j}{x-x_j} w_j}{\sum_{j=1}^p \frac{1}{x-x_j} w_j}$   $w_j = \frac{1}{\prod_{i \neq j} (x_j - x_i)}$   
[Tre13, Ch. 5]

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



## Global polynomial (Lagrange) interpolation?

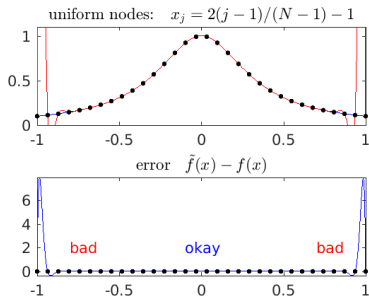
Want increase order  $p$ . Use *all* data, get single  $\tilde{f}(x)$ , so  $p = N$ ? “global”  
 $p = N = 32$ , smooth (analytic)  $f(x) = \frac{1}{1+9x^2}$  on  $[-1, 1]$ : (Runge 1901)





## Global polynomial (Lagrange) interpolation?

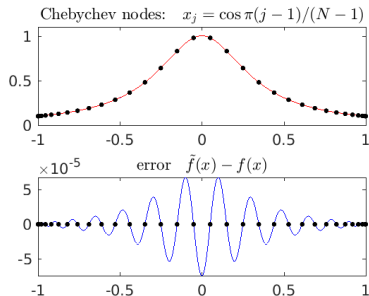
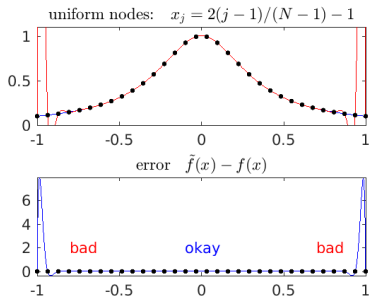
Want increase order  $p$ . Use *all* data, get single  $\tilde{f}(x)$ , so  $p = N$ ? “global”  
 $p = N = 32$ , smooth (analytic)  $f(x) = \frac{1}{1+9x^2}$  on  $[-1, 1]$ : (Runge 1901)



- 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...

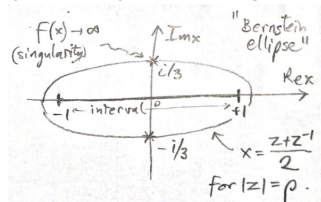
“Chebyshev”: means non-unif. grid density  $\sim \frac{1}{\sqrt{1-x^2}}$

- our first spectral method

$$\max \text{err} = \mathcal{O}(\rho^{-N})$$

exponential conv!

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



## Node choice and adaptivity

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

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

- data fitting applications: No (or noisy variants: kriging, Gaussian processes, etc)  
use local poly (central region only!), or something stable (eg splines) [GC12]
- almost all else, interp., quadrature, PDE solvers: Yes so pick good nodes!

## Node choice and adaptivity

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

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

- data fitting applications: No (or noisy variants: kriging, Gaussian processes, etc)  
use local poly (central region only!), or something stable (eg splines) [GC12]
- almost all else, interp., quadrature, PDE solvers: Yes so pick good nodes!

**Adaptivity idea**      global is inefficient if  $f$  smooth in most places, but not everywhere

## Node choice and adaptivity

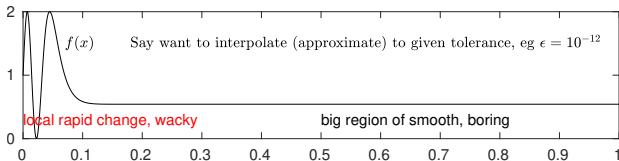
Recap: poly approx.  $f(x)$  on  $[a, b]$  has good & bad node sets  $\{x_j\}_{j=1}^N$

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

- data fitting applications: No (or noisy variants: kriging, Gaussian processes, etc)  
use local poly (central region only!), or something stable (eg splines) [GC12]
- almost all else, interp., quadrature, PDE solvers: Yes so pick good nodes!

### Adaptivity idea

global is inefficient if  $f$  smooth in most places, but not everywhere



## Node choice and adaptivity

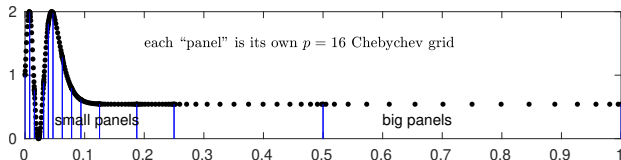
Recap: poly approx.  $f(x)$  on  $[a, b]$  has good & bad node sets  $\{x_j\}_{j=1}^N$

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

- data fitting applications: No (or noisy variants: kriging, Gaussian processes, etc)  
use local poly (central region only!), or something stable (eg splines) [GC12]
- almost all else, interp., quadrature, PDE solvers: Yes so pick good nodes!

### Adaptivity idea

global is inefficient if  $f$  smooth in most places, but not everywhere



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

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  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  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  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  Fourier series

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



# Global interpolation of periodic functions I

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

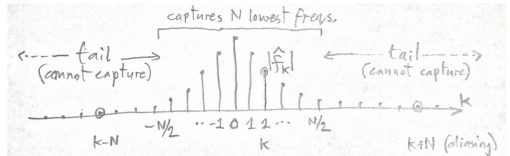
Periodic:  $f(x + 2\pi) = f(x)$  for all  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  Fourier series

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?

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!

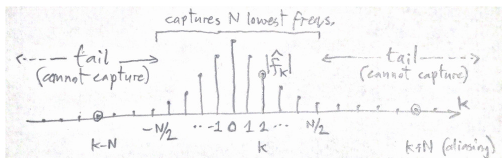
Periodic:  $f(x + 2\pi) = f(x)$  for all  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  Fourier series

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?

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

# Global interpolation of periodic functions I

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

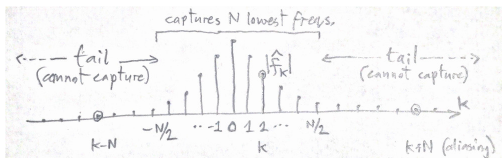
Periodic:  $f(x + 2\pi) = f(x)$  for all  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  Fourier series

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?

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

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

$= \hat{f}_k$  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!

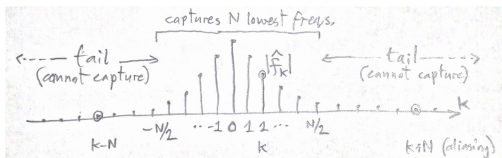
Periodic:  $f(x + 2\pi) = f(x)$  for all  $x$ ,  $f(x) = \sum_{n \in \mathbb{Z}} \hat{f}_k e^{ikx}$  Fourier series

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?

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

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

$= \hat{f}_k$  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 sum 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 first  $p$  derivs of  $f$  bounded

$$\hat{f}_k = \mathcal{O}(k^{-p}), \text{ tail sum } \mathcal{O}(N^{1-p}) \quad (p-1)\text{th order acc.} \quad (\text{better: [Tre00]})$$

## Global interpolation of periodic functions II

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

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 first  $p$  derivs of  $f$  bounded

$$\hat{f}_k = \mathcal{O}(k^{-p}), \text{ tail sum } \mathcal{O}(N^{1-p}) \quad (p-1)\text{th order acc.} \quad (\text{better: [Tre00]})$$

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  $\text{err} \sim \text{sum of } |\hat{f}_k| \text{ in tail } |k| \geq N/2$

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 first  $p$  derivs of  $f$  bounded

$$\hat{f}_k = \mathcal{O}(k^{-p}), \text{ tail sum } \mathcal{O}(N^{1-p}) \quad (p-1)\text{th order acc.} \quad (\text{better: [Tre00]})$$

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  $|\text{Im } x| \leq \alpha$   
then  $\hat{f}_k = \mathcal{O}(e^{-\alpha|k|})$ , exp. conv.  $\mathcal{O}(e^{-\alpha N/2})$  (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  $\text{err} \sim \text{sum of } |\hat{f}_k| \text{ in tail } |k| \geq N/2$

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 first  $p$  derivs of  $f$  bounded

$$\hat{f}_k = \mathcal{O}(k^{-p}), \text{ tail sum } \mathcal{O}(N^{1-p}) \quad (p-1)\text{th order acc.} \quad (\text{better: [Tre00]})$$

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  $|\text{Im } x| \leq \alpha$   
then  $\hat{f}_k = \mathcal{O}(e^{-\alpha|k|})$ , exp. conv.  $\mathcal{O}(e^{-\alpha N/2})$  (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  $N > 2k_{\max}$



## Global interpolation of periodic functions II

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

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 first  $p$  derivs of  $f$  bounded

$$\hat{f}_k = \mathcal{O}(k^{-p}), \text{ tail sum } \mathcal{O}(N^{1-p}) \quad (p-1)\text{th order acc.} \quad (\text{better: [Tre00]})$$

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  $|\text{Im } x| \leq \alpha$   
then  $\hat{f}_k = \mathcal{O}(e^{-\alpha|k|})$ , exp. conv.  $\mathcal{O}(e^{-\alpha N/2})$  (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  $N > 2k_{\max}$

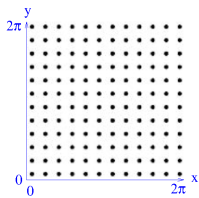
That’s theory. In real life you always **measure** your conv. order/rate!

Take-home:

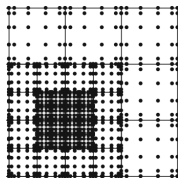
- $f$  smooth, periodic, global interpolation w/ uniform grid: spectral acc.
- key to spectral methods. FFT cost  $\mathcal{O}(N \log N)$  swaps from  $f(x_j)$  grid to  $\hat{f}_k$

## 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



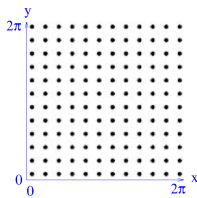
periodic, global



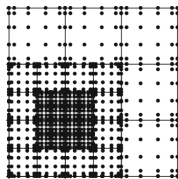
adaptive  $p = 6 \times 6$  Cheby

# 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

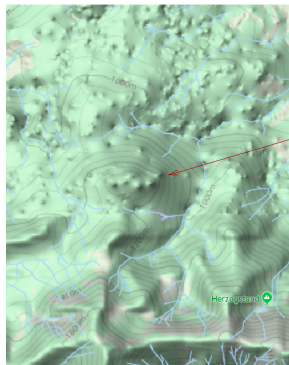
If *cannot* choose the nodes:  
interp.  $f(\mathbf{x})$  from scattered data  
 $\{\mathbf{x}_i\}$  is hard

Eg google terrain:  $f(\mathbf{x})$  rough  $\rightarrow$  v. low ord,  
see amusing transition in interp. type:

Or if know  $f$  smooth

fit local multivariate polynomial

If  $f$  noisy and smooth, many  
methods  
kriging, kernels, \*\*\*



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

pock-marks!

interp from  
Cartesian grid,  
more accurate

## Numerical integration (back to $d = 1$ )

Task: eval.  $\int_a^b f(x)dx$  accurately w/ least number of func. evals,  $N$

“quadrature”: nodes  $\{x_j\}$ , weights  $\{w_j\}$ , s.t.  $\int_a^b f(x)dx \approx \sum_{j=1}^N w_j f(x_j)$

Idea: get interpolant  $\tilde{f}$  thru data  $f(x_j) \rightarrow$  *integrate that exactly*

“intepolatory quadrature”

# Numerical integration (back to $d = 1$ )

Task: eval.  $\int_a^b f(x)dx$  accurately w/ least number of func. evals,  $N$

“quadrature”: nodes  $\{x_j\}$ , weights  $\{w_j\}$ , s.t.  $\int_a^b f(x)dx \approx \sum_{j=1}^N w_j f(x_j)$

Idea: get interpolant  $\tilde{f}$  thru data  $f(x_j) \rightarrow$  *integrate that exactly*

Examples:

- local piecewise linear  $\rightarrow$  composite trapezoid rule

$w_j = h$  except  $h/2$  at ends. low-order, err  $\mathcal{O}(N^{-2})$ , avoid!



- $N$ -node global poly  $\rightarrow$  gives  $\{w_j\}$  integrating degree  $N-1$  exactly

err  $\mathcal{O}(\rho^{-N})$

solve lin sys  $V^T \mathbf{w} = \{\int_a^b x^k dx\}_{k=0}^{N-1}$  (Newton–Coates)

- better: “Gaussian”  $\{x_j, w_j\}$  integrates deg.  $2N-1$  exactly! err  $\mathcal{O}(\rho^{-2N})$

Adaptive quadrature (Gauss in each panel) excellent: codes `quadgk`, `scipy`, etc

# Numerical integration (back to $d = 1$ )

Task: eval.  $\int_a^b f(x)dx$  accurately w/ least number of func. evals,  $N$

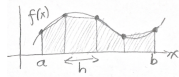
“quadrature”: nodes  $\{x_j\}$ , weights  $\{w_j\}$ , s.t.  $\int_a^b f(x)dx \approx \sum_{j=1}^N w_j f(x_j)$

Idea: get interpolant  $\tilde{f}$  thru data  $f(x_j) \rightarrow$  *integrate that exactly*

Examples:

- local piecewise linear  $\rightarrow$  composite trapezoid rule

$w_j = h$  except  $h/2$  at ends. low-order, err  $\mathcal{O}(N^{-2})$ , avoid!



- $N$ -node global poly  $\rightarrow$  gives  $\{w_j\}$  integrating degree  $N-1$  exactly

err  $\mathcal{O}(\rho^{-N})$  solve lin sys  $V^T \mathbf{w} = \{\int_a^b x^k dx\}_{k=0}^{N-1}$  (Newton–Coates)

- better: “Gaussian”  $\{x_j, w_j\}$  integrates deg.  $2N-1$  exactly! err  $\mathcal{O}(\rho^{-2N})$

Adaptive quadrature (Gauss in each panel) excellent: codes `quadgk`, `scipy`, etc

- periodic case:  $x_j = \frac{2\pi j}{N}$ ,  $w_j = \frac{2\pi}{N}$  excellent “periodic trap. rule”

easy to check integrates  $e^{ikx}$  exactly for  $|k| < N$ , “Gaussian”

$f$  analytic in  $|\operatorname{Im} x| < \alpha$  gives exp. conv.  $\mathcal{O}(e^{-\alpha N})$ , twice as good as interp!

# Numerical integration (back to $d = 1$ )

Task: eval.  $\int_a^b f(x)dx$  accurately w/ least number of func. evals,  $N$

“quadrature”: nodes  $\{x_j\}$ , weights  $\{w_j\}$ , s.t.  $\int_a^b f(x)dx \approx \sum_{j=1}^N w_j f(x_j)$

Idea: get interpolant  $\tilde{f}$  thru data  $f(x_j) \rightarrow$  *integrate that exactly*

Examples:

“interpolatory quadrature”

- local piecewise linear  $\rightarrow$  composite trapezoid rule

$w_j = h$  except  $h/2$  at ends. low-order, err  $\mathcal{O}(N^{-2})$ , avoid!



- $N$ -node global poly  $\rightarrow$  gives  $\{w_j\}$  integrating degree  $N-1$  exactly

err  $\mathcal{O}(\rho^{-N})$  solve lin sys  $V^T \mathbf{w} = \{\int_a^b x^k dx\}_{k=0}^{N-1}$  (Newton–Coates)

- better: “Gaussian”  $\{x_j, w_j\}$  integrates deg.  $2N-1$  exactly! err  $\mathcal{O}(\rho^{-2N})$

Adaptive quadrature (Gauss in each panel) excellent: codes `quadgk`, `scipy`, etc

- periodic case:  $x_j = \frac{2\pi j}{N}$ ,  $w_j = \frac{2\pi}{N}$  excellent “periodic trap. rule”

easy to check integrates  $e^{ikx}$  exactly for  $|k| < N$ , “Gaussian”

$f$  analytic in  $|\operatorname{Im} x| < \alpha$  gives exp. conv.  $\mathcal{O}(e^{-\alpha N})$ , twice as good as interp!

demo: `N=14; sum(exp(cos(2*pi*(1:N)/N)))/N - besseli(0,1)`

`ans = 1.3e-15`

# Advanced integration

- custom quadr. for singularity eg  $f(x) = \text{smooth} \cdot |x|^{-1/2}$   
or for a new set of funcs. "gen. Gaussian quad."
- high-order end-corrections to trap. rule
- 

(CCM: Manas Rachh)

(Alpert '99)



# Advanced integration

- custom quadr. for singularity eg  $f(x) = \text{smooth} \cdot |x|^{-1/2}$   
or for a new set of funcs. “gen. Gaussian quad.” (CCM: Manas Rachh)
- high-order end-corrections to trap. rule (Alpert '99)
- ⋮

## Higher dimensions $d > 1$

For  $d$  a few, tensor products of 1D  $n$ -node grids in each dim

other coord systems, eg sphere, can use tensor product in  $(\theta, \phi)$

adaptivity works: subdivide into boxes same pic as for interp.

## Much higher $d \gg 1$

Now exponential cost  $N = n^d$  kills you :(

Are “sparse grids” scaling better rely on funcs aligning w/ axes

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

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

\*\*\* check

## Numerical differentiation

Task: given ability to eval.  $f(\mathbf{x})$  anywhere, how get  $\nabla f(\mathbf{x})$  ? assume smooth

## Numerical differentiation

Task: given ability to eval.  $f(\mathbf{x})$  anywhere, how get  $\nabla f(\mathbf{x})$  ? assume smooth

Finite differencing idea, 1D:  $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \mathcal{O}(h^2)$  Taylor's thm

Want smallest error:

suggests taking  $h \rightarrow 0$  ?

Let's see how that goes...

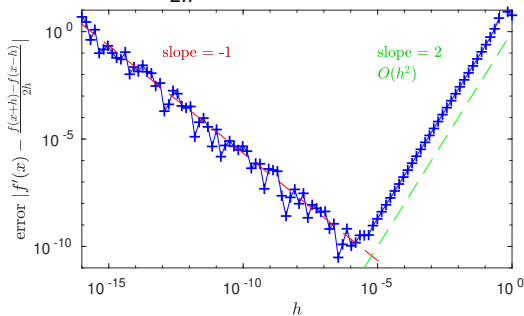
# Numerical differentiation

Task: given ability to eval.  $f(\mathbf{x})$  anywhere, how get  $\nabla f(\mathbf{x})$  ? assume smooth

Finite differencing idea, 1D:  $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \mathcal{O}(h^2)$  Taylor's thm

Want smallest error:  
suggests taking  $h \rightarrow 0$  ?

Let's see how that goes. . .



- shrinking  $\mathcal{O}(h^2)$  error gets swamped by a new growing error. . . what?

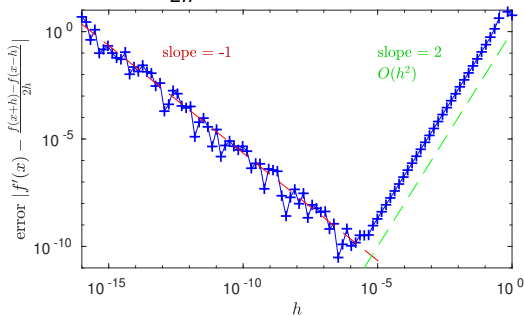
# Numerical differentiation

Task: given ability to eval.  $f(\mathbf{x})$  anywhere, how get  $\nabla f(\mathbf{x})$  ? assume smooth

Finite differencing idea, 1D:  $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \mathcal{O}(h^2)$  Taylor's thm

Want smallest error:  
suggests taking  $h \rightarrow 0$  ?

Let's see how that goes...



- shrinking  $\mathcal{O}(h^2)$  error gets swamped by a new growing error... what?
- CPU arithmetic done only to relative "rounding error"  $\epsilon_{\text{mach}} \sim 10^{-16}$
- subtracting v. close  $f(x+h)$  and  $f(x-h)$ : "catastrophic cancellation"
- balance two error types:  $h_{\text{best}} \sim \epsilon_{\text{mach}}^{1/3} \sim 10^{-5}$

Reading: floating point, backward stability [GC12, Ch. 5–6] [TBI97, Ch. 12–15]

## Better approach to differentiation ( $d = 1$ )

As w/ integration: get interpolant  $\rightarrow$  differentiate it exactly [Tre00, Ch. 6]

Get  $N \times N$  matrix  $D$  acting on func. values  $\{f(x_j)\}$  to give  $\{f'(x_j)\}$ . Has simple formula

# Better approach to differentiation ( $d = 1$ )

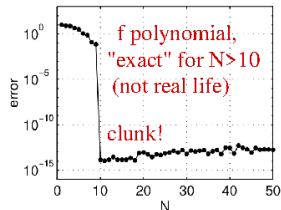
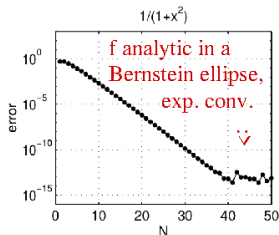
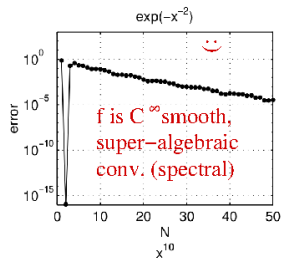
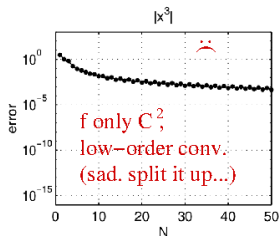
As w/ integration: get interpolant  $\rightarrow$  differentiate it exactly [Tre00, Ch. 6]

Get  $N \times N$  matrix  $D$  acting on func. values  $\{f(x_j)\}$  to give  $\{f'(x_j)\}$ . Has simple formula

Examples:

$N$  Chebychev nodes  
in  $[-1, 1]$

shown: max error in  $f'$



- for  $N$  large, dense  $D$  is never formed, instead applied via FFT

spectral solvers for ODE/PDEs. codes: chebfun, PseudoPack, dedalus... Lecture II

## Summary: we scratched the surface

Can integrate & differentiate smooth funcs given only point values  $f(x_j)$

Both follow from building a good (fast-converging) interpolant

For  $f$  smooth in 1D, can & should easily get *many* (10+) digits accuracy

### Concepts:

convergence order/rate

smoothness

spectral method

global vs local interpolant

adaptivity

rounding error & catastrophic cancellation

tensor products for 2D & 3D; higher gets harder

See references at end of slide, and come ask us stuff!



# LECTURE II: numerical differential equations

## GOALS

\*\*\*

## Reminder of types and applications of diff. eq.

- ODEs: eg pendulum  $u''(t) + \sin(u(t)) = 0$  task: solve  $u(t)$  efficiently & acc.  
needs initial conditions to kick it off eg  $u(0) = 1, u'(0) = 0$   
eg kinetics of system of well-mixed chemicals,  $\mathbf{u}(t)$  is vector
- time-independent PDEs, usually “elliptic” Mike will explain  
eg Poisson eqn  $\Delta u = g$ ,  $u(\mathbf{x}) =$  chemical concentration or electric potential  
 $\mathbf{x}$  means  $(x, y, \dots)$ ,  $\Delta u$  means Laplacian  $\partial^2 u / \partial x^2 + \partial^2 u / \partial y^2 + \dots =$  curvature of  $u$   
 $g(\mathbf{x}) =$  volume source of chemical or charge  
needs boundary conditions (BCs) on  $u$ , or decay at infinity  
eg viscous fluid flow:  $\mathbf{u}$  is velocity field, Stokes eqns  
eg what is ground state of quantum system? solving t-indep. Schrödinger  $\Delta u = Eu$
- time-dependent (evolution) PDEs  $u(\mathbf{x}, t)$  varies in space and time  
eg heat/diffusion equation  $\partial u / \partial t + \Delta u = g$   $g(\mathbf{x}, t) =$  source  
needs initial and BCs BCs may vary in time...

BCs: simple (eg periodic cube) vs complicated ( $u = 0$  on a nasty surface)

[Mike will in next talk overview the three flavors of PDE]

## 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 difference 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  $f(u^{n+1}) = \dots$

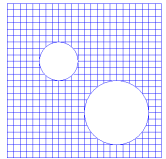
- Backward Euler: use 1st-order backward difference rule

Stability, but potentially at high cost

$$u'(t) = -\lambda u(t) \quad \lambda > 0$$

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

# Advanced finite difference techniques



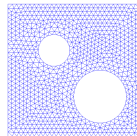
- Selecting stencils term by term (e.g. upwinding)
- Adaptive stencil selection (e.g. WENO)

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 neighbors 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:

A Green's func  $G$  (fundamental soln) needed

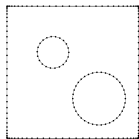
eg  $\Delta G = \delta$

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

heavily uses high-order integration methods

- see Jun Wang's talk later today!



# Summary

## Recommended accessible reading

- [GC12] A Greenbaum and T P Chartier, *Numerical methods*, Princeton University Press, 2012.
- [TBI97] L. N. Trefethen and D. Bau III, *Numerical linear algebra*, SIAM, 1997.
- [Tre00] Lloyd N. Trefethen, *Spectral methods in MATLAB*, Software, Environments, and Tools, vol. 10, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000.
- [Tre13] L. N. Trefethen, *Approximation theory and approximation practice*, SIAM, 2013,  
<http://www.maths.ox.ac.uk/chebfun/ATAP>.