Optimal solvers for partial differential equations

Ed Bueler

Dept of Mathematics and Statistics and Geophysical Institute
University of Alaska Fairbanks

DMS Colloquium 28 November, 2017

why this talk?

- I have been thinking about what are the goals of codes which numerically solve differential equations
- there are reliable black boxes for ODE IVPs
    - `ode45` in MATLAB
- what properties would a good PDE black box have?

Outline

how to approximately solve a PDE

what is an “optimal solver”?

multigrid

barriers & extensions to optimality

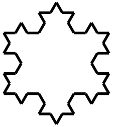
Poisson equation

- for much of this talk I'll use two example PDE problems

1. *Poisson equation* with Dirichlet boundary conditions:

$$-\nabla^2 u = f \qquad \text{on } \Omega \subset \mathbb{R}^d \text{ with } u\big|_{\partial\Omega} = g,$$

   - a linear elliptic PDE problem in dimension $d = 2$ or $d = 3$
   - recall that $\nabla^2 u = \nabla \cdot (\nabla u) = u_{xx} + u_{yy} + u_{zz}$
   - source $f(x, y, z)$ given
   - boundary values $g(x, y, z)$ given
   - will use various domains Ω including a square, a cube, and
     - a snowflake fractal



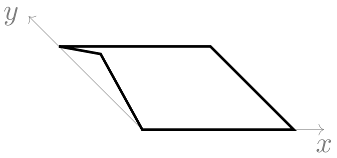
   - the solution $u(x, y, z)$ of $-\nabla^2 u = 1$ with $g = 0$ gives the expected time for a Brownian motion to first hit $\partial\Omega$

minimal surface equation

2. *minimal surface equation (MSE)* with Dirichlet b.c.s:

$$-\nabla \cdot \left(\frac{\nabla u}{\sqrt{1 + |\nabla u|^2}} \right) = 0 \qquad \text{with } u\big|_{\partial\Omega} = g.$$

- a nonlinear elliptic PDE in 2D
- here: square domain $\Omega = [0,1] \times [0,1]$
- the solution $u(x,y)$ gives the height of a zero-gravity soap bubble which spans a wire frame with height $z = g(x,y)$:



today's talk: mostly elliptic PDEs

both examples **1** & **2**

- are well-posed elliptic PDE BVPs
- seek solution u from an ∞-dimensional vector space
- *main idea*: a PDE BVP is a system of ∞ eqns in ∞ unknowns

fine print:

- both examples derivable from variational principles, thus well-posed
- the "∞-dimensional vector space" is a Sobolev space such as $H^1(\Omega)$

approximation: finite difference method

- most problems are not solvable exactly, so we
  - approximate by N equations in N unknowns
  - where $N \in \mathbb{Z}^+$ so $N \ll \infty$
- one method is *finite differences* (FDM), based on

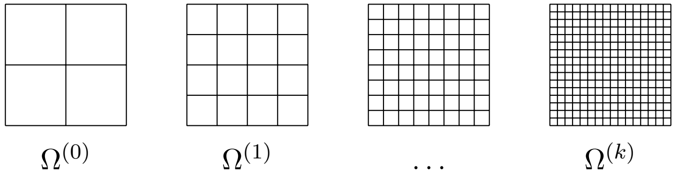
$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \approx \frac{f(x+h) - f(x)}{h}$$

- for the 2D Poisson equation:

$$u_{xx} + u_{yy} \approx \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}}{h^2}$$

structured grids notation

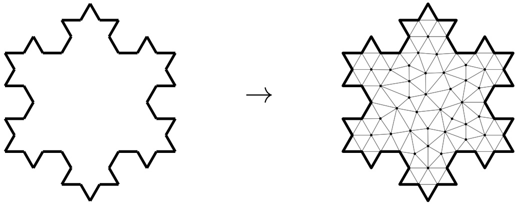
- a *structured grid* is a product of 1D grids
- consider sequence of such grids $\Omega^{(k)}$ on squares or cubes
- notation: *level* k grid has spacing h_k in each direction



- N_k is the number of equations and of unknowns
- "increasing resolution" means $h_k \to 0$ and $N_k \to \infty$
    - a.k.a. "refinement"
    - $N_k = O(h_k^{-d})$ grid points in d dimensions
    - typically: $N_{k+1} \approx 2^d N_k$

approximation: finite element method (FEM)

- FEM discretization is well-suited to unstructured meshes of arbitrary polygonal/polyhedral domains
  - e.g. triangulate the snowflake:



- FEM uses the *weak form* of the PDE
- example: for the Poisson equation it is

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v \qquad \forall v \in H_0^1(\Omega)$$

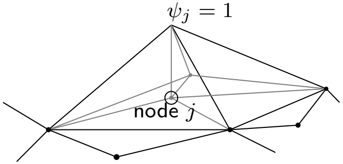
  - derivation: multiply PDE by v and integrate by parts

- Fall 2018: graduate seminar in FEM

finite element method

in more detail, the FEM uses

- a triangular/quadrilateral/tetrahedral/etc. mesh of N nodes on Ω
- an N-dimensional subspace $\mathcal{X} \subset H^1(\Omega)$
- basis of *hat functions* $\psi_j(x,y)$
    - $\psi_j(x_i, y_i) = 0$ if $i \neq j$
    - $\psi_j(x_j, y_j) = 1$



then the FEM

- defines

$$u(x,y) = \sum_j u_j \, \psi_j(x,y)$$

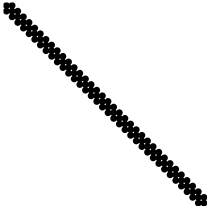
  for unknown coefficients $\mathbf{u} = \{u_j\} \in \mathbb{R}^N$ (N unknowns)
- requires the weak form to hold for all $v = \psi_i$ (N eqns)

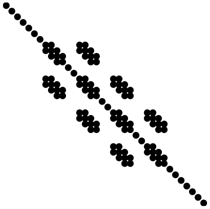
sparse matrices

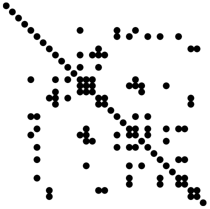
- both methods ($\mathrm{F}\genfrac{}{}{0pt}{}{\mathrm{D}}{\mathrm{E}}\mathrm{M}$) produce *sparse matrices*
- for example, the Poisson equation becomes a linear system

$$A\mathbf{u} = \mathbf{b}$$

  - $A \in \mathbb{R}^{N\times N}$ is sparse
  - A is symmetric positive definite (SPD)
- *pro tip*: Matlab's `spy(A)` shows nonzero structure


Poisson 1D (either method)


MSE FDM 2D square


Poisson FEM 2D snowflake

nonlinear PDEs make sparse matrices too

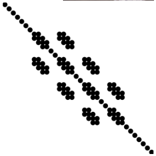
- F$\substack{\text{D}\\\text{E}}$M applied to a nonlinear elliptic PDE BVP gives a nonlinear (algebraic) system of N equations in N unknowns:

$$\mathbf{F}(\mathbf{u}) = 0$$

  - $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$ is a nonlinear function
  - call $\mathbf{F}(\mathbf{w})$ the *residual* if $\mathbf{w}$ is a guess at the solution
- usually apply Newton's method to solve:

$$J_{\mathbf{F}}(\mathbf{u}_\ell)\,\mathbf{s} = -\mathbf{F}(\mathbf{u}_\ell)$$
$$\mathbf{u}_{\ell+1} = \mathbf{u}_\ell + \mathbf{s}$$

  - $J_{\mathbf{F}}(\mathbf{w}) \in \mathbb{R}^{N\times N}$ is the Jacobian of $\mathbf{F}$
  - it is a sparse matrix (right)


MSE FDM 2D square

Outline

how to approximately solve a PDE

what is an “optimal solver”?

multigrid

barriers & extensions to optimality

define "optimal"

- consider N equations in N unknowns: $\mathbf{F}(\mathbf{u}) = 0$
  - residual $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$ is generally nonlinear
  - $\mathbf{F}(\mathbf{w}) = \mathbf{b} - A\mathbf{w}$ in the linear case

definition. an algorithm which solves $\mathbf{F}(\mathbf{u}) = 0$ in $O(N)$ work, as $N \to \infty$, is *optimal*

- if you have ever tried solving big, nontrivially-coupled systems of equations, you'll conclude optimality is generally hopeless

slide full of caveats

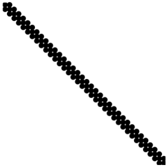
in the definition "an algorithm which solves $\mathbf{F}(\mathbf{u}) = 0$ in $O(N)$ work, as $N \to \infty$, is *optimal*":

- "solves" means: generates $\mathbf{u}_n$ so that $\frac{\|\mathbf{F}(\mathbf{u}_n)\|}{\|\mathbf{F}(\mathbf{u}_0)\|} \le$ tol
    - where $\mathbf{u}_0$ is an initial guess
    - in linear case $A\mathbf{u} = \mathbf{b}$, given any rounding error, only $O(\kappa(A)\epsilon_{\text{mach}})$ accuracy is possible anyway[1]
- "$O(N)$" hides constant; may depend on tol but not on N
- "work" $=$ (count of floating point operations)
    - or runtime, but timing on modern computers is really messy
- "$N \to \infty$" limit is notional only
    - real computers run out of memory
    - optimal algorithms are often memory-limited (that's a *feature*)

[1] for students of MATH 614

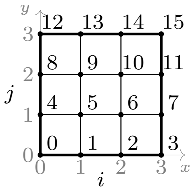
example: tridiagonal matrices

- an easy example of optimality
- Gauss elimination solves $A\mathbf{u} = \mathbf{b}$ in $8N - 6 = O(N)$ flops
    - need to avoid pivoting
- for SPD tridiagonal matrices, use Cholesky decomposition, again $O(N)$
- the 1D Poisson problem $-u'' = f$, and generally all ODE BVPs, have optimal solution methods

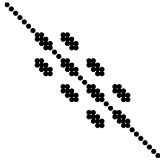


non-example: banded direct methods in 2D,3D

- for structured-grid FDM method on PDEs in 2D and 3D the bandwidth of A grows as $N \to \infty$
- for example, for MSE on $\Omega = [0,1] \times [0,1]$:



$\rightarrow$



- if bandwidth p then Cholesky does $O(Np^2)$ work
- thus for direct methods for PDE problems on structured grids with m points in each direction:
    - $N = m^2$ and $p = m$ so $O(N^2)$ work in 2D
    - $N = m^3$ and $p = m^2$ so $O(N^{7/3})$ work in 3D
- variable reordering like "minimum degree" helps . . . but not enough

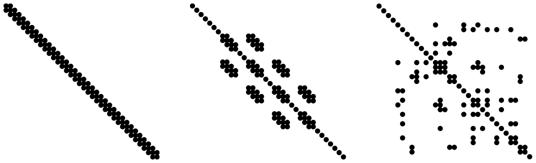
sparse matrices from PDEs have $O(N)$ mat-vec

- if
  - $A \in \mathbb{R}^{N\times N}$ is sparse, with
  - number of nonzeros per row bounded independent of N

  then the *work of computing* $A\mathbf{v}$ *is* $O(N)$
- computing $A\mathbf{v}$ is called a “mat-vec”
- condition is automatic for structured grids and any F$\genfrac{}{}{0pt}{}{\text{D}}{\text{E}}$M
- for typical FEM on an unstructured mesh,

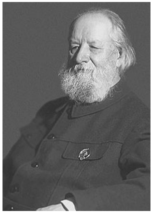
$$(\text{nonzeros in row } j \text{ of } A) = \operatorname{degree}(\text{node } j) + 1$$

so cost of $A\mathbf{v}$ is $O((\max\operatorname{degree})N)$


nonzeros per row $= 3$ $\qquad$ 9 $\qquad$ $\max\operatorname{degree} + 1$

Krylov methods

- for most numerical analysts of the 1980s and 1990s, "sparse mat-vecs are $O(N)$" was the new hope
- because naval engineer A. Krylov (1931) observed that the solution to $A\mathbf{u} = \mathbf{b}$ may be well-approximated by $\mathbf{v}$ in



$$\mathcal{K}_m(A, \mathbf{b}) = \operatorname{span}\left\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \dots, A^m\mathbf{b}\right\}$$

  - computing $\mathbf{v} \in \mathcal{K}_m(A, \mathbf{b})$ costs $O(mN)$
  - $\mathbf{v} \in \mathcal{K}_m(A, \mathbf{b}) \iff \mathbf{v} = p_m(A)\mathbf{b}$
- to solve $A\mathbf{u} = \mathbf{b}$ we want $\mathbf{u} = A^{-1}\mathbf{b} \approx p_m(A)\mathbf{b}$

conjugate gradients

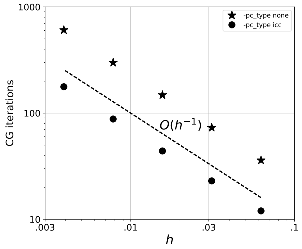
- example: *conjugate gradients* (CG) is a Krylov method
  - A must be SPD
  - CG generates the "best" iterates $\mathbf{u}_m$ from a Krylov space
    - the error $\mathbf{e}_m = \mathbf{u}_m - \mathbf{u}$ is minimal in norm $\|\cdot\|_A$
  - work per CG iteration is $O(N)$
  - thus work is $O(mN)$ for m iterations
- if $\kappa = \kappa_2(A)$ is the condition number of A then

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} \le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m$$

- it follows that $m = O(\sqrt{\kappa})$ iterations are needed

CG iterations increase with N

- unfortunately, if A is from FDM applied to the Poisson equation then
$$\kappa_2(A) = O(h^{-2})$$
- the number of CG iterations m increases with $N = O(h^{-d})$
    - $m = O(N^{1/2})$ in 2D $\implies O(N^{3/2})$ solver
    - $m = O(N^{1/3})$ in 3D $\implies O(N^{4/3})$ solver



- other Krylov methods are similar
- general reminder:

  a Krylov method can only be optimal if the number of iterations m is bounded independently of N

preconditioning

- by $\sim$1995 it was clear that Krylov methods by themselves were not the answer for solving big PDE problems
- but you don't have to accept the given system $A\mathbf{u} = \mathbf{b}$
- *definition*. given invertible M,
    - $(M^{-1}A)\mathbf{u} = M^{-1}\mathbf{b}$ is the *left-preconditioned* system
    - $(AM^{-1})M\mathbf{u} = \mathbf{b}$ is the *right-preconditioned* system
- new condition numbers $\kappa_2(M^{-1}A)$ or $\kappa_2(AM^{-1})$ can be much smaller
- "M^{-1}" must be a cheap method for this to help
    - e.g. Meijerink & van der Vorst (1977): incomplete LU and Cholesky factorizations

where we stand: an optimality lemma

- *lemma*. as $N \to \infty$, if $A \in \mathbb{R}^{N\times N}$ is SPD and if a symmetric preconditioning method produces bounded condition numbers,

  $$\kappa_2(M^{-1}A) \le B,$$

  where $B > 0$ is independent of N, then preconditioned CG is an optimal solver

- *optimality goal*: find preconditioners which make $\kappa_2(M^{-1}A)$ bounded independent of N

- *multigrid* is such a preconditioner

Outline

how to approximately solve a PDE

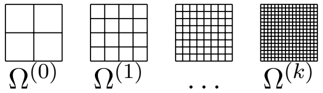
what is an "optimal solver"?

multigrid

barriers & extensions to optimality

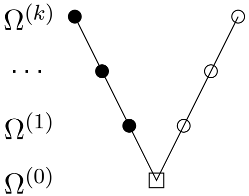
multigrid: what it does

- given sequence of grids $\{\Omega^{(j)}\}_0^k$



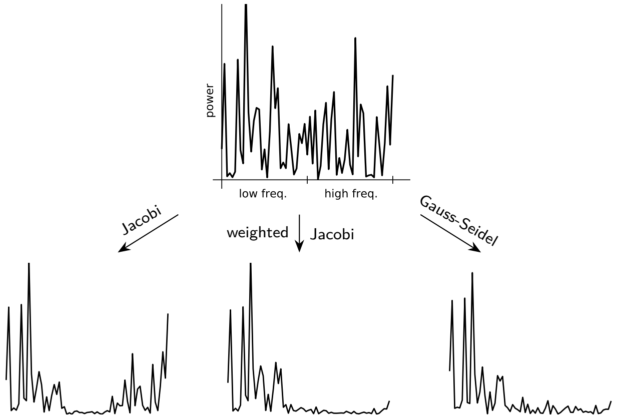
- given initial guess $\mathbf{w}$ on the finest grid $\Omega^{(k)}$, a multigrid cycle (e.g. a "V cycle") approximately solves $A\mathbf{u} = \mathbf{b}$

```
function $\textsc{Vcycle}(A, \mathbf{b}, \mathbf{w}, l)$
    if $l == 0$ then
        solve $A\mathbf{v} = \mathbf{b}$, e.g. by a direct solver
    else
        improve $\mathbf{w}$ on the level $l$ grid
        $\mathbf{r}^C = \big($restrict $\mathbf{r} = \mathbf{b} - A\mathbf{w}$ to $\Omega^{(l-1)}\big)$
        $\mathbf{z}^C = \textsc{Vcycle}(A^C, \mathbf{r}^C, 0, l-1)$
        $\mathbf{v} \leftarrow \mathbf{v} + \big($interpolate $\mathbf{z}^C$ to $\Omega^{(l)}\big)$
        improve $\mathbf{v}$ some more on the level $l$ grid
    return $\mathbf{v}$
```



multigrid uses cheap smoothers

- *question*: what does "improve $\mathbf{w}$ on the level l grid" mean?
  answer: smoothing
- many classical linear iterations are smoothing
  - e.g. weighted Jacobi and Gauss-Seidel using A
  - fast $O(N)$ operations
  - single iteration reduces high-frequency components of the error



multigrid: why it is $O(N)$

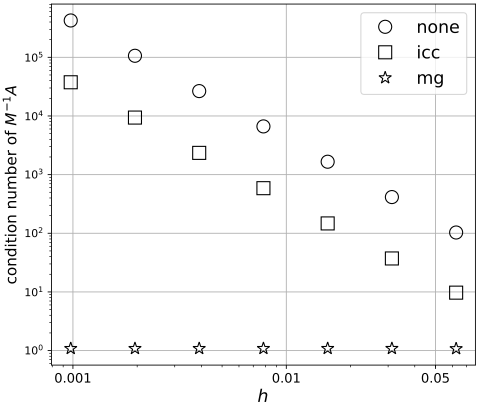
- multigrid is a systematic way of combining two actions
    - *smoothing*: filter out high frequencies of the error on your grid
    - *coarsening*: transfer to an easier grid
        - frequencies that were "medium-low" are now "high"
- restricting and interpolating on $\Omega^{(l)}$ is $O(N_l)$
- a single step of the smoother on $\Omega^{(l)}$ is $O(N_l)$
- thus total work on $\Omega^{(l)}$ is CN_l for fixed C
- then the total work of a V-cycle is a finite geometric series:

$$\begin{aligned}
&(\Omega^{(k)} \text{ work}) + (\Omega^{(k-1)} \text{ work}) + \cdots + (\Omega^{(1)} \text{ work}) + (\Omega^{(0)} \text{ work}) \\
&\quad = CN_k + CN_{k-1} + \cdots + CN_1 + C_0 \\
&\quad = CN_k + C\frac{N_k}{2^d} + \cdots + C\left(\frac{1}{2^d}\right)^{k-1} N_k + C_0 \\
&\quad \le 2CN_k + C_0
\end{aligned}$$

- re the coarsest grid ... who cares! ... C_0 is independent of N

multigrid on Poisson

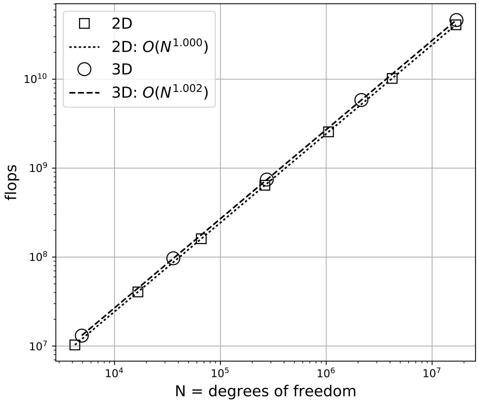
- V-cycle-preconditioned CG iterations on $\Omega = [0,1]^2$ Poisson



values of $\kappa_2(M^{-1}A)$ for 2D problem

multigrid on Poisson: evidence of optimality

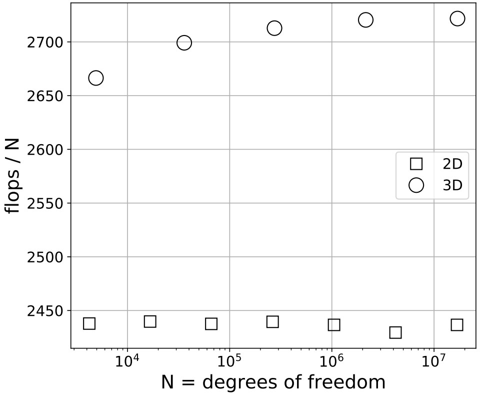
- V-cycle-preconditioned CG iterations on $\Omega = [0,1]^d$ Poisson



direct demonstration of $O(N)$ work

multigrid on Poisson: evidence of optimality

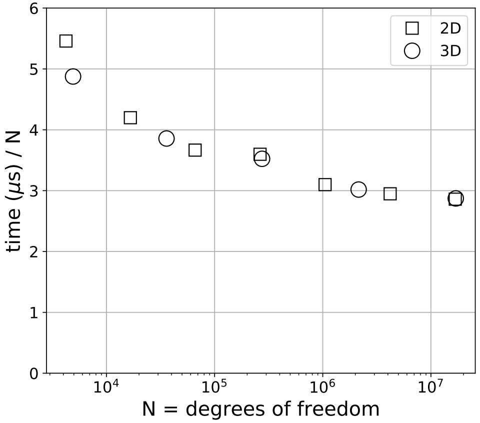
- V-cycle-preconditioned CG iterations on $\Omega = [0,1]^d$ Poisson



i.e. constant amount of work per degree of freedom

multigrid on Poisson: evidence of optimality

- V-cycle-preconditioned CG iterations on $\Omega = [0,1]^d$ Poisson



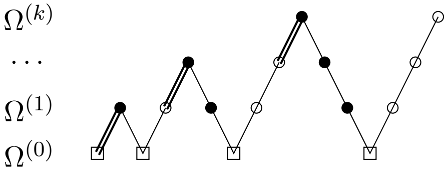
(almost) constant amount of time per degree of freedom

multigrid on MSE

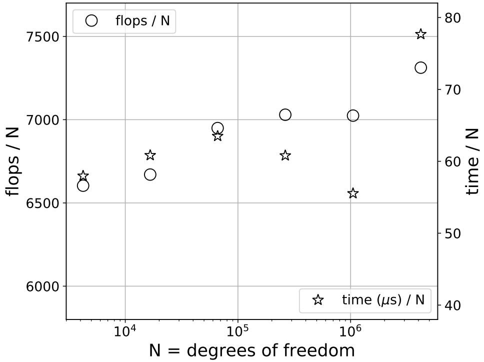
- recall the nonlinear MSE problem on $\Omega = [0,1]^2$:

$$-\nabla \cdot \left(\frac{\nabla u}{\sqrt{1 + |\nabla u|^2}} \right) = 0 \qquad \text{with } u\big|_{\partial\Omega} = g.$$

    - solved by Newton iteration
    - how to *find a convergent initial iterate* on a fine grid?
- multigrid solution is by nonlinear "F-cycle"
    - a.k.a. nested iteration with V-cycles
    - *start* on coarse grid $\Omega^{(0)}$
    - interpolating upward supplies good initial iterate



multigrid on MSE: evidence of optimality



- on finest 2049×2049 grid with $N = 4 \times 10^6$:

$$\text{total flops} = N \left(\frac{\text{flops}}{N}\right) = (4 \times 10^6)(7 \times 10^3) \approx 3 \times 10^{10}$$

  - runtime about 5 minutes total

algebraic multigrid

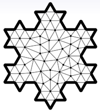
- what about multigrid on unstructured grids?
    - remember the snowflake?
    - for "geometric" multigrid (used so far) one need subgrids and grid-based restriction and interpolation operations
- new idea $\sim$1985 is *algebraic multigrid*
    - can be applied to *any* linear system $A\mathbf{u} = \mathbf{b}$
    - extracts analogs of "subgrid" and "smoother" and "interpolation" from A itself
    - . . . but tends to need elliptic PDEish properties to actually work
    - active research area for e.g. spectral graph theory
    - by a new generation of applied mathematicians

algebraic multigrid

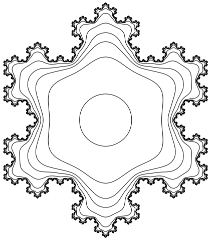
- what about multigrid on unstructured grids?
  - remember the snowflake?
  - for "geometric" multigrid (used so far) one need subgrids and grid-based restriction and interpolation operations
- new idea ∼1985 is *algebraic multigrid*
  - can be applied to *any* linear system $A\mathbf{u} = \mathbf{b}$
  - extracts analogs of "subgrid" and "smoother" and "interpolation" from A itself
  - . . . but tends to need elliptic PDEish properties to actually work
  - active research area for e.g. spectral graph theory
  - by a new generation of applied mathematicians

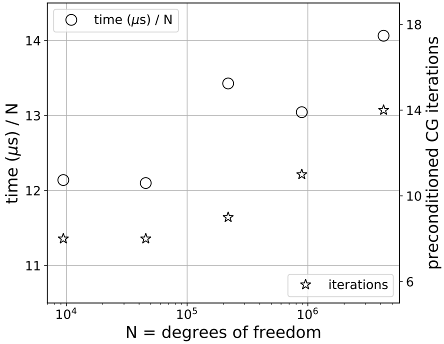
algebraic multigrid: evidence of (near-)optimality

- recall unstructured mesh on a snowflake polygon; level 2 at right



- generate level 5,6,7,8,9 approximations of the Koch fractal
- mesh them
- and test algebraic-multigrid-preconditioned CG method, on the Poisson equation $-\nabla^2 u = 1$, for optimality





wider applicability of multigrid

- multigrid was invented for Poisson and linear elliptic equations by Federenko (1962, 1964)
- but there was a period of darkness
- by 1980 or so optimism about multigrid was limited to one mathematician: Achi Brandt
  - a creator of algebraic multigrid,
  - and of a fully-nonlinear multigrid, the *full approximation scheme* (not covered here),
  - who started calling optimality "textbook multigrid efficiency," which isn't really helping
- since then, multigrid has succeeded on more and more applications
  - example: Brown et al. (2013), *Achieving textbook multigrid efficiency for hydrostatic ice flow*

Achi Brandt is my hero

Jed Brown, UAF MS 2006

Outline

how to approximately solve a PDE

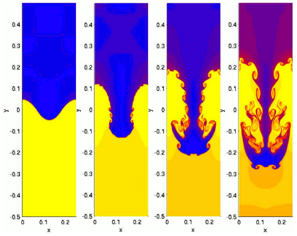
what is an "optimal solver"?

multigrid

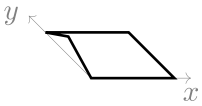
barriers & extensions to optimality

low regularity of solution

- multigrid is not always easy to make optimal
- consider nonlinear PDE BVP
    - or a PDE system like Stokes flow
    - or an implicit time step of Navier-Stokes

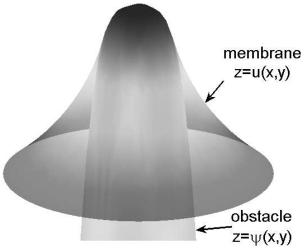


- try Newton-multigrid method
    - as we did with MSE
- *problem*: often # of Newton steps *and* Krylov steps grows as $h \to 0$ because of large gradients in the solution
- can demonstrate this with MSE for nonsmooth boundary conditions
    - not shown
    - Brandt suggests: combine multigrid & AMR



constrained problems

- other problems are not quite PDEs because they have inequality constraints
- causes two difficulties for Newton-multigrid methods:
  1. a free boundary implies low regularity (last slide)
  2. # of Newton steps is proportional to D/h where D is distance-to-move free boundary (from initial iterate)
- but there's another Brandt invention: *projected full approximation scheme*, a fully-nonlinear and constraint-adapted multigrid
- Max H. is working on it



numerical convergence, and spectral methods

- for a PDE BVP, as $h \to 0$ and $N \to \infty$ we first want convergence to the continuum solution
  - I have been assuming that our methods are convergent!
- spectral methods get very close to the continuum solution for very small N . . . compared to F$^{D}_{E}$M
  - *if* the geometry is simple (rectangle)
  - *and* the solution is smooth
- often A is dense
- but with such a spectral method, even $O(N^3)$ solution of the discrete equations is often acceptable because N is small

- recalling these things shows that optimality is *not* the only good goal

optimality, by numerical DE subfield

- for all DE problems, one needs to define N, the number of discrete degrees of freedom, before we can talk optimality
- ODE IVP
  - for $y' = f(t, y)$ and $y(t) \in \mathbb{R}^q$
  - on $[0, T]$ with Δt spacing in time
  - define:

$$N := \frac{T}{\Delta t}\, q$$

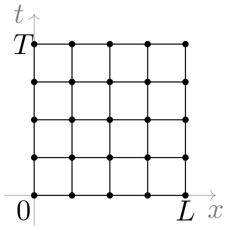
  - with this N, all ODE IVP methods are already optimal

optimality, by numerical DE subfield

- for all DE problems, one needs to define N, the number of discrete degrees of freedom, before we can talk optimality
- ODE IVP — already optimal ✓
- ODE BVP
  - F$^{\mathsf{D}}_{\mathsf{E}}$M generate tridiagonal systems
  - already optimal

optimality, by numerical DE subfield

- for all DE problems, one needs to define N, the number of discrete degrees of freedom, before we can talk optimality
- ODE IVP — already optimal ✓
- ODE BVP — already optimal ✓
- hyperbolic PDE IBVP
  - for example, wave equation with reaction: $u_t + a(u) \cdot \nabla u = f(u)$
  - F $\genfrac{}{}{0pt}{}{\text{D}}{\text{V}}$ Ms normally use explicit, CFL-limited time steps Δt
  - if we define $N = \frac{T}{\Delta t}\frac{L}{\Delta x}$, these methods are already optimal

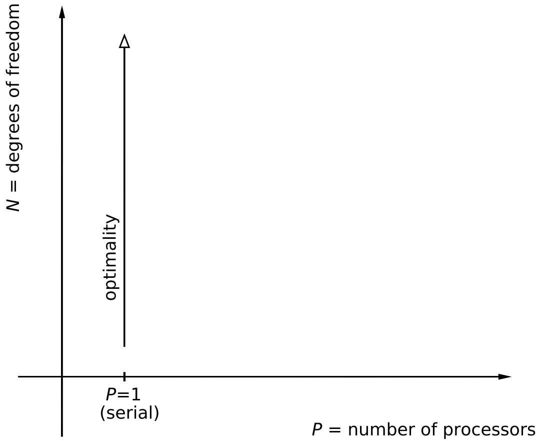


optimality, by numerical DE subfield

- for all DE problems, one needs to define N, the number of discrete degrees of freedom, before we can talk optimality
- ODE IVP — already optimal ✓
- ODE BVP — already optimal ✓
- hyperbolic PDE IBVP — already optimal ✓
- PDE BVP — optimal requires effort
- other PDE IBVP — optimal requires effort
  - for example, advection-diffusion-reaction equation with reaction: $u_t + a(u) \cdot \nabla u = \nabla \cdot (D \nabla u) + f(u)$

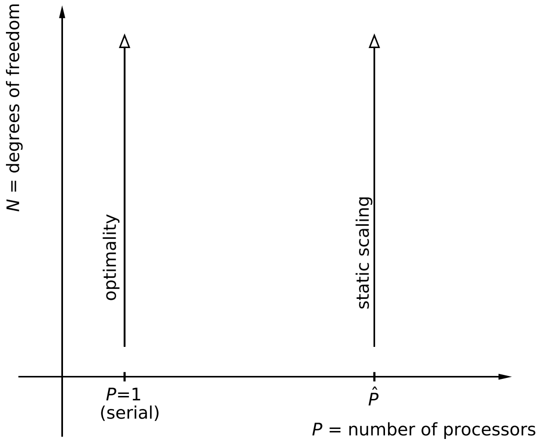
conflicting goal: parallel scaling

- sometimes you have a big machine with $\hat{P}$ processors
  - everything so far has been serial $(P = 1)$
  - on graph below, runtime is third axis



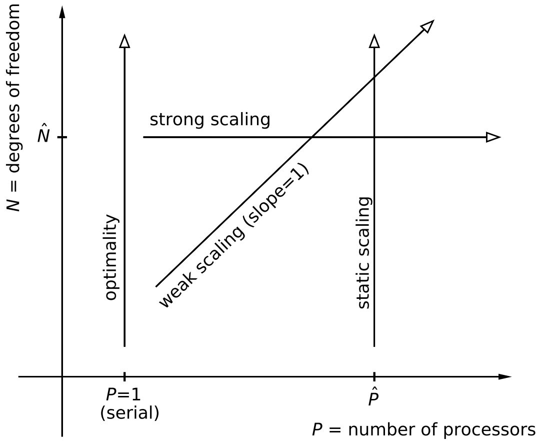
conflicting goal: parallel scaling

- sometimes you have a big machine with $\hat{P}$ processors
    - everything so far has been serial $(P = 1)$
    - on graph below, runtime is third axis



conflicting goal: parallel scaling

- sometimes you have a big machine with $\hat{P}$ processors
  - everything so far has been serial $(P = 1)$
  - on graph below, runtime is third axis



conclusion

- in approximately solving your well-behaved PDE-type problem on a modern computer,

  you should be solving the N equations in $O(N)$ work

  as you head toward $N = \infty$ unknowns
- this is a good *goal* for PDE BVPs
- to achieve it you must exploit
  - the locality (sparsity) of the problem, and
  - correlation (smoothness) of the solution
- multigrid is the only real hope!?
- talk to me about PETSc . . . I am writing a book about that

conclusion

- in approximately solving your well-behaved PDE-type problem on a modern computer,

  you should be solving the N equations in $O(N)$ work

  as you head toward $N = \infty$ unknowns
- this is a good *goal*[2] for PDE BVPs
- to achieve it you must exploit
  - the locality (sparsity) of the problem, and
  - correlation (smoothness) of the solution
- multigrid is the only real hope!?
- talk to me about PETSc . . . I am writing a book about that

[2]for dense systems $A\mathbf{u} = \mathbf{b}$ the goal is $O(N^2)$ . . . an open problem