Multiscale Methods in Scientific Computation

or

How I Learned to Stop Worrying and Love Multigrid

Scott MacLachlan

Scott.MacLachlan@colorado.edu

Department of Applied Mathematics

University of Colorado at Boulder

Outline

- Motivation for Scientific Computation
- Differential Equations and Linear Algebra
- Traditional Linear Solvers
- Motivation for Multilevel Techniques
- Overview of Multilevel Techniques
- Multigrid

Terminology

- h - Mesh spacing, typically small
- $n = 1/h$ - grid dimensions, typically large
- N - problem size, typically n^2 or n^3
- $O(h^\alpha)$ - a measure of smallness, for large α, an $O(h^\alpha)$ quantity is very small
- $O(n^\alpha)$, $O(N^\alpha)$ - a measure of largeness, for large α, an $O(n^\alpha)$ quantity is quite large
- L^2 - the class of functions which are square integrable
- H^p - the class of square integrable functions with p derivatives which are all square integrable

Why Compute?

- Interested in modelling physical processes

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport
  - Elastic Materials

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport
  - Elastic Materials
- Can describe these processes through differential equations (both ODEs and PDEs)

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport
  - Elastic Materials
- Can describe these processes through differential equations (both ODEs and PDEs)
- Cannot write down closed form solutions

Why Compute?

- Interested in modelling physical processes
  - Diffusion (Heat, Energy, Chemical)
  - Fluid Flow
  - Particle Transport
  - Elastic Materials
- Can describe these processes through differential equations (both ODEs and PDEs)
- Cannot write down closed form solutions
- Need to find (approximate) solutions in other ways

Why Compute?

- Computers are good at arithmetic manipulation of large amounts of data

Why Compute?

- Computers are good at arithmetic manipulation of large amounts of data
- Computers are good at visualization of complex structures

Why Compute?

- Computers are good at arithmetic manipulation of large amounts of data
- Computers are good at visualization of complex structures
- Need to be able to rewrite a differential equation as a problem that a computer can handle

Discretization of Differential Equations

- We are interested in rewriting a differential equation in a way that a computer can solve it

Discretization of Differential Equations

- We are interested in rewriting a differential equation in a way that a computer can solve it
- Differential equations give relationships between points that are infinitesimally close together

Discretization of Differential Equations

- We are interested in rewriting a differential equation in a way that a computer can solve it
- Differential equations give relationships between points that are infinitesimally close together
- To *discretize* a differential equation, we consider points of distance h apart instead

Finite Differences

- To discretize a differential equation, need to approximate derivatives
- Taylor Series give us a way:

$$u(x+h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3}u'''(x) + O(h^4)$$

$$u(x-h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3}u'''(x) + O(h^4)$$

Finite Differences

- So

$$u'(x) = \frac{u(x+h) - u(x)}{h} + O(h)$$
$$= \frac{u(x) - u(x-h)}{h} + O(h)$$
$$u''(x) = \frac{u(x-h) - 2u(x) + u(x+h)}{h^2} + O(h^2)$$

- In a similar way, we can approximate higher order derivatives
- Also can do partial derivatives

Finite Elements

- Redefine what it means for a function to solve a differential equation
- Allows theory for differential equations to be used in linear systems (e.g. existence and uniqueness)

Properties of Discretizations

- The matrices from discretizations tend to be
  - Sparse (number of nonzeros per row doesn't change with n)
  - Ill-conditioned
  - Symmetric (if DE is)
  - Positive-Definite (if DE is)

Dimensionality

- Systems become larger as we increase the number of dimensions.
- Suppose in 1 dimension we need n data points, then in 2 dimensions we'll need $n \times n = n^2$ and in 3 dimensions we'll need $n \times n \times n = n^3$.
- We're interested in solving 3 dimensional problems!

Classical Methods: Gaussian Elimination

- Write down equations in order
- Use first equation to eliminate first variable from all other equations
- Use second equation to eliminate second variable from all other equations
- $\vdots$
- Total cost for an $N \times N$ system: $O(N^3)$
- Total cost for an $n^3 \times n^3$ system: $O(n^9)$

Classical Methods: Jacobi

- Rewrite each equation:

$$(Ax)_i = b_i$$

$$a_{ii}x_i = b_i - \sum_{j \neq i} a_{ij}x_j$$

- Solve each equation for x_i using values of x_j from previous step
- Cost per "sweep": $O(N)$ if system is sparse
- Number of sweeps needed for convergence: $O(n^2)$
- Total cost for an $n^3 \times n^3$ system: $O(n^5)$

Classical Methods: Gauss-Seidel

- Rewrite each equation:

$$a_{ii}x_i = b_i - \sum_{j \neq i} a_{ij}x_j$$

- Solve each equation successively for x_i using values of x_j from previous equations
- Cost per "sweep": $O(N)$ if system is sparse
- Number of sweeps needed for convergence: $O(n^2)$ (but fewer than Jacobi)
- Total cost for an $n^3 \times n^3$ system: $O(n^5)$

Classical Methods: SOR

- Rewrite each equation:

$$a_{ii} y_i = b_i - \sum_{j \neq i} a_{ij} x_j$$

- Solve each equation successively for y_i using values of x_j from previous equations
- Choose new x_i to be an average of old x_i and y_i:

$$x_i^{\text{new}} = \omega y_i + (1 - \omega) x_i^{\text{old}}$$

- Cost per "sweep": $O(N)$ if system is sparse
- Number of sweeps needed for convergence: $O(n)$ with optimal ω
- Total cost for an $n^3 \times n^3$ system: $O(n^4)$

Krylov Subspace Methods

- Based on projections onto increasingly larger subspaces of $\mathbb{R}^N$
- Cost depends on cost of evaluating Ax, which is low ($O(N)$) for sparse matrices
- Many steps often necessary to reach solution, however

The Curse of Dimensionality

- All of the above methods take a number of operations proportional to N raised to some power $\alpha > 1$
- We're interested in problems where $N = n^3$
- $N^\alpha = n^{3\alpha}$ or $Nn^\alpha = n^{3+\alpha}$
- If we want to use Gaussian elimination to solve a 3-dimensional problem with 100 gridpoints in each direction, we'll need on the order of 10^{18} operations! Fastest computers perform about 10^{13} operations per second, so this would take days
- For 1000 gridpoints in each direction, need years!

The Curse of Dimensionality

- Jacobi and Gauss-Seidel also also scale poorly - an increase of a factor of 10 gridpoints in each direction results in an increase of 10^5 in computation time
- SOR scales better, but need to compute optimal ω - this is difficult for general problems (involves finding eigenvalues of a matrix)

What's Wrong?

- Gaussian Elimination is expensive because it doesn't use any information about the system
- Iterative methods use information about the system, but are most effective at reducing only a certain portion of the error
- Most effort in iterative methods is devoted to resolving a few components of the solution (more on this later)
- **McCormick-Brandt Conjecture**: The amount of work an algorithm does should be proportional to the amount of error reduction it produces
- *"Stalling numerical processes must be wrong"*

Hints from Physics

- From Physics (and Chemistry and Fluids and . . .): we derive differential equations - equations which give relations between points that are infinitesimally close together
- Finite propagation speeds mean that the influence of far away points is somehow smaller than nearby points
- Our numerical methods weight the influence of all connected points the same
- Physics seems to suggest that we need to spend more time on the “local” scale than on the “global” scale
- Our understanding of the universe in inherently multiscale!

Exact Multiscale Methods - Fourier

- Given a differential equation, consider solution space
- Typically this is either H^1 or H^2, but for "nice" boundaries and forcing, $u(x)$ is certainly in L^2
- A basis for L^2 is $\left\{e^{ikx}\right\}$
- Write

$$u(x) = \sum_{k=-\infty}^{+\infty} c_k e^{ikx}$$

Exact Multiscale Methods - Fourier

- This is an effective solution technique because (in general) the different basis elements decouple
- Communication between different frequencies in a Fourier expansion is limited
- Result is an efficient solution method

Exact Multiscale Methods - Wavelets

- $\{e^{ikx}\}$ is not the only basis for L^2
- Main disadvantage is that while the Fourier basis localizes in frequency, it does not in localize in space
- Wavelets are types of orthonormal bases which localize in space (and to various degrees in frequency)
- Construct wavelets using a shift-and-scale approach - all basis functions have the same profile

Example - Haar Wavelets

- Define

$$h(x) = \begin{cases} -1 & 0 \leq x < 0.5 \\ 1 & 0.5 < x \leq 1 \\ 0 & \text{else} \end{cases}$$

- Then take $h_{jk}(x) = 2^{j/2} h(2^j x - k)$
- $\{h_{jk}(x)\}$ forms a complete orthonormal basis for L^2
- Since basis functions are orthonormal, also have limited communication between scales

The Truth

- While $\{e^{ikx}\}$ and $\{h_{jk}(x)\}$ form complete orthonormal bases for L^2, generally there is no closed form for the coefficients of the basis expansion
- Instead, use a finite sum to approximate infinite sum
- Systems for the finite number of coefficients needed for this approximation are generally “easy” to solve - can be done in $O(N \log(N))$ time

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate
- In particular, components which are geometrically smooth are reduced at the slowest rate

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate
- In particular, components which are geometrically smooth are reduced at the slowest rate
- That is, the global components of the solution are slowest to resolve

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate
- In particular, components which are geometrically smooth are reduced at the slowest rate
- That is, the global components of the solution are slowest to resolve
- These components, however, can be represented on a coarser resolution

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate
- In particular, components which are geometrically smooth are reduced at the slowest rate
- That is, the global components of the solution are slowest to resolve
- These components, however, can be represented on a coarser resolution
- To speed up solution: Do simple iterations until they stall, then transfer problem to a coarse grid and solve it there

Another Way - Multigrid

- Careful analysis of Jacobi and Gauss-Seidel iterations reveal that not all error components are reduced at the same rate
- In particular, components which are geometrically smooth are reduced at the slowest rate
- That is, the global components of the solution are slowest to resolve
- These components, however, can be represented on a coarser resolution
- To speed up solution: Do simple iterations until they stall, then transfer problem to a coarse grid and solve it there
- How do you solve the coarse grid problem? Recursion

Multigrid Strategy

- After relaxation remaining error in solution is geometrically smooth
- If x^* is the approximate solution after relaxation, then $x - x^*$ is the smooth vector
- But, $A(x - x^*) = Ax - Ax^* = b - Ax^*$, or $Ae = r$
- Since e is smooth, we can represent it on a smaller space and (approximately) solve for it there (cheaper than doing this on the full space)
- Then, a better approximation to the solution will be $x^* + e$

Complementarity

- We use 2 different processes to reduce error in Multigrid: Relaxation and Coarse Grid Correction
- Relaxation (Jacobi or Gauss-Seidel, not SOR) effectively damps oscillitory error
- Coarse Grid Correction effectively resolves smooth error
- By alternating between the two, we can quickly resolve all components and drive the error to zero

Multigrid Cost

- Relaxation on a level with N unknowns cost $O(N)$ operations
- Transfer from a level with N unknowns to a level with N/k unknowns and back costs $O(N)$ operations
- Cost per V-cycle is then

$$\sum_{j=0}^{\log_k N} c\frac{N}{k^j} \leq \sum_{j=0}^{\infty} c\frac{N}{k^j} = \frac{cN}{1-1/k}$$

- Typically k is 2 in 1-D, 4 in 2-D, 8 in 3-D

Multigrid Convergence

- Theory applies to Multigrid using Finite Elements, although it works just as well on finite differences
- For finite element discretization of $Lu = f$, get weak form $a(u, v) =< f, v >$
- If $a(u, v)$ is symmetric and H^1-elliptic, then Multigrid will reduce the error to a fixed tolerance in a finite number of V-cycles (fixed for a given tolerance)
- That is, Multigrid solves the finite element discretization of $Lu = f$ in $O(N)$ operations

Multigrid Variants

- Algebraic Multigrid - for algebraic problems similar to discretizations of scalar elliptic PDEs
- Smoothed Aggregation Multigrid - for problems similar to discretizations of some systems of elliptic PDEs (linearized elasticity in particular)
- Self-Correcting AMG - current research at CU-Boulder in extending class of problems which multigrid methods can handle
- Multigrid for Hyperbolic PDEs - current research at CU-Boulder in extending class of problems which multigrid methods can handle

Summary

- Classical methods of solving linear systems arising from discretized differential equations do not scale as the problem size or number of dimensions increases
- Multiscale methods require only $O(N \log(N))$ or $O(N)$ operations
- Multiscale methods (much) more complicated to construct
- Particularly effective for elliptic problems, but class of suitable problems is growing
- Many varieties - best is often a hybrid tuned to a particular problem