Using Multigrid to Solve Time Dependent PDEs

Matthew W. Choptuik

CIAR Cosmology and Gravity Program

Department of Physics and Astronomy
University of British Columbia
Vancouver BC

MPI-AEI, Postdam Germany

SFB Videoseminar
Golm, Germany

February 4, 2008

For more detailed notes on multigrid, see
`http://laplace.physics.ubc.ca/~matt/Teaching/06Mexico/mexico06.pdf`

Outline

- Motivation
- Review of multigrid (MG) for elliptic problems
- Application of multigrid to a model parabolic problem
- Summary & Comments

Motivation

- From time to time encounter time dependent PDEs in numerical relativity and related fields that are “stiff”; i.e. whose solutions have a large dynamic range in intrinsic time-scales (perhaps unbounded in the continuum limit)
- Frequently (but not always) these systems are of “parabolic” type
- Examples include
  - Schrödinder equations appearing in treatment of Newtonian boson stars
  - Certain type of coordinate conditions for lapse and shift (driver conditions)
  - Geometrically-motivated PDEs other than Einstein’s equations, e.g. Ricci flow

Motivation

- Assume that finite difference (FD) techniques are being used: stiffness implies that time-implicit methods will be needed to avoid unnecessarily stringent restrictions on time step, Δt in terms of the spatial coordinate mesh spacings $\Delta x^i, i = 1, \ldots d$ (assume $\Delta x^i = O(h)$ for all i)

- Key goal: Assuming typical number of grid points per edge of spatial computational domain is $n \sim h^{-1}$, so that total number of points in spatial mesh is $N \sim n^d$, want methods that can

  1. Solve discrete equations with $O(N)$ work per time step (optimal from computational complexity point of view)
  2. Allow for large time steps, i.e. $\Delta t \sim h$, especially if stiff equations are being solved in concert with hyperbolic equations

- Multigrid techniques provide basis for such methods, and are applicable to *general* systems of parabolic nature.

- To understand how this works, best to start with multigrid as applied to *time-independent* PDEs, i.e. elliptic PDEs

Model elliptic problem

- Canonical model problem: 2-D Poisson equation

$$\nabla^2 u(x,y) \equiv u_{xx} + u_{yy} = f(x,y) \tag{1}$$

on the unit square

$$\Omega : 0 \leq x \leq 1\,,\ 0 \leq y \leq 1 \tag{2}$$

with (homogeneous) Dirichlet boundary conditions

$$u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0 \tag{3}$$

and $f(x,y)$ a specified function

Discretization of model problem

- Adopt *uniform* discretization: single, constant mesh spacing, h, in each coordinate direction

- Finite difference grid, Ω^h, has n grid points in each direction, $h = 1/(n-1)$; total number of points in discretization: $N \approx n^2$.

- Finite difference mesh points are defined by

$$\{(x_i, y_j) \equiv ((i-1)\,h, (j-1)\,h)\,,\ i, j = 1, 2, \cdots n\} \tag{4}$$

and adopt standard notation for grid function values, $u_{i,j}$

$$u_{i,j} \equiv u(x_i, y_j) \tag{5}$$

- Important note: Here and below will generally ignore treatment of boundary conditions—in general need to be careful with their treatment when using MG, particularly for case of non-Dirichlet conditions

Discretization of model problem

- Replace the continuum system (1) with a discrete version

$$L^h u^h = f^h \tag{6}$$

- Here u^h is the discrete solution, individual values denoted $u_{i,j}$, L^h is the discrete approximation of the differential operator, $L \equiv \partial_{xx} + \partial_{yy}$, and f^h is the discrete source function

- Need finite difference approximations for second derivatives u_{xx} and u_{yy}

- Use standard second-order, centred approximations:

$$u_{xx} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2) \tag{7}$$

$$u_{yy} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2) \tag{8}$$

Discretization of model problem

- Get desired discretization of the Poisson equation:

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = f_{i,j} \qquad 2 \leq i,j \leq n-1 \tag{9}$$

- This equation may be applied at all interior points
- Dirichlet boundary conditions provide (trivial) equations for boundary values on discrete domain:

$$u_{1,j} = u_{n,j} = u_{i,1} = u_{i,n} \qquad 1 \leq i,j \leq n \tag{10}$$

- Discretization results in a large $(N \times N)$, sparse linear system of equations:

$$\mathbf{Lu} = \mathbf{f} \tag{11}$$

Relaxation

- Key idea for relaxation techniques intuitive
- Associate a single equation, corresponding single unknown, $u_{i,j}$, with each mesh point in Ω^h
- Then repeatedly "sweep" through mesh, visiting each mesh point in some prescribed order
- Each time point is visited, adjust value of unknown at grid point so corresponding equation is ("instantaneously") satisfied
- Adopt a "residual based" approach to locally satisfying the discrete equations

Relaxation

- Consider general form of discretized BVP

$$L^h u^h = f^h \tag{12}$$

  and recast in canonical form

$$F^h\left[u^h\right] = 0\,. \tag{13}$$

- Quantity u^h which appears above is the *exact* solution of the difference equations
- Can generally only compute u^h in the limit of infinite iteration
- Thus introduce $\tilde{u}^h$: "current" or "working" approximation to u^h, labelling the iteration number by n, and assuming iterative technique *does* converges, have

$$\lim_{n\to\infty} \tilde{u}^h = u^h \tag{14}$$

Relaxation

- Associated with $\tilde{u}^h$ is residual, $\tilde{r}^h$

$$\tilde{r}^h \equiv L^h \tilde{u}^h - f^h \tag{15}$$

or in terms of canonical form (13),

$$\tilde{r}^h \equiv F^h \left[\tilde{u}^h\right] \tag{16}$$

- For specific *component* (grid value) of residual, $\tilde{r}^h_{i,j}$, drop the h superscript

$$\tilde{r}_{i,j} = \left[L^h \tilde{u}^h - f^h\right]_{i,j} \equiv \left[F^h\left[\tilde{u}^h\right]\right]_{i,j} \tag{17}$$

- For model problem have

$$\tilde{r}_{i,j} = h^{-2}\left(\tilde{u}_{i+1,j} + \tilde{u}_{i-1,j} + \tilde{u}_{i,j+1} + \tilde{u}_{i,j-1} - 4\tilde{u}_{i,j}\right) - f_{i,j} \tag{18}$$

- Relaxation: adjust $\tilde{u}_{i,j}$ so corresponding residual is "instantaneously" zeroed

Gauss-Seidel relaxation

- Gauss-Seidel relaxation: assuming lexicographic ordering of unknowns, $i = 1, 2, \cdots n$, $j = 1, 2, \cdots n$, i index varies most rapidly, residual is

$$\tilde{r}_{i,j} = h^{-2} \left(\tilde{u}^{(n)}_{i+1,j} + \tilde{u}^{(n+1)}_{i-1,j} + \tilde{u}^{(n)}_{i,j+1} + \tilde{u}^{(n+1)}_{i,j-1} - 4\tilde{u}^{(n)}_{i,j} \right) - f_{i,j} \qquad (19)$$

and corresponding Gauss-Seidel update is

$$\tilde{u}^{(n+1)}_{i,j} := \frac{1}{4} \left(\tilde{u}^{(n)}_{i+1,j} + \tilde{u}^{(n+1)}_{i-1,j} + \tilde{u}^{(n)}_{i,j+1} + \tilde{u}^{(n+1)}_{i,j-1} - h^2 f_{i,j} \right) \qquad (20)$$

Gauss-Seidel relaxation—convergence

- Solution of discrete system equivalent to driving residual vector $\tilde{\mathbf{r}}$

$$\tilde{\mathbf{r}} := \mathbf{L}^h \tilde{\mathbf{u}} - \mathbf{f} \tag{21}$$

  to 0

- Can write GS iteration in terms of action of (linear) operator ($N \times N$ matrix), $\mathbf{G}$

$$\tilde{\mathbf{r}}^{(n+1)} = \mathbf{G}\,\tilde{\mathbf{r}}^{(n)} = \mathbf{G}^2\,\tilde{\mathbf{r}}^{(n-1)} = \mathbf{G}^3\,\tilde{\mathbf{r}}^{(n-2)} = \cdots = \mathbf{G}^{n+1}\,\tilde{\mathbf{r}}^{(0)} \tag{22}$$

- Convergence can then be discussed in terms of spectrum of $\mathbf{G}$, in particular will want $\mathbf{G}$ to be a contraction map, so will want spectral radius of $\mathbf{G}$, $\rho(\mathbf{G})$, to satisfy

$$\rho(\mathbf{G}) < 1 \tag{23}$$

Gauss-Seidel relaxation—convergence

- Heuristically at least, can think of eigenvectors of $\mathbf{G}$ as having associated *frequency* or, equivalently, *wavelength* as defined with respect to the mesh, Ω^h

- Rate at which given frequency component of the residual $\tilde{\mathbf{r}}^{(n)}$ is reduced by the iteration is dependent on magnitude of corresponding eigenvalue

- *Mode analysis* (identical in spirit and implementation to Von Neumann analysis for FD approximations to time-dependent PDEs) shows that, asymptotically, convergence rate of GS iteration is dominated by *slow* convergence of lowest frequency (longest wavelength) components, leading to

$$\rho(\mathbf{G}) = 1 - O(h^2) \tag{24}$$

  so that it takes $O(n^2)$ sweeps (n is number of grid-pts per edge of Ω^h) to reduce the residual/solution error by any given constant factor

- Thus need $O(N^2)$ computational work to solve model problem

Illustration of action of GS iteration for model problem

- For illustrative purposes, *specify* continuum solution of model problem

$$u(x,y) \equiv \sin(\pi l_x)\sin(\pi l_y) \tag{25}$$

where $l_x, l_y \geq 1$ are integers, then *compute* corresponding source function

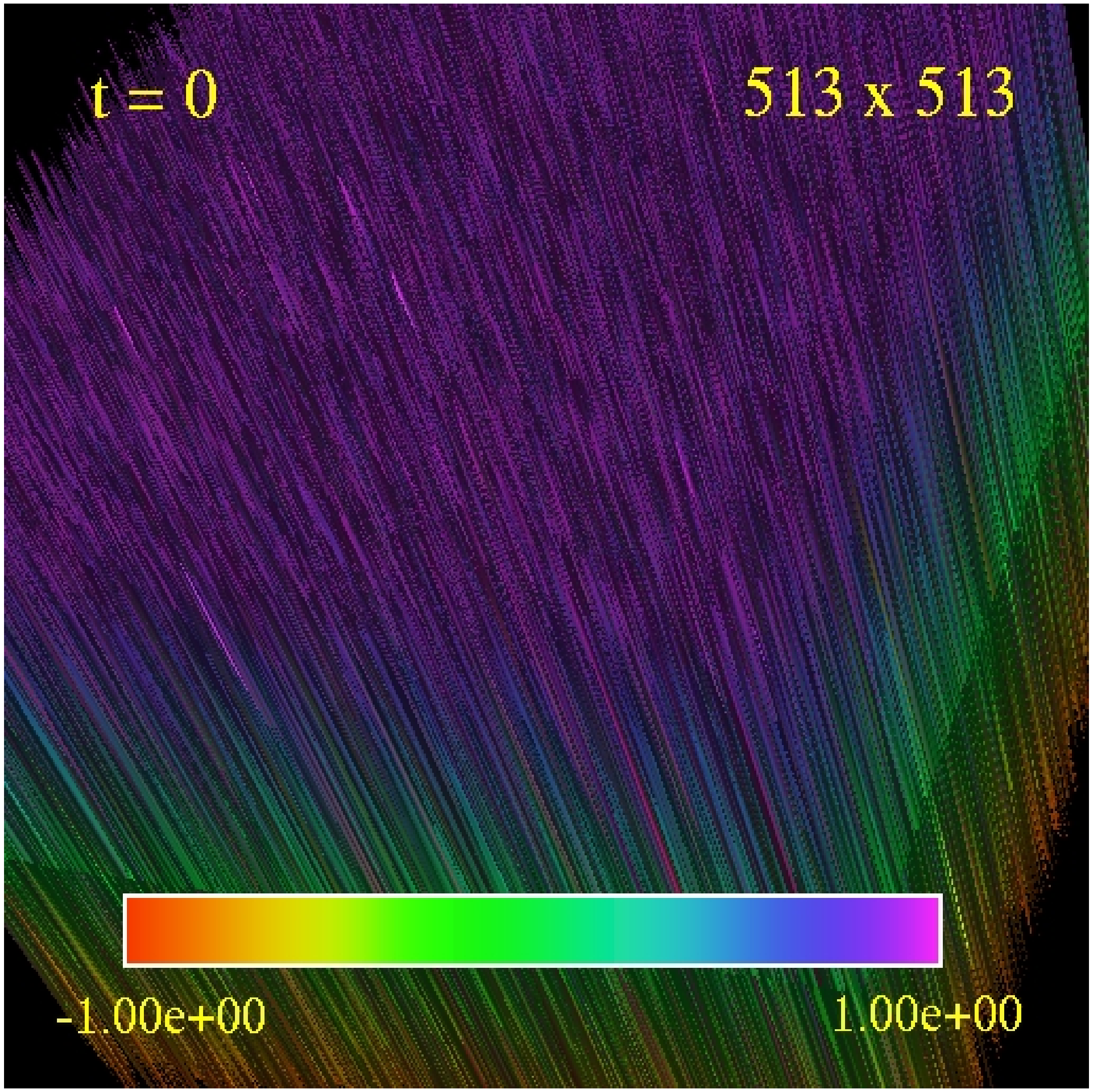
$$f(x,y) = -\pi^2\left(l_x^2 + l_y^2\right)\sin(\pi l_x)\sin(\pi l_y) \tag{26}$$

- Initialize solution to *random* values, uniformly distributed on $[-1,1]$, not least since this will generate initial error/residual vectors with significant components of *all* possible wavelengths; take $l_x = 1$ and $l_y = 2$

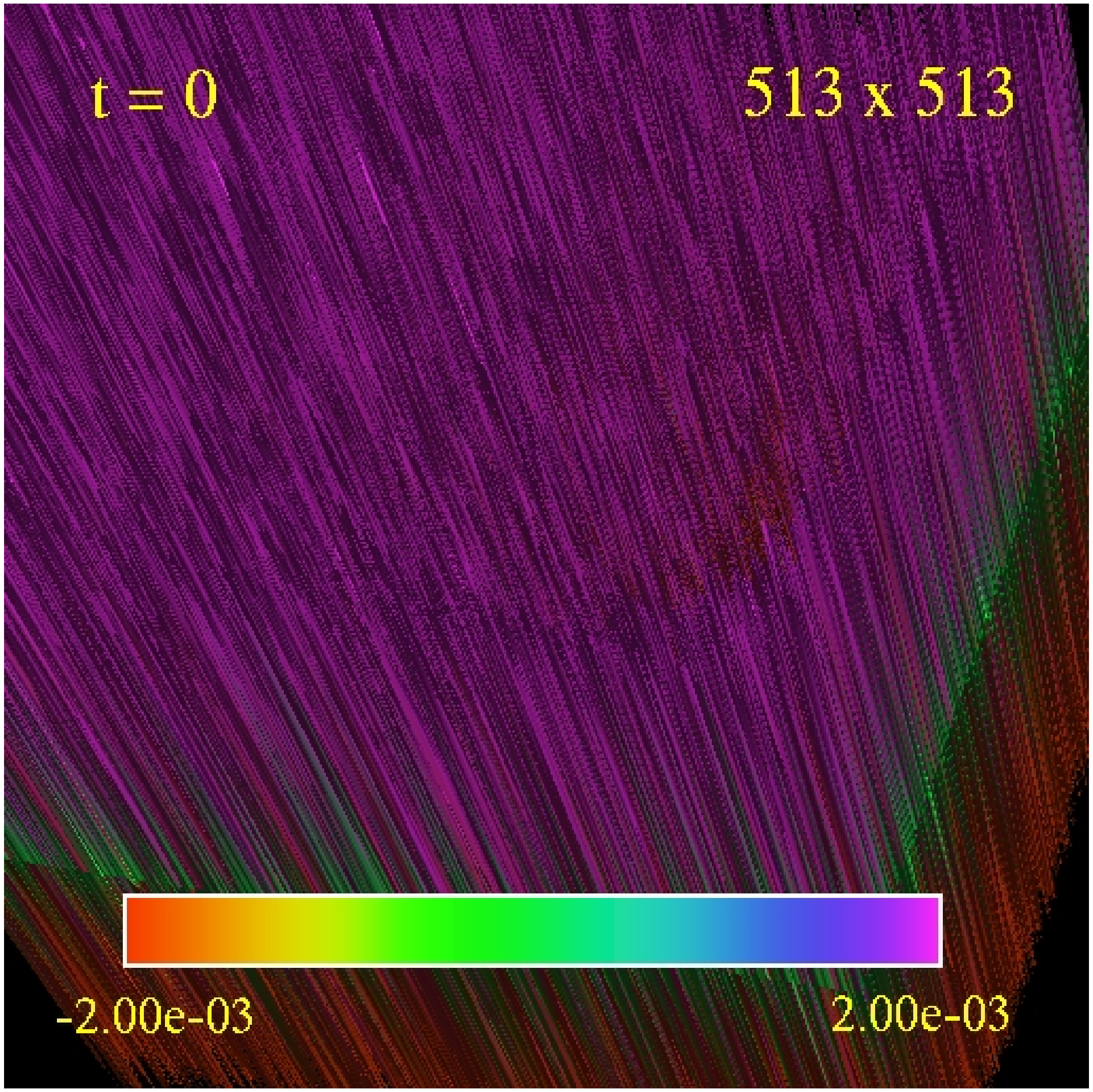
- Following animations show action of GS iteration on solution, solution error and residual, for relaxation sweep numbers

$$n = 1, 2, \ldots 127, 128, 256, 384, \ldots 12800, 14080, 16440, \ldots 128000 \tag{27}$$

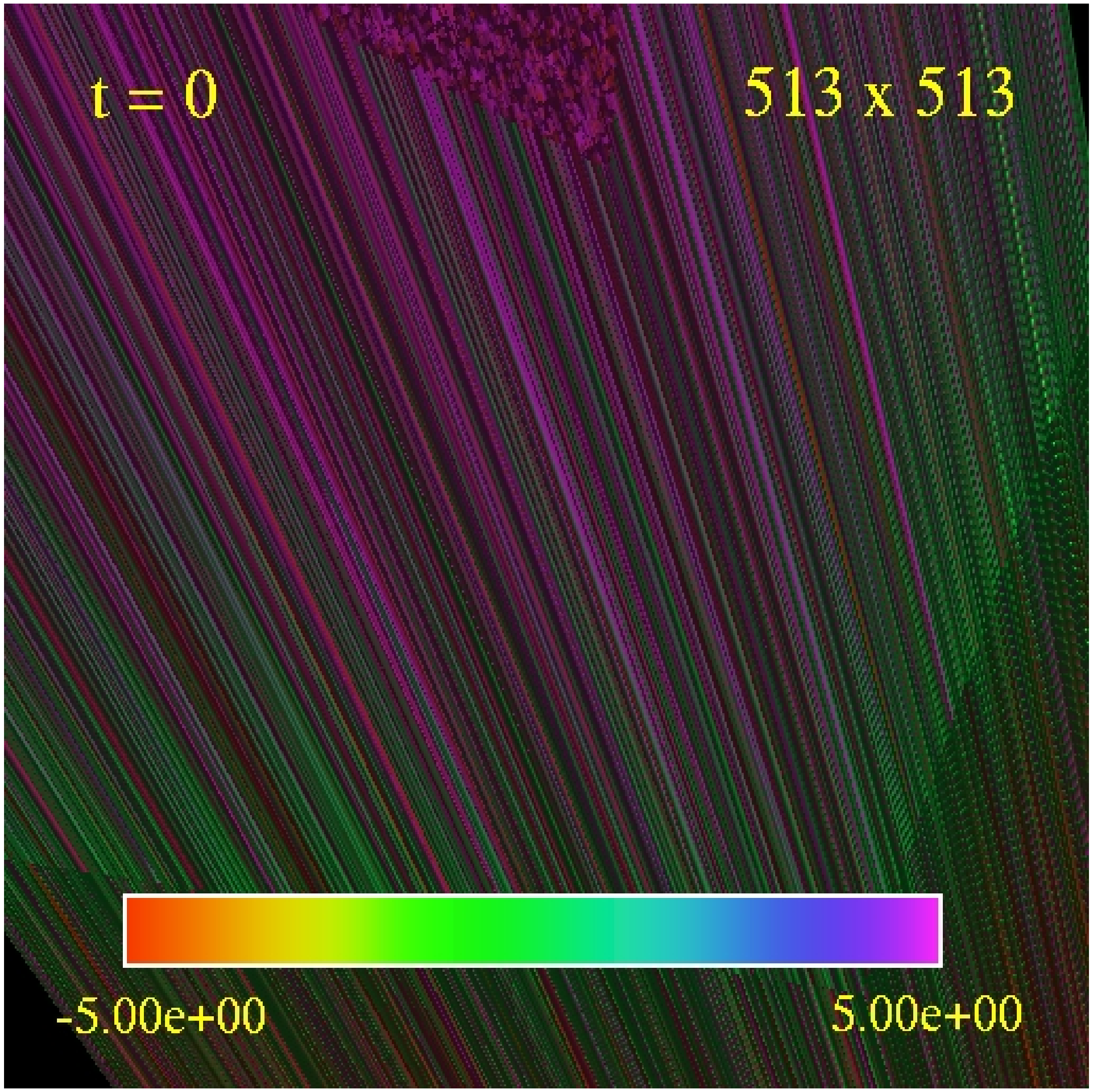
Effect of GS iteration on solution



Effect of GS iteration on solution error

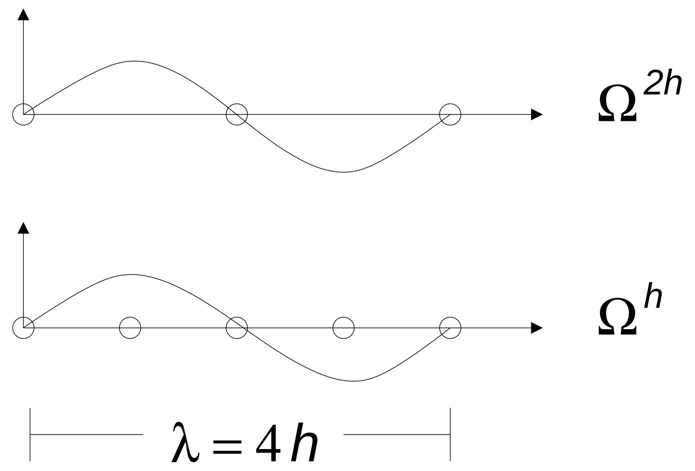


Effect of GS iteration on residual



Convergence of GS iteration—summary

- GS is an abysmal way of *solving* the discrete model problem (and discretized elliptic systems in general), but a very good way of *smoothing* the system (i.e. of reducing high frequency components in the solution error and residual)



- In particular, GS (and other relaxation schemes) very effective for reducing error/residual components on Ω^h that cannot be represented on a 2:1 coarser mesh, Ω^{2h}, i.e. that are above the Nyquist limit on Ω^{2h}, i.e. with wavelengths, $\lambda < 4h$; generally takes some constant (i.e. h-independent) number of sweeps to reduce magnitude of high-frequency components by given factor

Multigrid

- Key ideas

  1. Use relaxation to *smooth* residuals/error on Ω^h
  2. As soon as required correction to solution is smooth, can compute a good estimate for it via a coarse-grid problem, e.g. a problem on Ω^{2h}
  3. Once coarse problem is satisfactorily solved, use the coarse solution to update fine-grid unknown appropriately
  4. Apply 1. to 3. recursively: use problem on Ω^{4h} to accelerate solution of problem on Ω^{2h}, Ω^{8h} problem to accelerate Ω^{4h} solution etc.

- Multigrid in a nutshell

  - Use multi-scale (hierarchical) relaxation to efficiently smooth solution error/residual on all frequency/wavelength scales

- To accomplish this, also need proper operators to transfer problems and solutions from fine to coarse grids and vice versa; will not discuss these in any detail here

Multigrid

- Use hierarchy of meshes $\Omega^h, \Omega^{2h}, \Omega^{4h}, \Omega^{8h}, \ldots$ (generally use 2:1 refinement ratio for efficiency, algorithmic simplicity); label each distinct mesh spacing with integer ℓ
$$\ell = 1, 2, \cdots \ell_{\max} \tag{28}$$
where $\ell = 1$ and $\ell = \ell_{\max}$ label coarsest and finest mesh spacings respectively

- Thus have
$$h_{\ell+1} = \frac{1}{2} h_\ell \qquad\qquad n_{\ell+1} \sim 2^d\, n_\ell \tag{29}$$

- Use ℓ itself to denote resolution associated with a grid function, e.g. define u^ℓ via
$$u^\ell \equiv u^{h_\ell} \tag{30}$$

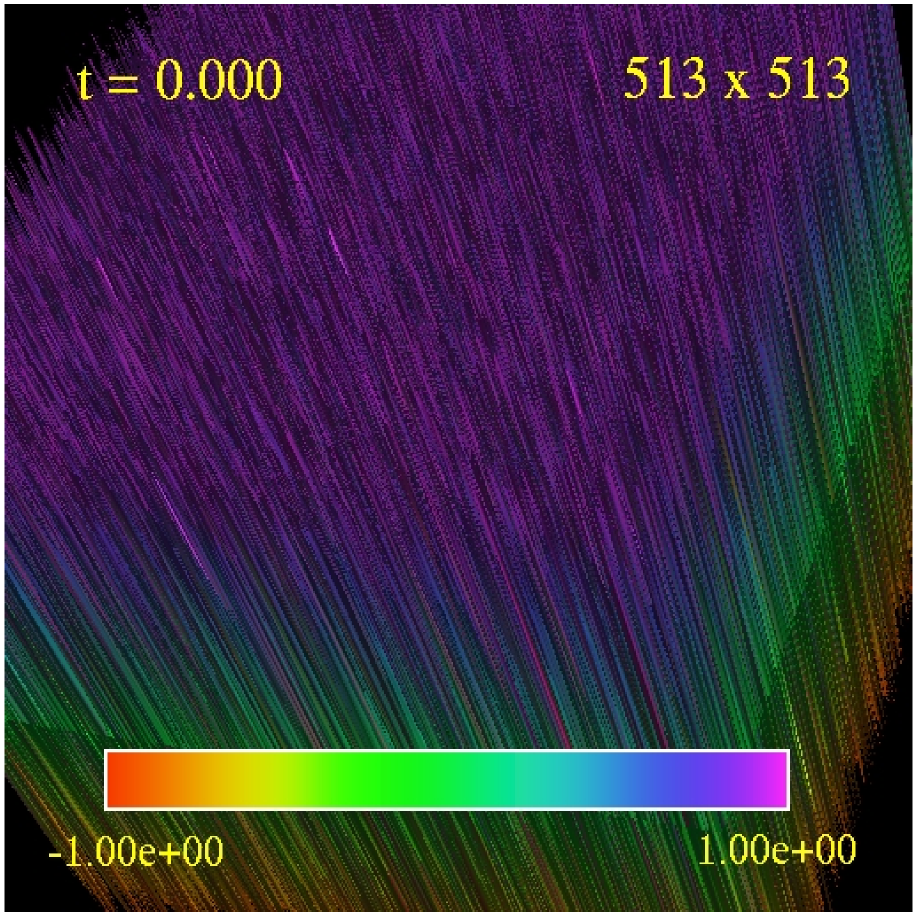
- Note: General multigrid iteration involves solution of problems
$$L^\ell u^\ell = s^\ell \tag{31}$$
where, apart from the finest grid problem, the source function, s^ℓ, will *not* coincide with the "right hand side of the PDE", f^ℓ

Pseudo-code of typical multigrid iteration (V-cycle)

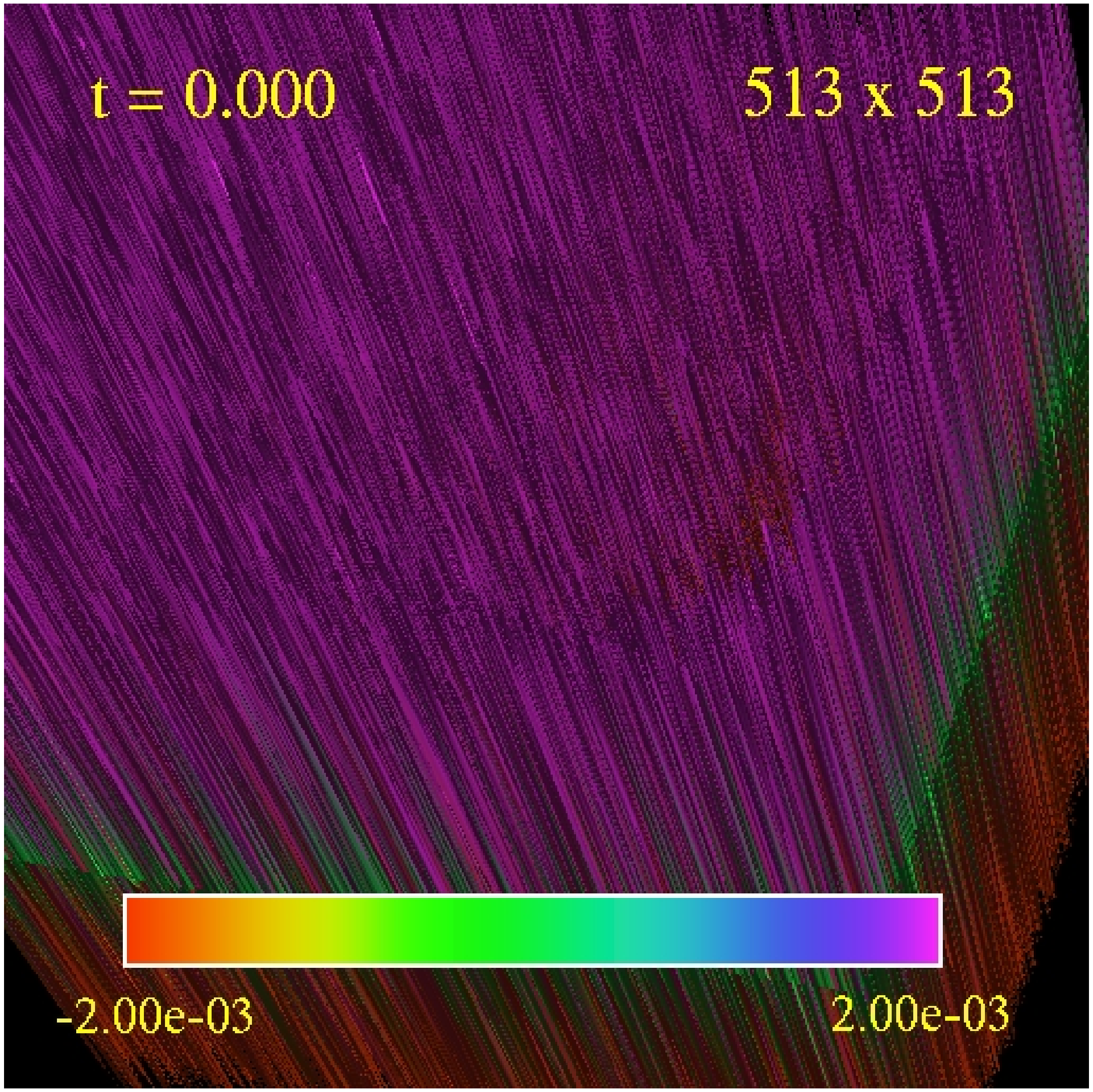
```
procedure vcycle ( $\ell$, $p$, $q$ )
   Cycle from fine to coarse levels
   do $m = \ell , 2 , -1$
      Apply pre-coarse-grid-correction (CGC) smoothing sweeps
      do $p$ times $u^m$ := relax ( $u^m$, $s^m$, $h^m$ ) end do
      Set up coarse grid problem
      $[u^{m-1}, s^{m-1}]$ := setup_coarse ( $u^m$, $s^m$, $h^m$ )
   end do
   Solve coarsest-level problem
   $u^1$ := solve_coarse ( u$^1$, s$^1$, h$^1$ )
   Cycle from coarse to fine levels
   do $m = 2 , \ell , +1$
      Apply coarse-grid correction
      $u^m$ := update_fine ( $u^m$, $u^{m-1}$ )
      Apply post-CGC smoothing sweeps
      do $q$ times $u^m$ := relax ( $u^m$, $s^m$, $h^m$ ) end do
   end do
end procedure
```

Effect of MG iteration on solution

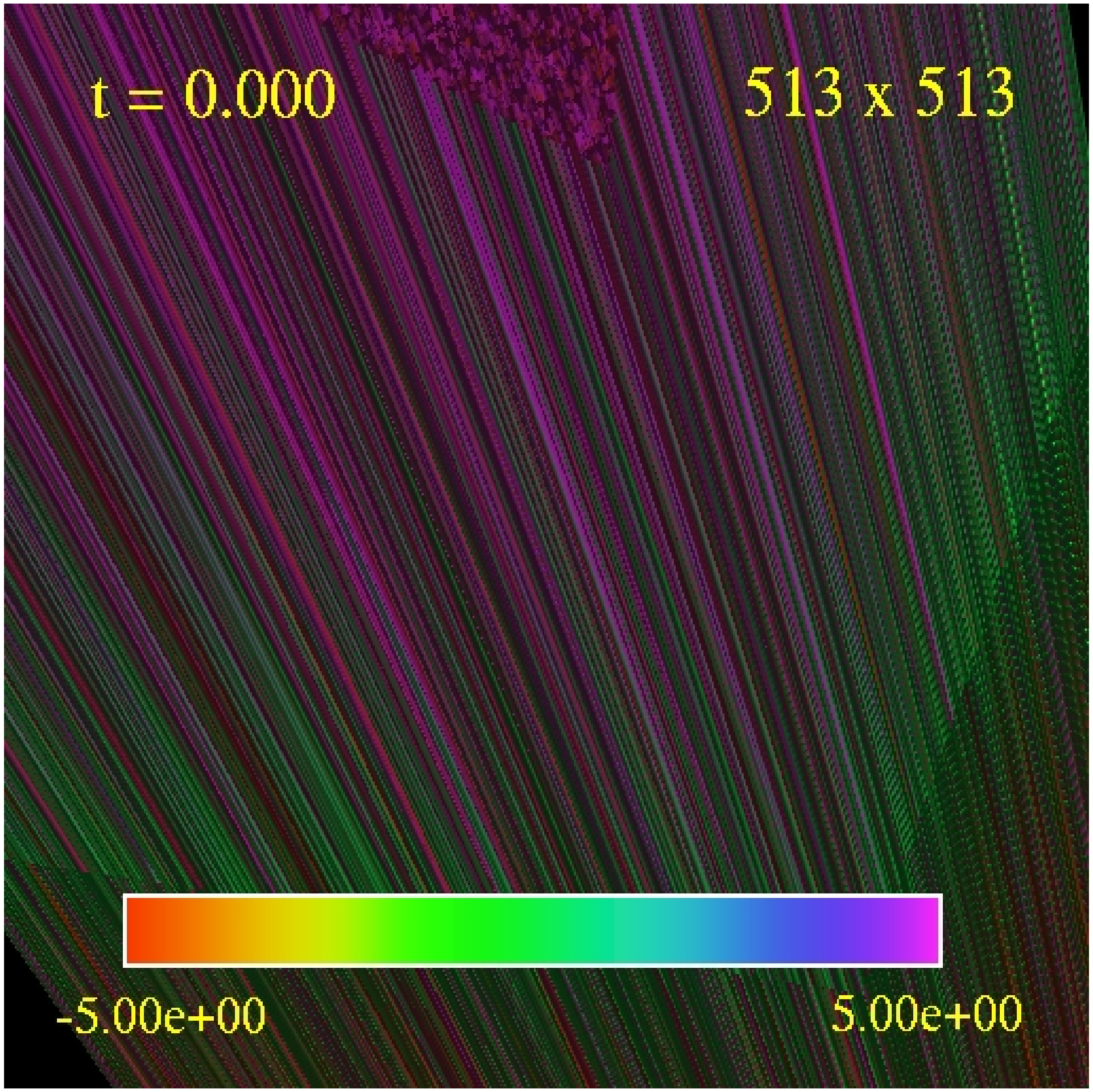
- Apply 5 V-cycles ($p = 1$, $q = 2$), using same random initial conditions as previously
- t label measures relaxation work in units of fine-grid relaxation sweep (dominant cost for MG algorithm)



Effect of MG iteration on solution error



Effect of MG iteration on residual



Multigrid for time-dependent problems (at last!)

- Again, illustrate general technique using simple model problem: 2D diffusion equation (heat equation) with homogeneous, Dirichlet boundary conditions

$$u_t\,(t,x,y) = \nabla^2 u = u_{xx} + u_{yy} \tag{32}$$

on

$$\Omega : 0 \le x \le 1\,,\ 0 \le y \le 1\,,\ t \ge 0 \tag{33}$$

with initial conditions

$$u(0,x,y) = u_0(x,y) \tag{34}$$

(u_0 specified) and boundary conditions

$$u(t,0,y) = u(t,1,y) = u(t,x,0) = u(t,x,1) = 0 \tag{35}$$

Multigrid for diffusion equation

- Use fully-implicit $O(h^2)$ Crank-Nicholson approximation on uniform grid with $\Delta x = \Delta y = h$, $\Delta t = \lambda h$ (in abuse of terminology, will refer to λ as "Courant number")

$$\frac{u_{i,j}^{n+1} - u_{i,j}^{n}}{\Delta t} = \frac{1}{2}\left(\Delta^h u_{i,j}^{n+1} + \Delta^h u_{i,j}^{n}\right) \tag{36}$$

  where

$$\Delta^h u_{i,j} = h^{-2}\left(u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}\right) \tag{37}$$

- Identify $u_{i,j}^h \equiv u_{i,j}^{n+1}$, then (36) is of the form

$$L^h u^h = f^h \tag{38}$$

  with

$$L^h \equiv \left[\Delta t^{-1} - \frac{1}{2}\Delta^h\right] \qquad\qquad f^h \equiv \left[\Delta t^{-1} + \frac{1}{2}\Delta^h\right] u_{i,j}^n \tag{39}$$

- Now use multigrid to solve (38) at every time step

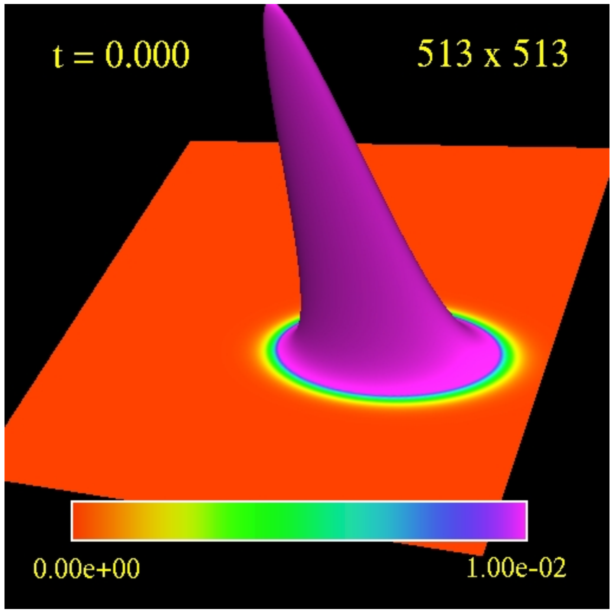
MG solution of diffusion equation

- Initial data given by

$$u_0(x,y) = \exp\left(-\left(\left(x-0.6\right)/0.05\right)^2 - \left(\left(y-0.7\right)/0.10\right)^2\right) \tag{40}$$

- $\lambda = 0.0125$: relatively small value chosen for purposes of animation

- Can use $\lambda = 1.0$ or larger, but for such large values, accuracy of calculation suffers considerably



Computational cost

- Compare with another technique that can be used to compute $O(h^2)$ implicit approximate solution of diffusion equation in $O(N)$ time: Alternating Direction Implicit Method (ADI)

- From $u_t = Lu = (\partial_{xx} + \partial_{yy})u$ have

$$u^{n+1} = \exp\left(\Delta t L\right)\, u^n \tag{41}$$

or

$$\exp\left(-\frac{\Delta t}{2}L\right)\, u^{n+1} = \exp\left(\frac{\Delta t}{2}L\right)\, u^n \tag{42}$$

- Expanding to $O(\Delta t^2) = O(h^2)$ accuracy, and denoting the usual $O(h^2)$ approximation of L by L^h

$$\left(1 - \frac{\Delta t}{2}L^h\right) u^{n+1} = \left(1 + \frac{\Delta t}{2}L^h\right) u^n \tag{43}$$

Computational cost

- Straightforward to show that last expression can be "factored" as

$$\left(1 - \frac{\Delta t}{2}\partial^h_{xx}\right)\left(1 - \frac{\Delta t}{2}\partial^h_{yy}\right)u^{n+1} = \left(1 + \frac{\Delta t}{2}\partial^h_{yy}\right)\left(1 + \frac{\Delta t}{2}\partial^h_{xx}\right)u^{n} + O(h^3) \tag{44}$$

  where ∂^h_{xx} and ∂^h_{yy} are the usual $O(h^2)$ approximations of ∂_{xx} and ∂_{yy}

- Can then solve (44) using alternating sweeps in x and y directions. Each sweep requires the solution of n tridiagonal systems in n unknowns.

- Total cost is $O(n^2) = O(N)$

Scaling of computational cost for model problem

- Numerical experiments used $n_x - 1 = n_y - 1 = n - 1 = 64, 128, 256, 512, 1024$, corresponding to discretization levels, $\ell = 1, 2, 3, 4$ and 5, with a number of time steps, $n_t^\ell = 2^{\ell-1} n_t^1$
- Measured rate, R, of computation is $\kappa T_{\mathrm{CPU}}/(n_t n_x n_y)$ where κ is a normalizing constant
- R should be constant for $O(N)$ scaling

n	R_{ADI}	R_{MG}
64	1.00	1.42
128	1.01	1.44
256	1.09	1.74
512	1.28	1.90
1024	1.15	2.10

- MG slowdown for larger N probably due to caching effects

Summary & comments

- Multigrid methods can be used to solve time-dependent finite difference equations in $O(N)$ time ($N =$ number of points in spatial discretization)

- Most useful for PDEs that have "stiffness", and thus generally require implicit treatment to avoid need for unnecessarily small time steps (stability), bad scaling of computational cost as $h \to 0$

- Have illustrated technique for simple model problem: even in this case performance of MG compares favorably to ADI

- However, in contrast to ADI and most other methods, MG readily generalizes to
  - Evolution equations involving general elliptic operators on the RHSs (what we encounter in general relativity, and other sets of geometric PDEs, e.g. Ricci flow)
  - Nonlinear equations
  - Systems of equations

  and $O(N)$ performance can also be expected in these cases