Multigrid for Poisson's Equation

The task is to solve the Poisson equation on a square *quickly*. The unit square is discretized by an n x n grid, so the cells have $\Delta x = \Delta y = 1/(n\text{-}1)$, and we propose to solve the discretized equations iteratively by explicit time-stepping:

$$\frac{\partial V}{\partial t} = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + f(x,y), (x,y) \in \Omega$$

$V_{1j}, V_{nj}, V_{i1}, V_{in}$ given by Dirichlet BC, never changed

$\alpha = \Delta t / \Delta x^2$

for $k = 0,1,...$

  for $i,j = 2,...,n-1$

$$V_{ij}^{(k+1)} = V_{ij}^{(k)} + \alpha \cdot \left(V_{i,j+1}^{(k)} + V_{i,j-1}^{(k)} + V_{i+1,j}^{(k)} + V_{i-1,j}^{(k)} - 4V_{i,j}^{(k)} + \Delta x^2 \cdot f(x_i, y_j) \right)$$

  end

end

with initial values

$$V_{ij}^{(0)} = 0$$

Notes

1. $\alpha < 1/4$ is necessary for stability (see below)
2. The central difference approximation to the Laplace operator is second order accurate.

How many steps are necessary to obtain an accurate solution?

The answer is produced most expeditiously by Fourier analysis. In order to illustrate with simpler formulas we consider the 1D problem with solution $u = 0$,

$$u_t = u_{xx}, u(0) = u(1) = 0, \Delta x = 1/(n-1),$$

$$u(x,0) = f(x) \Rightarrow u_j^0 = f(j\Delta x)$$

for $k = 0,1,...$

$$u_j^{k+1} = u_j^k + \alpha \cdot \left(u_{j-1}^k - 2u_j^k + u_{j+1}^k \right), j = 2,...,n-1$$

end

where the initial guess is $u = f(x)$. By separation of variables, the solution u_j can be written as discrete sine series (this takes care of the boundary conditions)

$$u_j^k = \sum_{m=1}^{n-1} c_m^k \sin \frac{mj\pi}{n}; \text{ Note that } \sin(x+d) + \sin(x-d) = 2\sin x \cos d$$

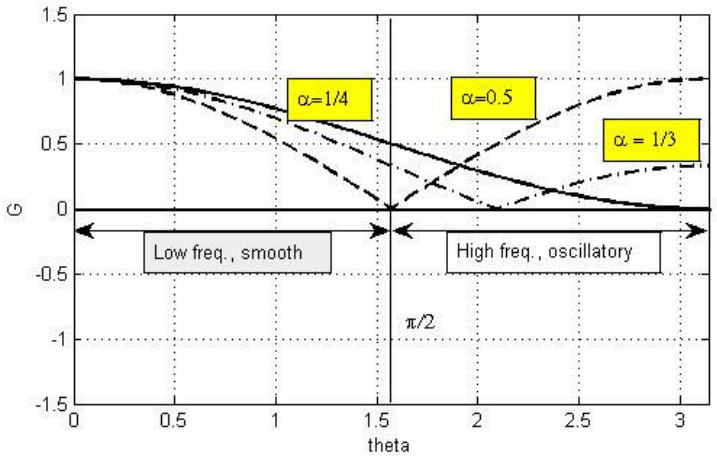
the Discrete Fourier Transform. Let us see what happens to one of the harmonics, say

$$u_j = c_m \sin(j\theta), \theta = m\pi / n;$$

$$u_{j-1} - 2u_j + u_{j+1} = ... = -4\sin^2(\theta/2) \cdot u_j$$

so

$$c_m^{k+1} = G(\alpha,\theta)c_m^k, G = (1-4\alpha\sin^2(\theta/2)), \theta = m\pi\Delta x$$



which defines the "von Neumann amplification factor" G for the wave-number m, with phase-shift per cell θ. The smallest θ is π/n, and the largest is $\pi(n-1)/n$. So, we have ($\sin x \approx x$ for small x)

$$G(\pi/n) \approx 1-\alpha(\pi/n)^2, G(\pi) \approx 1-4\alpha$$

It is clear that $\alpha < 1/2$ is necessary for the iteration to decrease *all* the harmonics. Then

$$\max|G| = \max(4\alpha-1, 1-\alpha(\pi/n)^2)$$

• What is the "optimal" choice, i.e. the α which minimizes max|G| ?
For the 2D case the condition is $\alpha < ¼$.
• Exercise: Prove!

The number of steps N to reduce all harmonics (initially assumed to have amplitude $O(1)$) to an amplitude d is given by

$$G^N < d \text{ or } N > \frac{\ln d}{\ln(1-\frac{\pi^2}{2n^2})} \approx 2n^2\frac{-\ln d}{\pi^2}$$

so the iteration is *extremely* slow for large n. The graph of G vs. θ shows that the

smooth components, say with $\theta < \pi/2$,

are responsible for the slow convergence, whereas the

oscillatory components with $\theta > \pi/2$,

are quickly damped. For α = 1/4, G is smaller than 1/2, and α = 1/3 is optimal in the sense of giving

$$\min_{\alpha} \max_{\pi/2<\theta<\pi} |G(\alpha,\theta)|$$

Exercise:
Prove! and compute what the min is.
This is where the multi-grid idea enters: The time-stepping *is* efficient at reducing the short harmonics ($\theta > \pi/2$) if $\alpha < 1/2$. But the long harmonics can be well represented on a coarser grid on which they will appear more oscillatory. Indeed, half of them will be oscillatory according to the definition.
So the following is a natural attempt at a two-grid iteration to solve **Au** + **f** = 0 where **A** corresponds to an elliptic operator like the Laplace operator, with all negative eigenvalues. Indeed, we have

$\mathbf{v}^T\mathbf{A}\mathbf{v} \le -\mu\mathbf{v}^T\mathbf{v}$
and $\mu = \pi^2/n^2$ in the above 1D example.

0. $\mathbf{v} = 0$ - take a better initial guess if it exists
1. Run $N1$ timesteps (= iterations) $\mathbf{v} = \mathbf{v} + \alpha(\mathbf{A}\mathbf{v} + \mathbf{f})$. This reduces the oscillatory components of the residual $\mathbf{r} = \mathbf{A}\mathbf{v}+\mathbf{f}$:
even if the solution itself u is oscillatory due to oscillatory f !!
2. Solve the correction equation $\mathbf{A}\mathbf{e} = \mathbf{r}$ on a coarser grid:
restrict $\mathbf{r}$ onto $\mathbf{r}_C$;
(*) solve $\mathbf{A}_C\mathbf{e}_C = \mathbf{r}_C$;
3. Interpolate (prolong) the coarse correction $\mathbf{e}_C$ onto the fine grid as $\mathbf{e}$;
correct $\mathbf{v}$: $\mathbf{v} = \mathbf{v} + \mathbf{e}$;
If not converged, goto 1.

When the exact solve step (*) is replaced by an approximate iterative solution by recursive use of even coarser grids the multi-grid V-cycle algorithm appears (Briggs p. 48) on a hierarchy of grids which we label 0 for finest and *lmax* for coarsest. The grids here will be obtained by successively doubling the mesh-width, and the number of gridpoints on the finest level ($2^M + 1$) is such that all gridpoints on level m appear also on all finer levels.

The intergrid transfer operators are defined next (see Briggs). The prolongation from stepsize ($h = \Delta x$) $2h$ to h will be done by linear interpolation, and the restriction can be done by simple injection, or by the Galerkin construction. The grid-levels are denoted by superscripts and component numbers are subscripts,

$$\text{Prolong}, I_{2h}^{h} : \begin{cases} v_{2j}^{h} := v_j^{2h} \\ v_{2j+1}^{h} := \frac{1}{2}(v_j^{2h} + v_{j+1}^{2h}) \end{cases}, \text{Restrict}, I_h^{2h} : v_j^{2h} := v_{2j}^{h}$$

$$\mathbf{I}_{2h}^{h} = \mathbf{P}_n = 0.5\begin{pmatrix} 2 & 0 & 0 & \dots \\ 1 & 1 & 0 & \dots \\ 0 & 2 & 0 & \dots \\ 0 & 1 & 1 & \dots \\ 0 & 0 & 2 & \dots \\ 0 & 0 & 1 & \dots \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 2 \end{pmatrix}, n \times (n+1)/2$$

$$\mathbf{I}_{h}^{2h} = \mathbf{R}_{(n+1)/2} = \begin{cases} \text{Injection} : \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 1 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & 1 & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}, (n+1)/2 \times n \\ \text{Galerkin} : 0.5\mathbf{P}_n^T \end{cases}$$

The Galerkin restriction is also called “full weighting”.

There remains to define how to construct the matrix $\mathbf{A}_C$ from **A,** generally, how to construct **A** for a given grid level.

1. From discretization on the grid level:
If we set **A** = difference approximation to d^2/dx^2 on the current grid, computation is required to set up $\mathbf{A}_C$ from **A** because Δx changes. If we multiply the equations by Δx^2 - as done above - the *matrices* look the same on all grid levels, but the right-hand side **f** has to be scaled:

$$\mathbf{A} = \begin{pmatrix} -2 & 1 & 0 & ... & 0 \\ 1 & -2 & 1 & ... & 0 \\ 0 & 1 & -2 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & ... & 0 & 1 & -2 \end{pmatrix}, \quad \mathbf{f} = \Delta x^2 \begin{pmatrix} f(x_2) \\ f(x_3) \\ f(x_4) \\ ... \\ f(x_{n-1}) \end{pmatrix}$$

2. The Galerkin recipe for constructing $\mathbf{A}_C$ from **A** may be explained as follows:
Find an approximate solution to **Au** + **f** = 0 in the subspace **u** = **Pv**, e.g. with **u** a linear interpolant to a coarse grid function **v**, such that the *restriction* of the residual vanishes:

$$\mathbf{RAPv} + \mathbf{Rf} = 0$$

With **R** as injection, we simply pick every second of the equations. The Galerkin recipe is symmetric,

$$\mathbf{P}^T\mathbf{APv} + \mathbf{P}^T\mathbf{f} = 0$$

Note:
If **A** is a symmetric positive definite matrix (such as the Laplace matrix) the Galerkin **v** is the best in the sense of minimizing the "energy error" $(\mathbf{v}-\mathbf{u})^T\mathbf{A}(\mathbf{v}-\mathbf{u})$ over the subspace **P**. This property enables error estimates. However, the MG never solves the equation exactly on any grid, so the benefit of the "optimality" in actual computation may be limited.

Exercise: Prove that for the Laplace matrix in 2D, the three variants:

i) Difference approximation on the coarse grid
ii) Restriction as Injection
ii) Restriction as full weighting

give the same matrices $\mathbf{A}_C$ (possibly with different scalings)

The MG algorithm needs a simple interface to the difference equations. It requires *no* coefficient matrices, only - for a candidate solution **v** on a grid hierarchy - :

1. A basic iterative smoother for the residual $\mathbf{v}^{(n+1)} := \text{smooth}(\mathbf{v}^{(n)})$
2. The residual **r** = **Av**+**f** itself

When the basic smoother is a time-stepper,

$$\mathbf{v}^{(n+1)} = S(\mathbf{v}^{(n)}), S(\mathbf{x}) = \mathbf{x} + \alpha\mathbf{r},$$
$$\mathbf{r}(\mathbf{x}) = \mathbf{Ax} + \mathbf{f};$$
$$\because \mathbf{r}(\mathbf{x}) = (S(\mathbf{x}) - \mathbf{x})/\alpha$$

we may choose evaluation of *S* as the only interface. The *relaxation* parameter - is the α we saw above.