Cycling Choices in Algebraic Multigrid

Abigail Gregory
Harvey Mudd College
abby_gregory@hmc.edu

Background

Purpose

Consider a simple system of equations represented by
    $Ax = b$,
where A is an nxn matrix and x and b are nx1 vectors. Knowing A and b, we solve for x by multiplying by the inverse of A. This takes $O(n^3)$ time. This is too slow for huge matrices.
If A is sparse, meaning it contains mostly 0's, several iterative methods can approximate x much more quickly. Algebraic multigrid (AMG) is one such method. PyAMG implements AMG in Python.
I implemented a solver in PyAMG that changes with each iteration rather than running the same way every time. Customizing the iterations improves performance for matrices that ran poorly with PyAMG before.

Relaxation

Algebraic multigrid applies a relaxation method on multiple grids. The relaxation method is derived from the following logic.
It should be the case that
    $Ax = b$,
and therefore that
    $Ax - b = 0$.
When x is an approximation, however,
    $Ax - b = r$.
When x is correct, r is an nx1 vector of 0's. Relaxation methods try to smooth r toward 0.

The following is an example with matrix A, illustrated here with a blue square at each non-zero value and a white square at each zero value in the matrix.
Plotting the values in r throughout the relaxation shows how r converges toward 0.

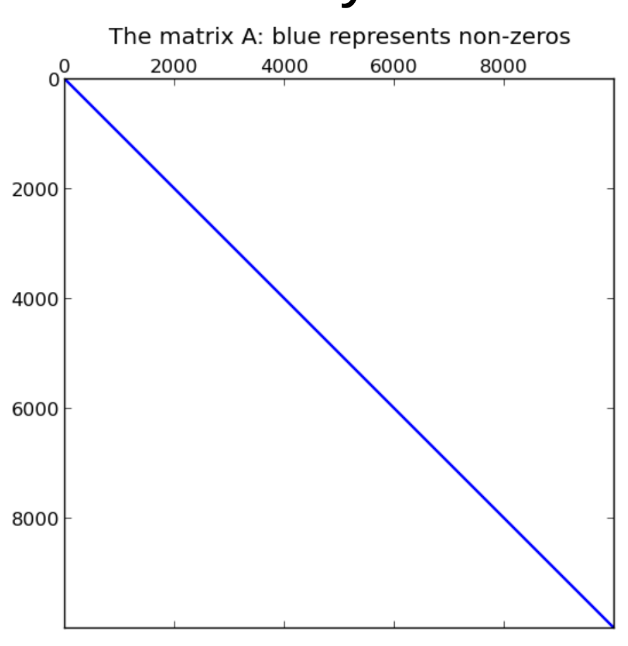

Figure 1

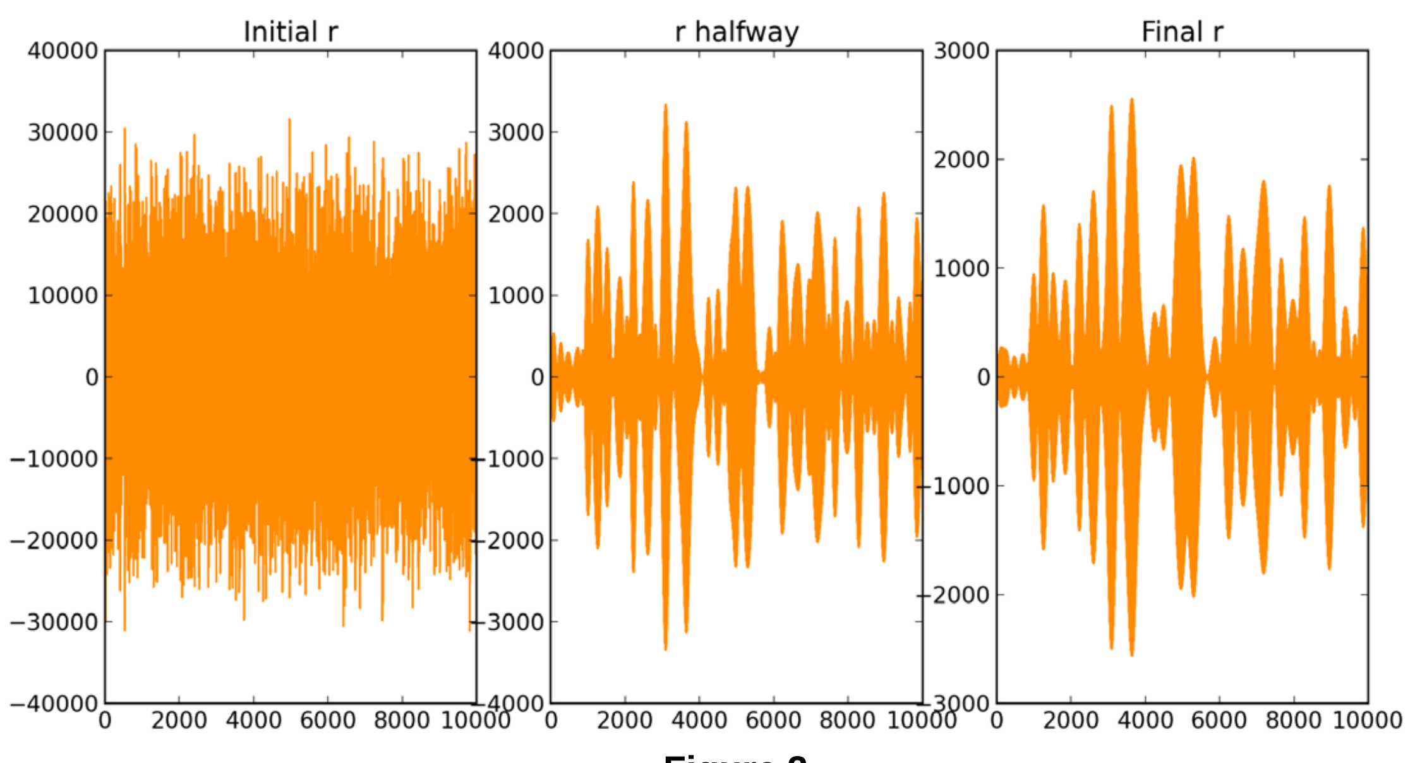

Figure 2

Relaxation is not sufficient by itself because it smoothes the high-frequency, but not the low-frequency, components of r. In the plots above, the high-frequency components of r are smoothed between the first two panels. Little progress is made between the last two panels, since relaxation cannot eliminate the low-frequency components that remain.

Multigrid

To smooth the low-frequency components of r, relaxation is applied on multiple grids. Here is an example of converting between 1D grids.

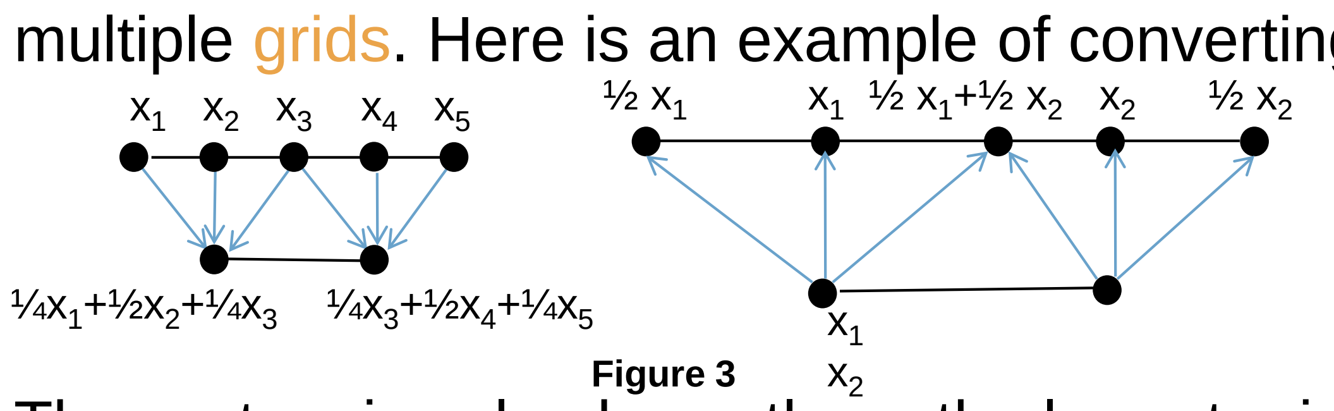
x_1 x_2 x_3 x_4 x_5
$\frac{1}{4}x_1+\frac{1}{2}x_2+\frac{1}{4}x_3$
$\frac{1}{4}x_3+\frac{1}{2}x_4+\frac{1}{4}x_5$
$\frac{1}{2}x_1$ x_1 $\frac{1}{2}x_1+\frac{1}{2}x_2$ x_2 $\frac{1}{2}x_2$
x_1
x_2

Figure 3

The system is solved exactly on the lowest grid, which is small enough to solve on, but the solve is not exact because grid conversion loses some data.
Moving to a different grid alters the resolution of the system. In the lower grid, low-frequencies become higher-frequencies and relaxation can target them. The residual shrinks more with AMG than relaxation, and the low frequency components never dominate, so r does not look sinusoidal.

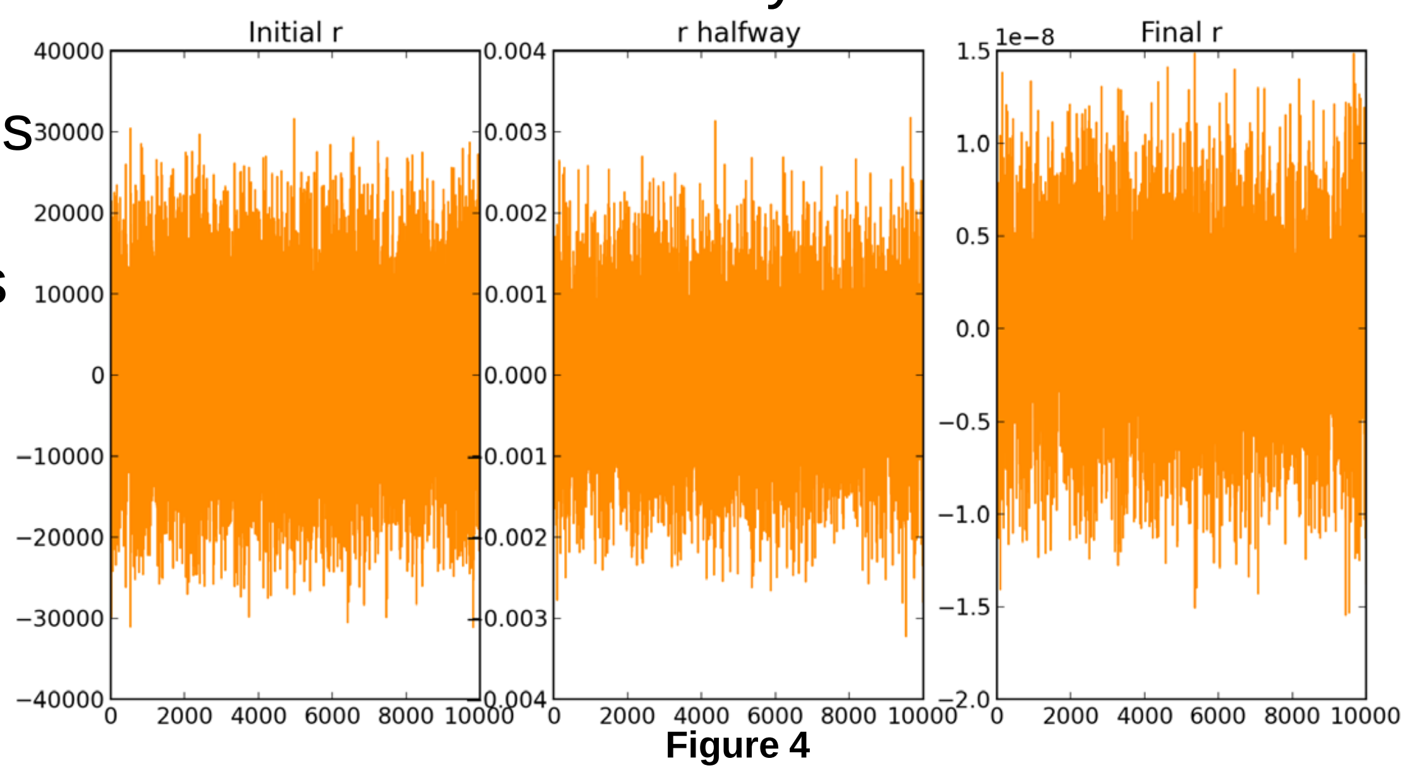

Figure 4

Methods

Modeling cycle time

In order to make better choices about the next cycle, I developed a model that estimates the time the cycle will take.
The dominating costs of AMG are the three matrix-vector multiplications that occur at each grid-level: one to convert to the next grid down, one to convert back up to the current grid, and one for the relaxation. These operations are timed during the first cycle of AMG. The solve time at the lowest grid is included by timing the calculation of A inverse and multiplication of A inverse and b.
Plotted here with matrices of various sizes are the time a cycle took in AMG (purple), my model's estimate (green), and the estimate using a standard model (black).
The model is fairly accurate. Also its data is collected during the first cycle when the significant multiplications are happening anyway, so estimates are cheap.

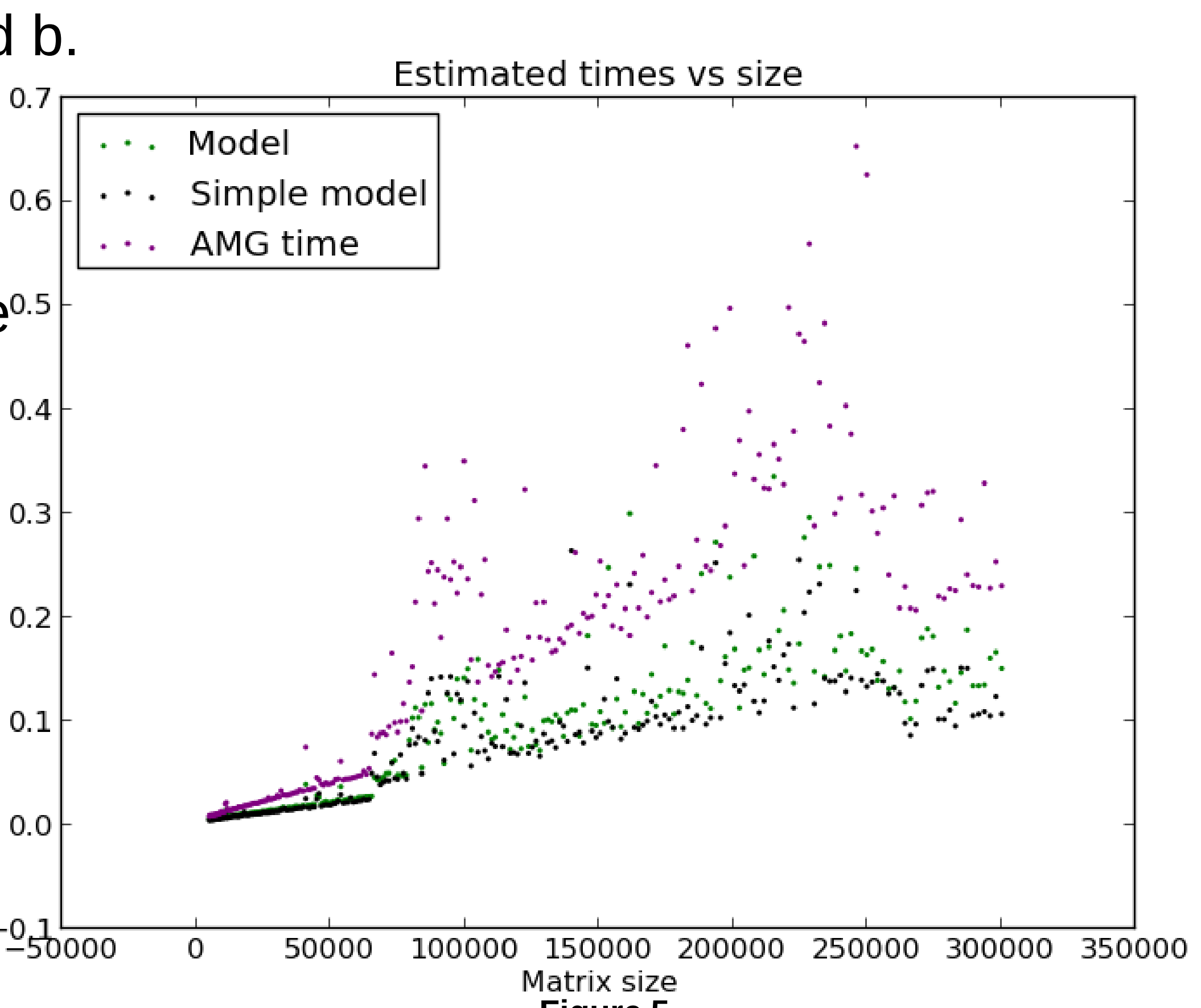

Figure 5

Properties of cycles

Iterations of algebraic multigrid are called cycles. Several properties of each cycle may be adjusted to affect the way in which the norm of r decreases. (Note that the norm of r approaches 0 as the vector r approaches the vector 0.) I considered the number of relaxations and the solve-depth used.

Relaxations

By default, PyAMG performs one iteration of relaxation at each grid-level. Using more iterations can help to bring down the residual more quickly, but more calculations slow down the cycle.
The iterations are incremented if the norm of the residual dropped by a factor less than 1.2 in the last cycle and the model estimates that the cycle will take less than 5 times as long as the default cycle.
This matrix represents a real engineering structural problem. When relaxations are modified, the residual drops much more quickly.

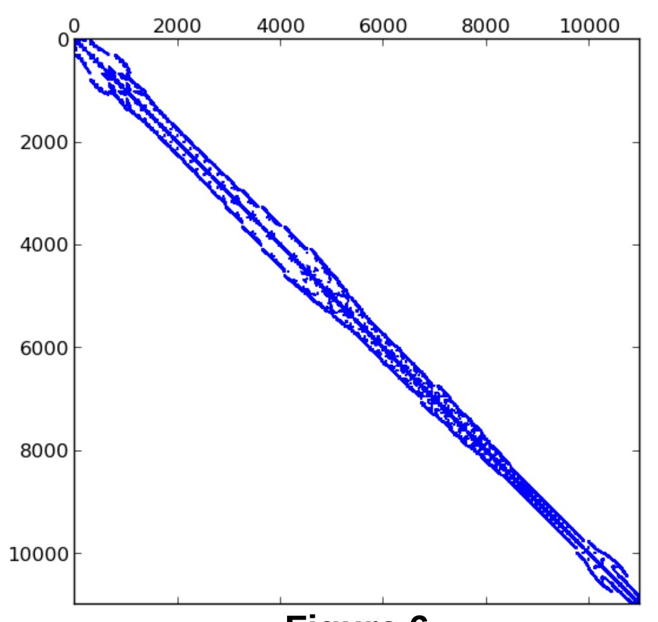

Figure 6

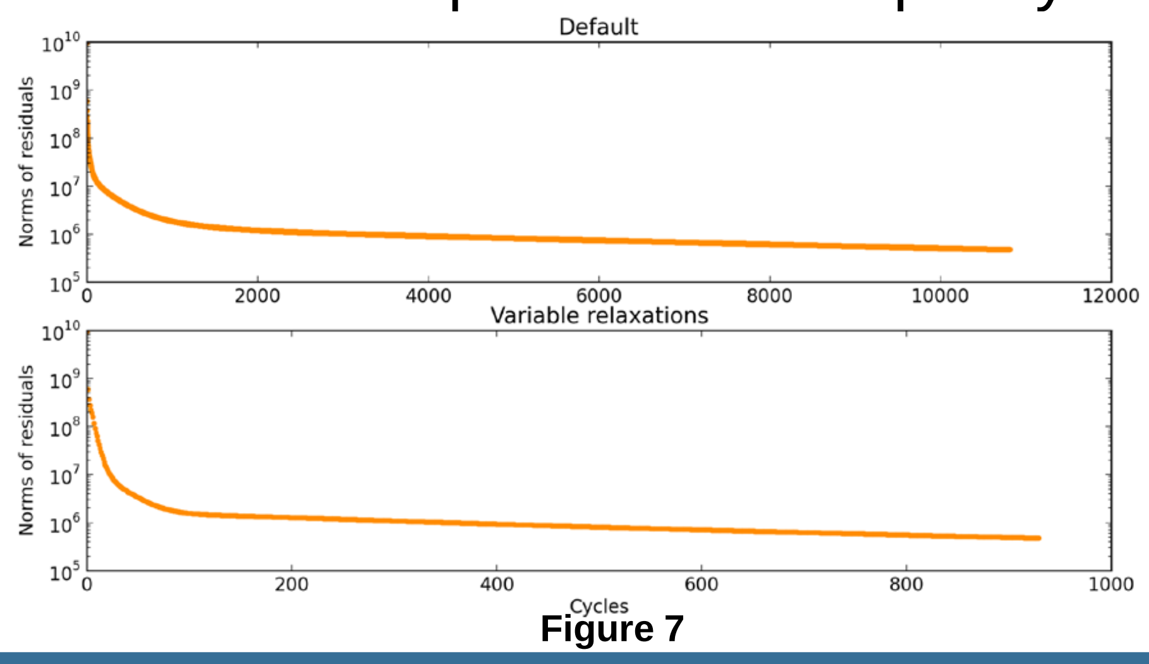

Figure 7

Solve depth

AMG chooses a default number of grids to use for each matrix. Using fewer grids and solving at a higher depth in the cycle produces a more accurate result, as grids introduce inaccuracy, but solving with a larger matrix takes longer. It is difficult to discern whether the convergence will improve enough to be worth the cost. My implementation changes the depth if the slope of the residual plot has been constant for half of the cycles run so far.
This is an area that could be explored in much greater detail in the future, as the current solution only has a marginal effect on the number of cycles and generally increases the overall solve time.

Thanks

Professor Luke Olson has provided guidance throughout this project. Professor Craig Zilles and Jill Peckham were a phenomenal support system and Steven Hussung a phenomenal research partner. I appreciate all of your support!

Passionate on Parallel
2012 REU Program

parallel COMPUTING INSTITUTE

This REU is co-funded by the ASSURE program of the Department of Defense in partnership with the National Science Foundation REU Site program under Award No. 1004311.