FAS Multigrid in Chombo

June 26, 2018

1 Introduction

This is an application with illustrates the use of the AMR Full Approximation Scheme (FAS) multigrid solver for nonlinear problems within Chombo (Adams et al., 2016).

Here, we first describe the nonlinear problem that the code solves (section 2). In section 3 we describe some of options available in the application, and in section 4 we demonstrate the accuracy of the code. Finally, in section 5, we describe how we implement the algorithm on an AMR hierarchy.

2 Example problem

We follow Henson (2002) and consider the nonlinear problem

$$-\nabla^2 u + \gamma\, u e^u = f, \tag{1}$$

where γ is a constant, f is some forcing term and u is the solution we wish to obtain. The domain is a 2D unit square ($0 < x < 1, 0 < y < 1$), and boundary conditions are given by $u = 0$ on all sides.

The size of γ determines the strength of the nonlinearity; when $\gamma = 0$ the problem reduces to solving a Poisson equation. We choose the forcing f such that the problem has an analytic solution, allowing us to determine the accuracy of our numerical scheme.

The example code contains two difference problems. The problem which will be solver is determined by the `main.problem` parameter in the inputs file, which should be set to either 0 for the ‘exact’ problem (section 2.1)or 1 for the ‘inexact’ problem’ (section 2.2).

2.1 Exact

Taking

$$u = (x - x^2)(y - y^2), \tag{2}$$

the forcing term becomes

$$f = 2\left[(x - x^2) + (y - y^2)\right] + \gamma(x - x^2)(y - y^2)e^{(x-x^2)(y-y^2)}. \qquad (3)$$

As Henson (2002) note, the solution here is a second order polynomial so there is no discretization error with our second order scheme. Hence, we refer to this test case as ‘exact’.

2.2 Inexact

Choosing

$$u = (x^2 - x^3)\sin(3\pi y), \qquad (4)$$

the forcing term becomes

$$f = \left\{\left[9\pi^2 + \gamma e^{(x^2-x^3)\sin(3\pi y)}\right](x^2 - x^3) + 6x - 2\right\}\sin(3\pi y), \qquad (5)$$

which will now have a notable discretization error.

3 Options

The `inputs` file controls various aspects of the application.

3.1 Grids

When `grids.max_level` is greater than 0, the grid hierarchy is determined by refining all cells where

$$\max(\nabla f)\Delta x > \texttt{grids.refine\_threshold}. \qquad (6)$$

Alternatively, you can specify the desired grids by setting `grids.read_in_grids=true` and using the format introduced at the end of the inputs file.

4 Tests

The python script `runTests.py` executes a convergence test on a fixed AMR hierarchy of grids and prints the results. A typical output for $\gamma = 100$ and the ‘exact’ test problem looks like

```
==================================
Convergence test
----------------------------------
Nx   | Max err  | L1       | L2       | Rate  || Runtime (s)  | Rate
16   | 1.67e-04 | 5.60e-05 | 7.28e-05 | 0.000 ||   0.103      | 0.000
32   | 5.11e-05 | 1.45e-05 | 1.92e-05 | 1.634 ||   0.066      | 0.159
```

```
64   | 1.40e-05 | 3.66e-06 | 4.88e-06 | 1.825 ||   0.086      | 0.330
128  | 3.66e-06 | 9.18e-07 | 1.23e-06 | 1.913 ||   0.219      | 0.633
256  | 9.34e-07 | 2.30e-07 | 3.07e-07 | 1.959 ||   0.656      | 0.748
512  | 2.36e-07 | 5.74e-08 | 7.67e-08 | 1.979 ||   2.555      | 0.974
1024 | 5.93e-08 | 1.44e-08 | 1.92e-08 | 1.990 ||  11.466      | 1.122
================================
```

The convergence of the error ϵ between successive computations, i, is calculated as

$$r_N = \frac{\epsilon(N_x^i)/\epsilon(N_x^{i-1})}{N_x^i/N_x^{i-1}}. \tag{7}$$

The ratio between successive runtimes τ is scaled with the number of grid cells

$$r_t = \frac{\tau(N_x^i)/\tau(N_x^{i-1})}{N_x^i/(N_x^{i-1})^2}. \tag{8}$$

5 FAS Multigrid algorithm

The FAS approach to geometric multigrid is a well established method for dealing with nonlinear problems - see e.g. Briggs et al. (2000); Henson (2002) for a detailed introduction. The application of this method to an AMR hierarchy requires some care, as noted by Martin and Cartwright (1996); Martin (1998) for linear multigrid problems. Here we describe the AMR FAS algorithm used in Chombo, as originally written by Mark Adams, following the format introduced in the Chombo Design document.

We wish to solve the equation

$$N^{comp}\psi^{comp} = \rho^{comp}, \tag{9}$$

on an AMR hierarchy $\{\Omega\}_{l=0}^{l_max}$. The algorithm (algorithm 1) proceeds in two stages. First we solve down to the coarsest AMR level, then the problem on this level is solved by standard FAS multigrid `FASVCycle`, before completing the AMR portion of the algorithm. In the psuedocode descriptions, the restriction R and interpolation I operators are as described in Adams et al. (2016). $N^{nf}(\psi^\ell, \psi^{\ell-1})$ is a two-level discretization of the nonlinear operator, whilst $N^\ell(\psi^\ell)$ is a single-level discretization. The crucial difference is that the two-level version includes a reflux correction.

Note that the `relax` procedure in both `AMRFASVCycle` and `FASVCycle` requires a coarse-fine boundary condition. During FAS multigrid it is the actual solution, not the error, which is smoothed, so boundary conditions are not simply $\psi = 0$. A box on an initially refined level $\ell + 1$ with a coarse-fine interface will retain its coarse-fine interface throughout the algorithm, so a boundary condition must be specified for smoothing steps. This boundary condition comes from level ℓ, which must be appropriately coarsened. The ability to coarsen this boundary condition places a limit on the maximum depth of `FASVCycle`.

Algorithm 1 AMR FAS algorithm. In `relax`, λ is the relaxation parameter for a Gauss-Seidel scheme. Note that bottom solver in FASVCycle is simply relaxation.

```
1: Residual $= \rho - N(\psi)$
2: **while** $||$Residual$|| > \epsilon||\rho||$ **do**
3:     AMRFASVCycle($\ell^{max}$)
4:     Residual $= \rho - N(\psi)$
5: **procedure** AMRFASVCYCLE(level $\ell$)
6:     **if** $\ell = \ell^{max}$ **then** $r^\ell = \rho^\ell$
7:     **if** $\ell > 0$ **then**
8:         relax($\psi^\ell$, $\psi^{\ell-1}$, $r^l$)
9:         $r^{\ell-1} = R(r^\ell) - N^{nf}(\psi^\ell, \psi^{\ell-1})$ on $\Omega^{\ell-1}$
10:        $r^{\ell-1} = R(r^\ell - N^{nf}(\psi^\ell, \psi^{\ell-1}))$ on $\mathcal{C}(\Omega^\ell)$
11:        $r^{\ell-1} = r^{\ell-1} + N^{\ell-1}(R(\psi^\ell))$ on $\mathcal{C}(\Omega^\ell)$
12:        $\psi^{\ell,save} = \psi^\ell$ on $\Omega^ell$
13:        AMRFASVCycle($\ell - 1$)
14:        $\psi^{\ell,corr} = I(\psi^{\ell-1}) - \psi^{\ell,save}$ on $\Omega^ell$
15:        $\psi^\ell = \psi^\ell + \psi^{\ell,corr}$ on $\Omega^ell$
16:        relax($\psi^\ell$, $\psi^{\ell-1}$, $r^l$)
17:    **else**
18:        FASVCycle($\psi^0, r^0$)
19: **procedure** FASVCYCLE($\psi^\ell$, $r^l$)
20:    $\psi^{\ell,save} = \psi^\ell$
21:    **if** $\ell = \ell^{\text{max depth}}$ **then**
22:        relax($\psi^\ell$, $\psi^{\ell-1}$, $r^\ell$)
23:    **else**
24:        relax($\psi^\ell$, $\psi^{\ell-1}_{valid}$, $r^\ell$)
25:        $r^{\ell-1} = R(r^\ell - N^\ell(\psi^\ell) + N^{\ell-1}(R(\psi^\ell))$
26:        FASVCycle($R(\psi^\ell)$, $r^{\ell-1}$)
27:        $\psi^\ell = \psi^\ell + I(e^{\ell-1})$
28:        relax($\psi^\ell$, $\psi^{\ell-1}_{valid}$, $r^\ell$)
29:    $e^\ell = \psi^\ell - \psi^{\ell,save}$
30: **procedure** RELAX($\psi^\ell$, $\psi^{\ell-1}$, $r^l$)
31:    Fill $\psi^\ell$ BCs using quadratic interpolation with $\psi^{\ell-1}$
32:    $\psi^\ell = \psi^\ell + \lambda(N^{nf}(\psi^\ell, \psi^{\ell-1}) - r^\ell)$ on $\Omega^\ell$
```

References

M. Adams, P. Colella, D. T. Graves, J. N. Johnson, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document, 2016.

William L Briggs, Steve F McCormick, and Van E Henson. *A multigrid tutorial*. Society for Industrial and Applied Mathematics, 2000. ISBN 9780898714623.

V E Henson. Multigrid Methods for Nonlinear Problems: An Overview. 2002. URL `http://www.osti.gov/scitech/biblio/15002749`.

D. F. Martin. *An Adaptive Cell-Centered Projection Method for the Incompressible Euler Equations*. PhD thesis, University of California at Berkeley, 1998.

D F Martin and K L Cartwright. Solving Poisson's Equation using Adaptive Mesh Refinement. Technical report, EECS Department, University of California, Berkeley, 1996. URL `papers3://publication/uuid/1E791D61-2448-407E-8FDE-B4785A7C78EA`.