Self-Gravitating AMR Algorithm Specification

Dan Martin and Phil Colella
Applied Numerical Algorithms Group *

August 8, 2003

Contents

*This work supported by the NASA Earth and Space Sciences Computational Technologies Program and by the U.S. Department of Energy: Director, Office of Science, Office of Advanced Scientific Computing Research under Contract DE-AC02-05CH11231.

1 Equations

We are solving the compressible flow equations with self-gravity:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F} = \boldsymbol{S}_U \tag{1}$$

$$\frac{\partial \boldsymbol{W}}{\partial t} + \sum_{d=0}^{D-1} A_d \frac{\partial \boldsymbol{W}}{\partial x_d} = \boldsymbol{S}_W \tag{2}$$

$$\Delta\phi = 4\pi\mathcal{G}\rho \tag{3}$$

where the conserved variables are $\boldsymbol{U} = (\rho, \rho\vec{u}, \rho e)$, the primitive variable vector is $\boldsymbol{W} = (\rho, \vec{u}, p)$, and $\mathcal{G}$ is the gravitational constant. The source terms due to self-gravity are: $\boldsymbol{S}_U = (0, -\rho\nabla\phi, -\rho\vec{u}\cdot\nabla\phi)$ and $\boldsymbol{S}_W = (0, -\nabla\phi, 0)$.

2 Algorithm

We use the basic AMR multilevel advance scheme outlined in [1] as implemented in [3], which consists of recursive single-level advance steps followed by a set of synchronization operations when two or more levels reach the same solution time. The only substantive difference between the algorithm presented here and that in [1] is due to the presence of the self-gravity source term S.

2.0.1 Variables

We have two separate versions of ϕ, the self-gravity potential. $\phi^{comp,\ell}(t)$ is computed using a multilevel elliptic solve:

$$L^{comp}\phi^{comp} = 4\pi\mathcal{G}\rho \quad \text{for} \quad \ell_{base} \le \ell \le \ell_{max}, \tag{4}$$

where L^{comp} is a multilevel operator. $\phi^{\ell}(t)$ is computed using a single-level elliptic solve:

$$L^{\ell}\phi^{\ell} = 4\pi\mathcal{G}\rho^{\ell} \quad \text{on } \Omega^{\ell}, \tag{5}$$

where Ω^{ℓ} is the set of cells which have been refined to level ℓ. The difference between the two is denoted by $\delta\phi^{\ell} = \phi^{comp,\ell} - \phi^{\ell}$.

The single-level advance for level ℓ will advance the solution on that level from time t^{ℓ} to $t^{\ell} + \Delta t^{\ell}$. At the beginning of the level advance, we have

the conserved variables at the old time $\boldsymbol{U}^\ell(t^\ell)$, along with an old-time approximation to the self-gravity potential term $(\nabla\phi)^{OLD}$, the level-operator version of the potential $\phi^{\ell,OLD}$, and the correction term $\delta\phi$, which was computed during a previous synchronization step. A pseudocode description of the recursive algorithm for the single-level advance may be found in Figure 1.

---

procedure **advance** (ℓ)
  $U^\ell(t^\ell + \Delta t^\ell) = U^\ell(t^\ell) - \Delta t D\vec{F}^\ell + \Delta t^\ell S_U^{OLD,\ell}$ on Ω^ℓ
  Solve $L^\ell\phi^{NEW,\ell} = 4\pi\mathcal{G}\rho(t^\ell + \Delta t^\ell)$
  if $\ell < \ell_{max}$
    $\delta F_d^{\ell+1} = -F_d^\ell$ on $\zeta_{+,d}^{\ell+1} \cup \zeta_{-,d}^{\ell+1}, d = 0, ..., \mathbf{D} - 1$
  end if
  if $\ell > 0$
    $\delta F_d^\ell := \frac{1}{n_{ref}^{\ell-1}} < F_d^\ell >$ on $\zeta_{+,d}^\ell \cup \zeta_{-,d}^\ell, d = 0, ..., \mathbf{D} - 1$
  end if
  for $q = 0, ..., n_{ref}^\ell - 1$
    advance$(\ell + 1)$
  end for
  synchronize(ℓ)
  $t^\ell := t^\ell + \Delta t^\ell$
  $n_{step}^\ell := n_{step}^\ell + 1$
  if $(n_{step}^\ell = 0 \bmod n_{regrid})$ and $(n_{step}^{\ell-1} \neq 0 \bmod n_{regrid})$
    regrid(ℓ)
  end if
end **advance**

Figure 1: Pseudo-code description of the recursive single-level advance for hyperbolic conservation laws with self-gravity.

2.1 Single-Level Advance

First, we compute the second-order upwinded hyperbolic fluxes $\boldsymbol{F}^{half,\ell}$ in the same way as in [3]. We then can compute the conservative updates

$\boldsymbol{U}^\ell(t^\ell + \Delta t^\ell)$:

$$\boldsymbol{U}^\ell(t^\ell + \Delta t^\ell) = \boldsymbol{U}^\ell(t^\ell) - \Delta t D^\ell \boldsymbol{F}^{half,\ell} + \Delta t^\ell \boldsymbol{S}_U^{old,\ell}, \tag{6}$$

where

$$\boldsymbol{S}^{old,\ell} = (0, \rho^{half,\ell}(\nabla\phi^{comp})^\ell(t^\ell), \rho^{half,\ell}\vec{u}^{half,\ell} \cdot (\nabla\phi^{comp})^\ell(t^\ell)) \tag{7}$$

$$\rho^{half,\ell} = \frac{1}{2}(\rho^\ell(t^\ell) + \rho^\ell(t^\ell + \Delta t^\ell)) \tag{8}$$

$$\vec{u}^{half,\ell} = \frac{1}{2}(\vec{u}^\ell(t^\ell) + \vec{u}^\ell(t^\ell + \Delta t^\ell)) \tag{9}$$

Note that with the proper ordering of the update, this can be computed explicitly. First, compute the update for ρ. Once $\rho^\ell(t^\ell + \Delta t^\ell)$ has been computed, (8) can be evaluated, and the source term for the momentum update may be computed. Once the momentum update has been computed, (9) may be evaluated, and the energy update may be computed.

Also, once the fluxes have been computed, the flux registers $\delta\boldsymbol{F}_U$ are incremented appropriately, as in [3].

Finally, compute an updated value for ϕ^ℓ by performing a single-level elliptic solve using single-level operators:

$$L^\ell\phi^\ell(t^\ell + \Delta t^\ell) = 4\pi\mathcal{G}\rho \tag{10}$$

If $\ell > 0$ then use quadratic coarse-fine interpolation boundary conditions with a time-interpolated coarse-level ϕ with t the time at which the solve is being performed ($t = t^\ell + \Delta t^\ell$ in this case):

$$\phi^\ell(t) = I(\phi^\ell(t), \frac{t - (t^{\ell-1} - \Delta t^{\ell-1})}{\Delta t^{\ell-1}}\phi^{\ell-1}(t^{\ell-1}) + \frac{t^{\ell-1} - t}{\Delta t^{\ell-1}}\phi^{\ell-1}(t^{\ell-1} - \Delta t^{\ell-1}) + \delta\phi^{\ell-1}) \tag{11}$$

Since ϕ^ℓ is only necessary for computing coarse-fine boundary conditions with finer levels (the source terms in the updates are computed using $\nabla\phi^{comp}$), ϕ^ℓ need only be computed if a finer level exists.

Then, if a finer level exists, it is advanced n_{ref}^ℓ times with $\Delta t^{\ell+1} = \frac{1}{n_{ref}^\ell}\Delta t^\ell$. Once any finer levels have been advanced to time $t^\ell + \Delta t^\ell$, we perform the synchronization step.

2.2 Synchronization

There are four parts to the synchronization in this algorithm. First, a flux-correction step is performed to maintain conservation. The gravitational potential term ϕ is computed using a multilevel elliptic solve. Then, a correction to the source term is computed to make the update second-order accurate in time. Finally, finer-level solutions are averaged down to the covered regions of coarser levels. A pseudocode description of the synchronization step may be found in Figure 2.

---

procedure **synchronize** (ℓ)
  $U^\ell(t^\ell + \Delta t^\ell) := U^\ell(t^\ell + \Delta t^\ell) - \Delta t^\ell D_R(\delta F^{\ell+1})$
  if $(\ell = \ell_{base})$
    Solve $L^{comp}\phi^{comp} = 4\pi\mathcal{G}\rho$ for $\ell \geq \ell_{base}$
    for $\ell = \ell_{base}, \ell_{max}$
      $\delta\phi^\ell = \phi^{comp} - \phi^\ell$
      compute $(\nabla\phi)^{\ell,NEW}$
      $\delta(\nabla\phi)^\ell = (\nabla\phi)^{NEW,\ell} - (\nabla\phi)^{OLD,\ell}$
      compute $\delta\boldsymbol{S}^\ell$
      $\boldsymbol{U}^\ell := \boldsymbol{U} + \frac{\Delta t}{2}\delta\boldsymbol{S}^\ell$
    end for
  end if
  $U^\ell(t^\ell + \Delta t^\ell) = Average(U^{\ell+1}(t^\ell + \Delta t^\ell), n^\ell_{ref})$ on $\mathcal{C}_{n^\ell_{ref}}(\Omega^{\ell+1})$
end **synchronize**

Figure 2: Pseudo-code description of the sychronization portion of the algorithm for hyperbolic conservation laws with self-gravity. Note that much of the synchronization computation is only carried out if $(\ell = \ell_{base})$, where ℓ_{base} is the coarsest level which has reached the synchronization time.

Note that synchronization is a multilevel operation which is performed for all levels which have reached the time t^{sync}. In practice, this is accomplished by checking to see if the current level ℓ has been advanced to the same time as the next coarser level $\ell - 1$. If this is the case, we drop down to the next coarser level. The coarsest level which has reached t^{sync} is denoted as ℓ_{base}.

2.2.1 Reflux for conservation

As in [3], a refluxing operation is performed to ensure conservation at coarse-fine interfaces:

$$\boldsymbol{U}^{\ell} = \boldsymbol{U} - \Delta t D_R(\delta \boldsymbol{F}_U^{\ell+1}) \tag{12}$$

2.2.2 Computation of ϕ^{comp}

We also compute ϕ^{comp} using multilevel operators:

$$L^{comp}\phi^{comp}(t^{sync}) = 4\pi\mathcal{G}\rho \quad \text{for} \quad \ell \geq \ell_{base} \tag{13}$$

If $\ell_{base} > 0$, then we use the coarse-fine boundary condition (11) for $\ell = \ell_{base}$ and $t = t^{sync}$.

We use the composite and level-operator solutions for ϕ to compute $\delta\phi$, which is a correction to a level-operator-computed ϕ to account for the effects of finer levels:

$$\delta\phi^{\ell} = \phi^{comp,\ell}(t^{sync}) - \phi^{\ell}(t^{sync}) \quad \text{for} \quad \ell \geq \ell_{base} \tag{14}$$

Also, we then compute the gradients used in the flux computation using the composite solution:

$$(\nabla\phi^{comp})^{\ell}(t^{sync}) = \begin{cases} Av^{F\to C}(G^{comp}\phi^{comp}(t^{sync})) & \text{in uncovered regions} \\ Av^{F\to C}(\langle G^{comp}\phi^{comp}(t^{sync})\rangle) & \text{on covered regions} \end{cases} \tag{15}$$

where G^{comp} is the face-centered composite gradient operator, $\langle..\rangle$ is the appropriate averaging/coarsening operator from level $\ell+1$ to level ℓ, and $Av^{F\to C}$ is the face-to-cell averaging operator.

2.2.3 Source Term Correction

To make the scheme second-order in time, we then compute a correction to the source-term treatment so the final update uses the midpoint rule. We first compute the difference between the old-time and new-time $\nabla\phi$ and use this to compute a source-term correction:

$$\delta(\nabla\phi)^{\ell} = (\nabla\phi^{comp})^{\ell}(t^{\ell}) - (\nabla\phi^{comp})^{\ell}(t^{\ell} - \Delta t^{\ell}) \quad \text{for} \quad \ell \geq \ell_{base} \tag{16}$$

$$\delta\boldsymbol{S}_U^{\ell} = (0, \rho^{half,\ell}\delta(\nabla\phi)^{\ell}, \rho^{half,\ell}\vec{u}^{half,\ell}\delta(\nabla\phi)^{\ell}) \tag{17}$$

Finally, we apply the correction:

$$\boldsymbol{U}^{\ell}(t^{\ell}+\Delta t^{\ell}) := \boldsymbol{U}^{\ell}(t^{\ell}+\Delta t^{\ell}) + \frac{\Delta t^{\ell}}{2}\delta \boldsymbol{S}_{U}^{\ell} \quad \text{for} \quad \ell \geq \ell_{base} \tag{18}$$

2.2.4 Average finer solution to coarser levels

At this point, the composite solution has been updated, so we then average the finer-level solution onto covered regions of coarser levels.

3 Software Design

The self-gravity code will be implemented using the `AMRGodunov` code framework [3]. The only real structural difference between the `AMRPolytropicGas` example and the self-gravity code example is the addition of the self-gravity terms, and the use of the `AMRSolver` and `LevelSolver` elliptic solver classes from Chombo [2] to manage the elliptic solves required to compute the various forms of the gravitational potential used in this algorithm. A basic diagram of the class relationships between the `Chombo` and `SelfGravity` classes is depicted in Figure 3.

References

[1] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, May 1989.

[2] P. Colella, D. T. Graves, T. J. Ligocki, D. F. Martin, D. Modiano, D. B. Serafini, and B. Van Straalen. Chombo Software Package for AMR Applications - Design Document. unpublished, 2000.

[3] P. Colella, D. T. Graves, T. J. Ligocki, and B. Van Straalen. AMR Godunov unsplit algorithm and implementation. unpublished, 2002.

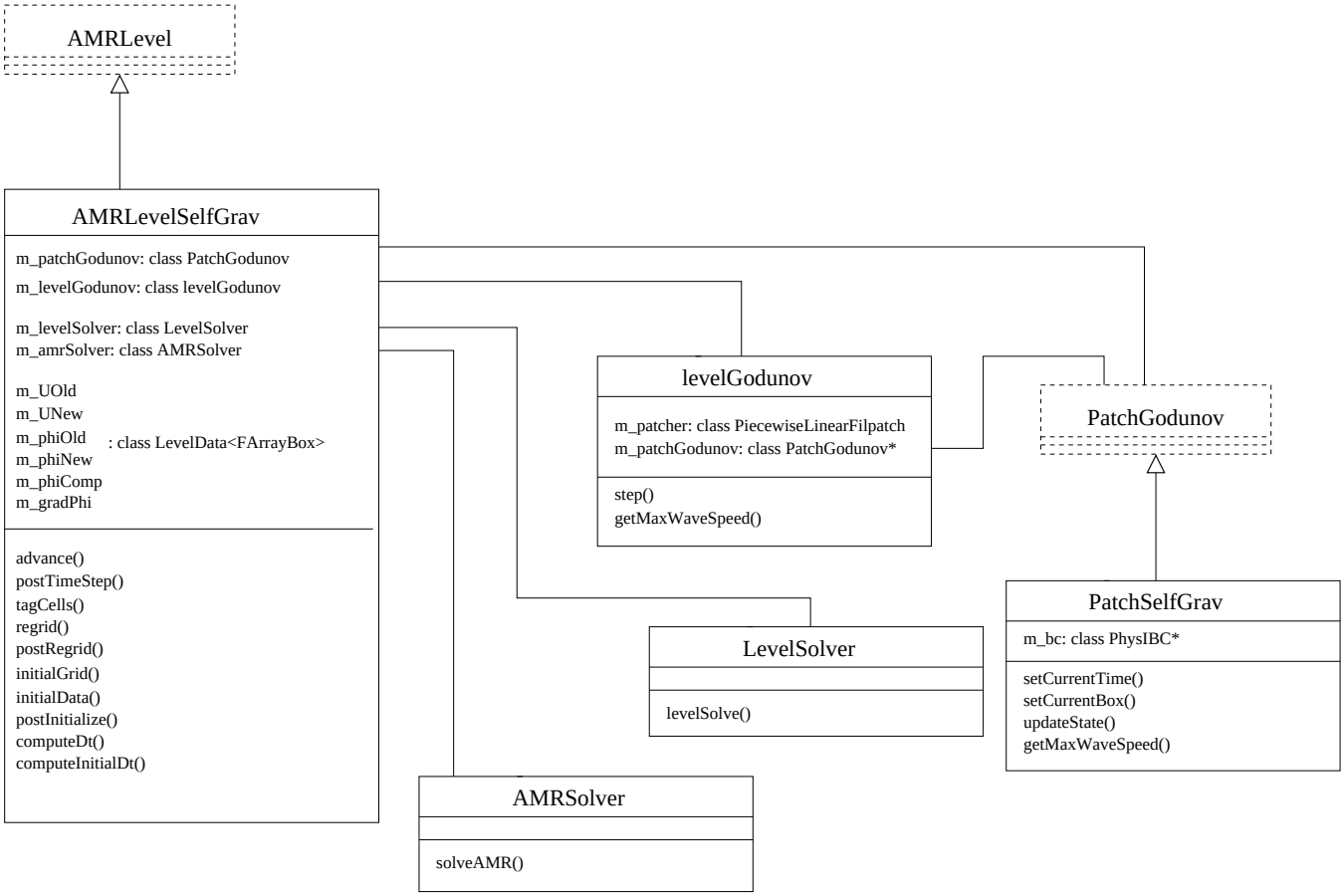

Figure 3: Software configuration diagram for the self-gravity code showing basic relationships between AMRGodunov classes and Chombo classes for the Self-Gravity example