A Summary of Choptuik's Implementation of Mesh Refinement for the Wave Equation in Spherical Symmetry

October 1994

1 Introduction

In the past few years, Matthew Choptuik has reported on some exciting new results in numerical relativity, namely, the discovery of critical phenomena in the black hole formation via. imploding scalar fields [2]. The success of his code depends crucially on the use of a mesh refinement algorithm due to Berger and Oliger [1]. Realizing that his code could be used as a benchmark for mesh refinement applied to numerical relativity, Choptuik presented a "stripped down" version of his code (`ad`) in March of 1994. This version takes out all of the physics complications by simply evolving a scalar field via. the wave equation in spherical symmetry. What is left is the "bare essentials" of the mesh refinement algorithm. This paper is a summary of the ideas behind Choptuik's code and the Berger and Oliger mesh refinement algorithm. The first two sections (Getting the Source Code and The Model Problem) are taken word for word from Choptuik's documentation that accompanies the code. The next section(s) describe the mesh refinement technique used.

2 Getting the Source Code(Choptuik)

Source code, documentation, sample control files and some utilities are available via anonymous `ftp` from `helmholtz.ph.utexas.edu`. It is assumed that the code will be installed on a "typical" scientific workstation (Sun, SGI, IBM RS6000, Dec Alpha, ...) running Unix, and that `C` and `FORTRAN 77` compilers and a `perl` interpreter are available.

To retrieve `ad`, `cd` to the directory where you want `ad` installed, then enter:

```
ftp helmholtz.ph.utexas.edu
anonymous
<user name>@
```

Once the `ftp` connection has been established, type

```
bin
cd pub
get ad.tar.Z
quit
```

Now execute

```
zcat ad.tar.Z | tar xvf -
```

to reconstruct the `ad` directory hierarchy. The file `ad/README` will tell you how to proceed.

3 The Model Problem(Choptuik)

`ad` solves the wave equation in spherical symmetry using second-order finite difference techniques and adaptive mesh refinement. The fundamental dynamical variable is a scalar field, $\phi(r,t)$, whose evolution is governed by

$$\frac{\partial^2 \phi}{\partial t^2} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 \frac{\partial \phi}{\partial r}\right) = 3\frac{\partial}{\partial r^3}\left(r^2 \frac{\partial \phi}{\partial r}\right) \tag{1}$$

I rewrite this equation in first-order (in time) form, by introducing auxiliary variables Φ and Π defined by:

$$\Phi \equiv \frac{\partial \phi}{\partial r} \qquad\qquad \Pi \equiv \frac{\partial \phi}{\partial t} \tag{2}$$

With these definitions, equation (1) may be rewritten as:

$$\frac{\partial \Phi}{\partial t} = \frac{\partial \Pi}{\partial r} \qquad\qquad \frac{\partial \Pi}{\partial t} = 3\frac{\partial}{\partial r^3}\left(r^2 \Phi\right) \tag{3}$$

The wave equation is solved on the finite domain $0 \le r \le R$, $0 \le t \le T$. Introducing a standard finite difference notation for grid functions (with a `FORTRAN` convention for the spatial grid index):

$$f_j^n \equiv f\left(r_j, t^n\right) \equiv f\left((j-1)\,\triangle r\,, n\,\triangle t\,\right) \tag{4}$$

where $\triangle r$ and $\triangle t$ are the *uniform* spatial (radial) and temporal mesh-spacings respectively, I use the following second-order finite difference approximations of (3):

$$\frac{\Phi_j^{n+1} - \Phi_j^{n-1}}{2\,\triangle t} = \frac{\Pi_{j+1}^n - \Pi_{j-1}^n}{2\,\triangle r} \tag{5}$$

$$\frac{\Pi_j^{n+1} - \Pi_j^{n-1}}{2\,\triangle t} = 3\,\frac{r_{j+1}^2\,\Phi_{j+1}^n - r_{j-1}^2\,\Phi_{j-1}^n}{r_{j+1}^3 - r_{j-1}^3} \tag{6}$$

Note that the spatial index, j, runs over the values $1, 2, \ldots,$ `NR`, where `NR` $\equiv R/\triangle r + 1$, and the temporal index, n, runs over the values $0, 1 \ldots,$ `NT`, where `NT` $\equiv T/\triangle t + 1$. (5) and (6) are trivially solved for Φ_j^{n+1} and Π_j^{n+1}, yielding the basic algebraic equations used to advance the solution from one time-step to the next:

$$\Phi_j^{n+1} := \Phi_j^{n-1} + \frac{\triangle t}{\triangle r}\left(\Pi_{j+1}^n - \Pi_{j-1}^n\right) \tag{7}$$

$$\Pi_j^{n+1} := \Pi_j^{n-1} + \frac{6\,\triangle t}{r_{j+1}^3 - r_{j-1}^3}\left(r_{j+1}^2\,\Phi_{j+1}^n - r_{j-1}^2\,\Phi_{j-1}^n\right) \tag{8}$$

(see `subroutine elfppi()` and `subroutine elfpii()` in `unigrid.f`). Equations (7–8) are used to update *interior* grid function values: i.e. those with spatial grid indexes $2 \le j \le$ `NR`. At the boundaries $j = 1$, $j =$ `NR`, the following additional conditions are used (see `subroutine elfpp()` and `subroutine elfpi()` in `unigrid.f`):

$$\Phi_1^{n+1} := 0 \tag{9}$$

$$\Pi_1^{n+1} := \frac{4\Pi_2^{n+1} - \Pi_3^{n+1}}{3} \tag{10}$$

$$\Phi^{n+1}_{\mathtt{NR}} := \left(\frac{3}{\triangle t}+\frac{3}{\triangle r}+\frac{2}{r_{\mathtt{NR}}}\right)^{-1}\left(\frac{4\Phi^{n}_{\mathtt{NR}}-\Phi^{n-1}_{\mathtt{NR}}}{\triangle t}+\frac{4\Phi^{n+1}_{\mathtt{NR-1}}-\Phi^{n+1}_{\mathtt{NR-2}}}{\triangle r}\right) \tag{11}$$

$$\Pi^{n}_{\mathtt{NR}} := \left(\frac{3}{\triangle t}+\frac{3}{\triangle r}+\frac{2}{r_{\mathtt{NR}}}\right)^{-1}\left(\frac{4\Pi^{n}_{\mathtt{NR}}-\Pi^{n-1}_{\mathtt{NR}}}{\triangle t}+\frac{4\Pi^{n+1}_{\mathtt{NR-1}}-\Pi^{n+1}_{\mathtt{NR-2}}}{\triangle r}\right) \tag{12}$$

Equations (9–10) are based on regularity of the scalar field ϕ at the origin:

$$\lim_{r\to 0}\ \phi(r,t) = \phi_0(t) + r^2\,\phi_2(t) + \cdots \tag{13}$$

while (11–12) are based on the (presumed) outgoing-only nature of solutions of (1) at large r

$$\lim_{r\to\infty}\ r\phi(r,t) = f(r-t) \tag{14}$$

which implies

$$\lim_{r\to\infty}\ \frac{\partial\Phi}{\partial t} + \frac{\partial\Phi}{\partial r} + \frac{\Phi}{r} = O\left(r^{-3}\right) \approx 0 \tag{15}$$

$$\lim_{r\to\infty}\ \frac{\partial\Pi}{\partial t} + \frac{\partial\Pi}{\partial r} + \frac{\Pi}{r} = 0 \tag{16}$$

Equations (11–12) follow from (15–16) by replacement of first derivatives with second-order *backwards* differences applied at $(r_{\mathtt{NR}}, t^{n+1})$.

In order to regularize (damp high frequency components) the numerical solution, `ad` applies a form of dissipation advocated by Kreiss & Oliger [3] for use with wave-equation discretizations such as (7–8). For example, following the computation of the advanced values Φ^{n+1}_j from (7), dissipation is added via:

$$\Phi^{n+1}_j := \Phi^{n+1}_j - \frac{\epsilon}{16}\left(\Phi^{n-1}_{j+2} - 4\Phi^{n-1}_{j+1} + 6\Phi^{n-1}_{j} - 4\Phi^{n-1}_{j-1} + \Phi^{n-1}_{j-2}\right) \tag{17}$$

where $\epsilon > 0$ is an adjustable parameter which must satisfy $\epsilon < 1$ for stability. A precisely analogous update is applied to Π^{n+1}_j. The value of ϵ is implemented as a run-time parameter, `EPSDIS`, which is read from the control file `ad.control`. (See `subroutine adddis()` in `unigrid.f` and calls to `adddis()` in `ad.f` and `ctres.f`).

`ad` also uses the following discretization of (2b) to maintain an approximation of the scalar field ϕ itself:

$$\Pi^n_j = \frac{\phi^{n+1}_j - \phi^{n-1}_j}{2\,\triangle t} \tag{18}$$

This yields algebraic equations for the update of the mesh quantities ϕ^{n+1}_j:

$$\phi^{n+1}_j := \phi^{n-1}_j + 2\,\triangle t\,\Pi^n_j \tag{19}$$

which are implemented in `subroutine elfph()` in `unigrid.f`.

Finally, chiefly for diagnostic purposes, `ad` computes approximations to (i) the radial energy density, dm/dr, (ii) the spatial integral, m (known as the *mass aspect* function) of dm/dr and (iii) m/r. The following discrete forms are used (see `subroutine clcmas()` in `unigrid.f`):

$$\left.\frac{dm}{dr}\right.^{n+1}_{j} := 2\pi r_j^2\left({\Phi^{n+1}_j}^2 + {\Pi^{n+1}_j}^2\right) \tag{20}$$

$$m_j^{n+1} := m_{j-1}^{n+1} + \frac{1}{2}\left(r_j - r_{j-1}\right)\left(\frac{dm}{dr}{}_j^{n+1} - \frac{dm}{dr}{}_{j-1}^{n+1}\right) \tag{21}$$

$$\frac{m}{r}{}_j^{n+1} := \frac{m_j^{n+1}}{r_j} \tag{22}$$

4 Grid Description and Data Structures

The program `ad`maintains a tree structure of nested, aligned, uniform meshes. They are uniform in the sense of scale: all meshes have the same ratio $\Delta t/\Delta r$. Each grid is characterized by the level, L, of discretization where $L = 1$ is the coarsest level of discretization. An example grid with three levels of refinement $(L = 1, 2, 3)$ is:

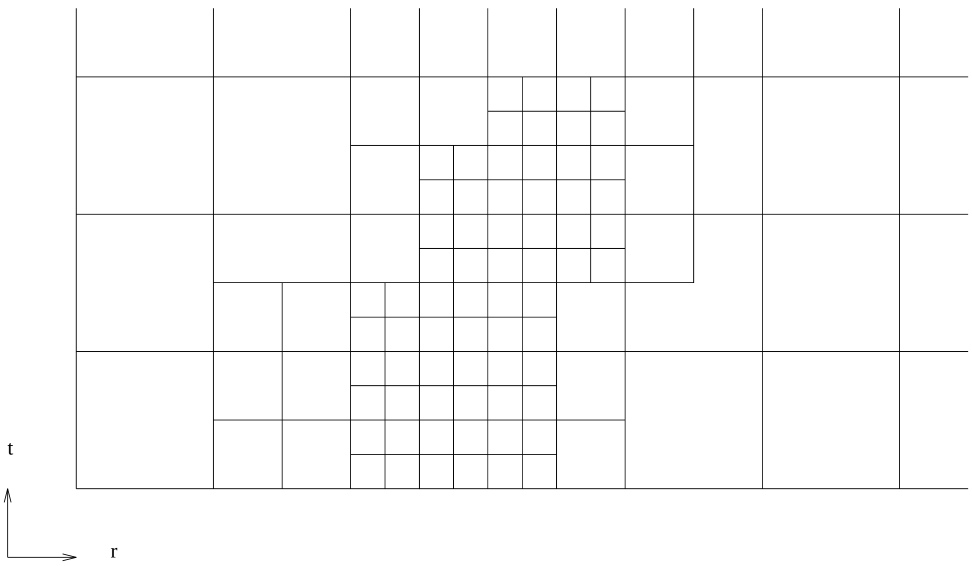
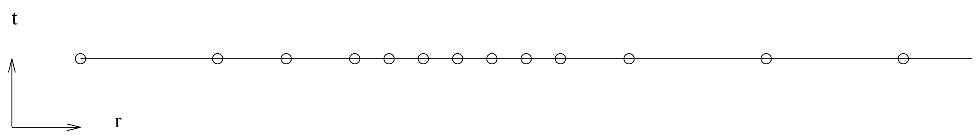


Fig. 1

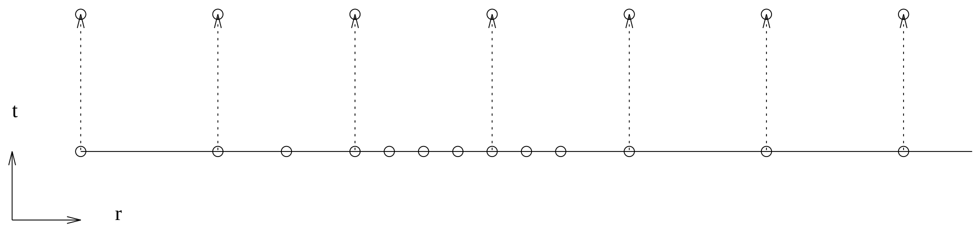
The data structure used to keep track of all of the subgrids is an ordered tree data structure, where each node represents a grid. Each node can also have multiple descendents. In this representation, we see that a node will have multiple descendents if the coarse grid corresponding to that node has several fine grids containded in it.

5 Integration Algorithm

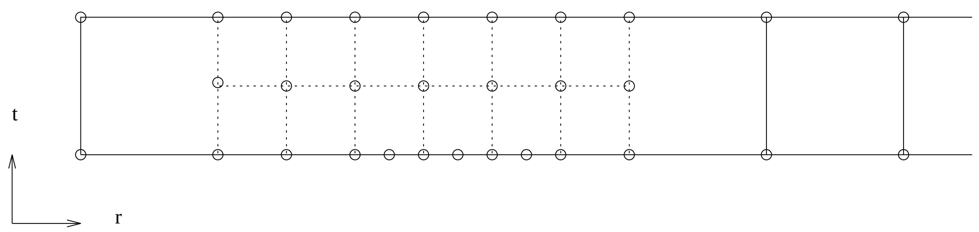
The evolution scheme is executed by starting at the coarsest grid level and working down through to the finest grid level. The algorithm is best demonstrated by example. Lets say we wanted to evolve a grid with refinement shown in Fig. 1. The initial spatial surface has the following structure:



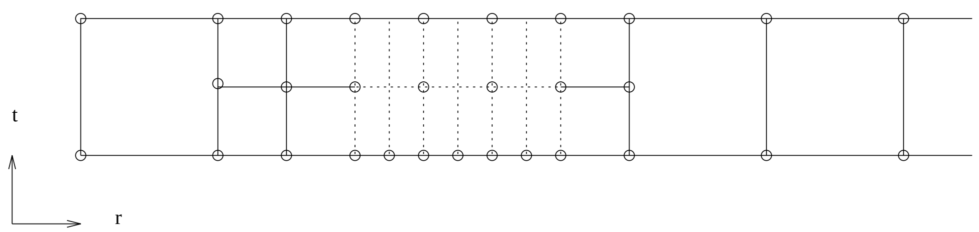
The first level grid ($L = 1$) is evolved, using the finite difference updating equations:



The next level ($L = 2$) is evolved, using extrapolated values of the field on the coarse grid as boundary values for the fine grid:



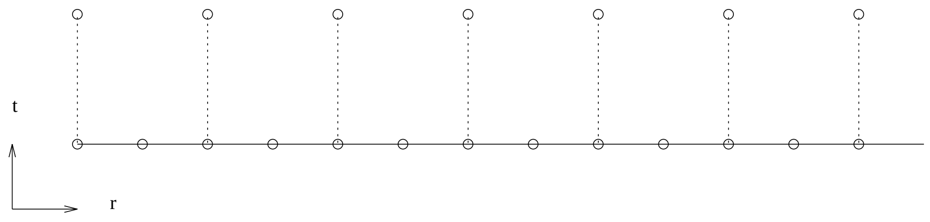
At any time during the updating procedure where a fine grid evolves the same point as a coarser grid, the value found during the fine grid evolution takes precedence and overwrites the field value found during the coarser grid evolution. Finally the last level ($L = 3$) is evolved, again using the extrapolated values of the field on the coarser grid it is contained in as boundary values:



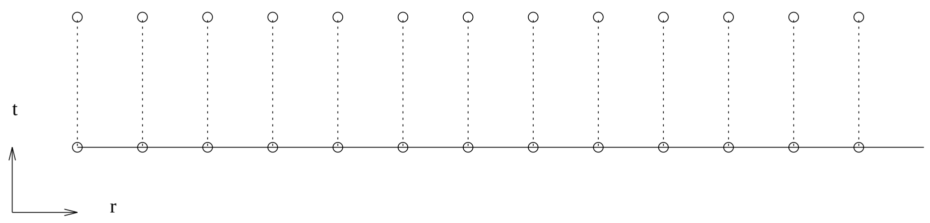
6 Grid Refinement: Flagging Points that Need Refinement

Deciding which points are in need of refinement is done by approximating the local truncation error using Richardson extrapolation. The basic idea is to integrate a grid using only half of the spatial resolution as would normally be used (i.e. ignoring every other spatial grid point). The result is then compared to the result obtained by integrating using the normal spatial resolution (using all grid points). The results are pointwise compared, and if the difference between the two is larger than some predecided tolerence level, the point is flagged for regridding.

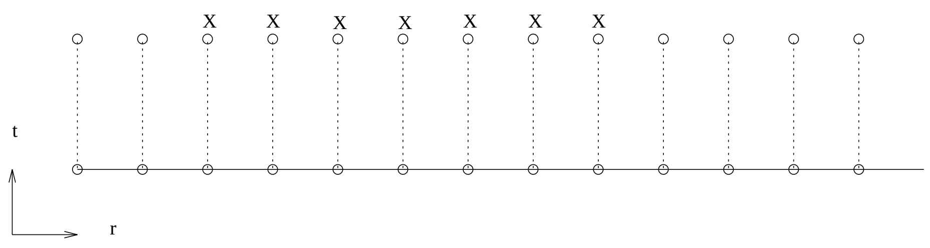
First, evolve using half of the spatial resolution ...



Next, evolve using full spatial resolution ...



Finally, flag grid points with local truncation error (difference between field values in different resolutions) above tolerance ...



References

[1] M. J. Berger and J. Oliger, *J. Comp. Phys*, **53**, 484 (1984)

[2] M. W. Choptuik, *Phys. Rev. Lett.*, **70**, 9–12 (1993)

[3] H.-O. Kreiss and J. Oliger, **Methods for the Approximate Solution of Time Dependent Problems**, GARP Publications Series No. 10, (1973)