ELSEVIER

Computers & Geosciences 34 (2008) 8–13

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

AGRIF: Adaptive grid refinement in Fortran ☆

Laurent Debreu*, Christophe Vouland, Eric Blayo

Projet IDOPT, Laboratoire de Modélisation et Calcul, 51 rue des Mathématiques, 38400 Saint Martin d'Hères, France

Received 9 November 2005; received in revised form 17 January 2007; accepted 18 January 2007

Abstract

Adaptive grid refinement in Fortran (AGRIF) is a Fortran90 package for the integration of adaptive mesh refinement (AMR) features within existing finite difference codes. The package first provides model-independent Fortran90 procedures containing the different operations in an AMR process: time integration of grid hierarchy, clustering, interpolations, updates, etc. The package then creates the Fortran90 model-dependent part of the code based on an entry file written by the user.

The basic idea of AGRIF is to make use of Fortran90 pointers to successively address the variables of the different grids of an AMR process. As pointers can be used exactly like other (static) variables in Fortran, most of the original code will remain unchanged.

Keywords: Adaptive mesh refinement; Structured grids; Fortran90; Pointers

1. Introduction

Discretization errors of partial differential equations may often be written as an increasing function of the computational grid resolution. One way to build efficient numerical algorithms is to start with a relatively coarse resolution grid which covers the entire domain of computation and then to locally increase the resolution in certain areas. For time-dependent problems, adaptive mesh refinement (AMR) methods allow the resolution to be dynamically adjusted. Since the work of Berger and Oliger (1984), AMR for structured grids has become very popular. AMR has a broad domain of application and its efficiency has been clearly demonstrated. However, implementation of the AMR method remains a difficult task, which often discourages model developers from investigating its potential power.

Hereafter, we will denote the Berger and Oliger AMR algorithm for structured meshes by BOSAMR. AMRCLAW is a Fortran77 package for conservation laws that was based on the BOSAMR algorithms. AMRCLAW is now included in CLAWPACK.[1] With the relatively recent appearance of object-oriented languages, programming BOSAMR methods has become simpler, and several packages have since been developed. Most of them are implemented in C++, which appeared before

☆ http://www-lmc.imag.fr/IDOPT/AGRIF, http://www.iamg.org/CGEditor/index.htm.

*Corresponding author. Tel.: +33 4 76 51 48 60; fax: +33 4 76 63 12 63.

E-mail address: Laurent.Debreu@imag.fr (L. Debreu).

[1] Leveque, R.J., 2006. CLAWPACK (Conservation Law Package) v4.3. http://www.amath.washington.edu/~claw.

0098-3004/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2007.01.009

Fortran90. The two most frequently used are Overture[2] and Chombo.[3] These packages basically provide C++ classes dealing with grid definition and grid operations.

As for similar tools in Fortran90, there is PARAMESH,[4] which is an AMR package based on single cell refinement. This approach can lead to highly efficient parallelization and highly flexible refinement. However it differs from the original idea of Berger and Oliger (1984), which was to use rectangular patches (composed of several cells), on which the original uniform grid code could be applied.

In this paper, we introduce a novel approach that allows the implemention of BOSAMR features in an existing Fortran model (77 or 90). Central to the method are the use of pointers and the possibility of taking advantage of the full compatibility between Fortran77 and Fortran90. These ideas are incorporated in the adaptive grid refinement in Fortran (AGRIF) software.

The present note is organized as follows: the main operations involved in a BOSAMR process are briefly reviewed in Section 2, while Section 3 describes the basic ideas underlying the AGRIF package.

2. Brief description of the BOSAMR method

In this section, we briefly summarize the ideas underlying the AMR method for structured grids. A full description is provided in Berger and Oliger (1984) and Berger and Colella (1989). The main idea is to locally adjust the numerical resolution using a mathematical and/or physical criterion, and thereby create refined grids (or remove existing ones) where and when necessary. The AMR strategy features a hierarchy of resolution levels, each of which contains a set of grids (cf. Fig. 1).

Every grid is covered by a parent (coarser) grid, the root level consisting of one coarse resolution grid covering the entire domain of computation. The spatial and temporal resolutions are divided by a given integer r when going from a given level l to a finer level $l+1$ (typically $r=2,3$ or 4). If Δh_l and Δt_l denote the grid spacing and the time step on every grid at level l, then $\Delta h_l/\Delta h_{l+1}=\Delta t_l/\Delta t_{l+1}=r$.

With such a convention, once the grid spacing and the time step have been chosen on the root grid, the corresponding stability criterion based on the Courant–Friedrichs–Levy (CFL) parameter is automatically fulfilled for all grids.

2.1. Integration of the grid hierarchy

The integration of the grid hierarchy starts at the root level. The solution on the root grid is first advanced one coarse time step Δt_0. This solution is then used to provide (by interpolation) boundary conditions for the grids at level 1, which can then be advanced r time steps $\Delta t_1=\Delta t_0/r$, and so on recursively for grids at deeper levels. Once a solution has been computed on a grid at a given level, it is used to update the corresponding solution on its parent grid.

The integration algorithm is summarized in Fig. 2. It is a recursive procedure which is called at level 0.

2.2. Regridding

As mentioned previously, the grid hierarchy may change at every coarse time step or at regular intervals. A refinement criterion is applied, which detects grid points where a finer grid is necessary or where an existing fine grid is no longer useful. Then the clustering procedure (which creates the new rectangular grids) is applied to each level. The AGRIF software implements the clustering algorithm described in Berger and Rigoutsos (1992).

3. Description of AGRIF

3.1. AMR in Fortran90

The underlying aim of AGRIF is to make as few changes as possible in the existing Fortran code. To this end, we use the concept of pointers.

Pointer variables were not part of the standard Fortran77 syntax, but were introduced in an extension to Fortran77, where they were termed *Cray Pointers*, and were finally incorporated in the standard version of Fortran90.

A pointer variable is able to address different locations in the memory during the computation. This is exactly what we would like to do in a BOSAMR process, when going from the computation on one grid level to the computation on

[2]Brown, D., Henshaw, W., Quinlan, D., 2006. Overture v21. http://www.llnl.gov/CASC/Overture.

[3]Colella, P., Graves, D.T., Ligocki, T.J., Martin, D.F., Modiano, D., Serafini, D.B., Van Straalen, B., 2007. Chombo v2.0. http://seesar.lbl.gov/anag/chombo/index.html.

[4]PARAMESH: MacNeice, P., Olson, K., 2007. PARAMESH v4.0. http://www.physics.drexel.edu/~olson/paramesh-doc/Users_manual/amr.html.

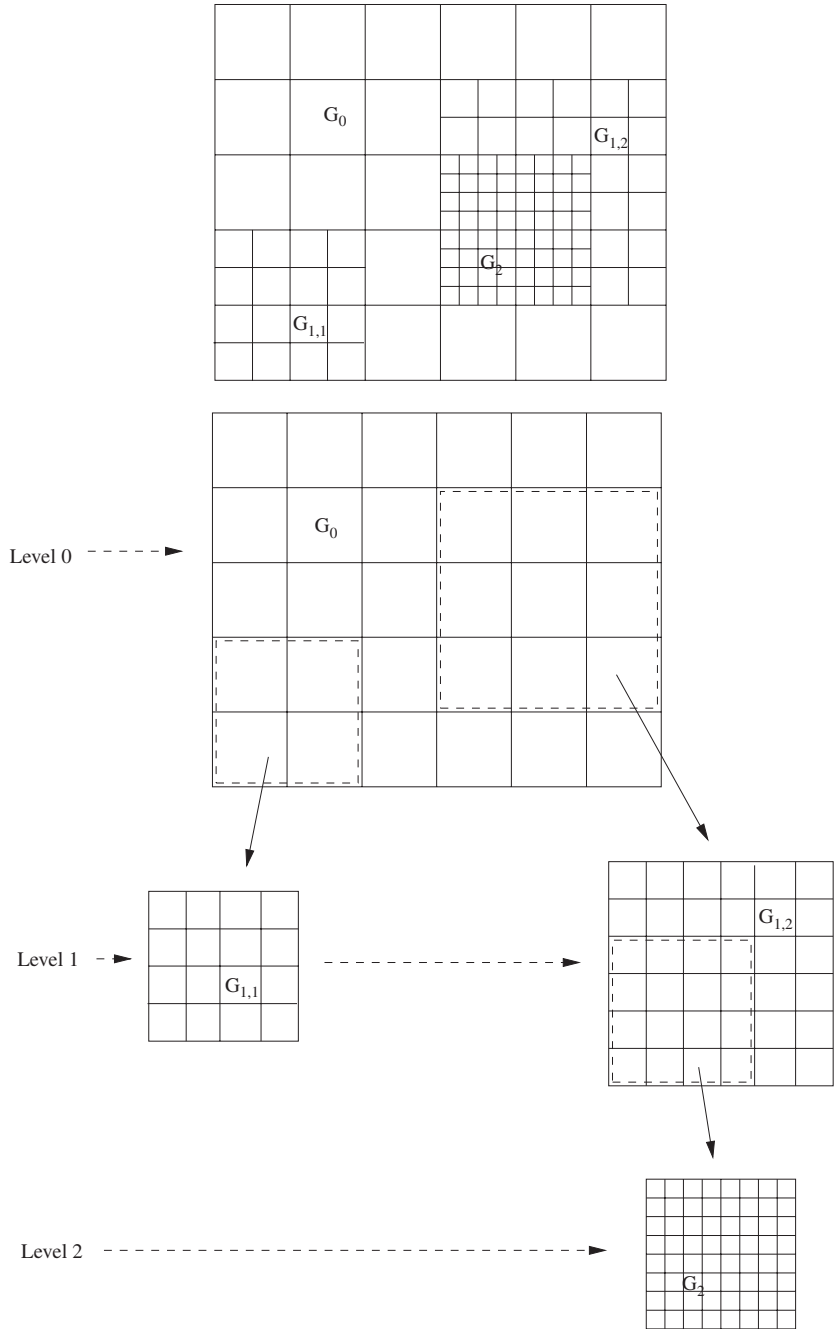

Fig. 1. An example of grid hierarchy: a root grid G_0 has two child grids $G_{1,1}$ and $G_{1,2}$; at a second level of refinement G_2 is a child grid of $G_{1,2}$.

another. In our implementation, a unique set of pointers, saved in a common block, will successively address the memory locations corresponding to the variables of the different grids, in the order prescribed by the integration algorithm.

This is illustrated in Fig. 3. Let u and v be two distinct variables of the original numerical model, defined in a common block. In the adaptive version of the model, u and v are declared as pointers, and each grid of the hierarchy is represented by a derived type variable containing the local values of u and v on this grid. Then, each time a grid has to be integrated, the pointers u and v are linked to the corresponding local variables of the grid.

```
Recursive Procedure INTEGRATE(l)
  If l == 0 Then nbstep = 1
  Else nbstep = refinement-ratio
  Endif
  Repeat nbstep times
      Do one time step Δt_l on all grids at level l
      If level l+1 exists Then
          Compute boundary conditions at level l+1
          INTEGRATE(l+1)
          update level l
      Endif
  End Repeat
  End Procedure INTEGRATE
```

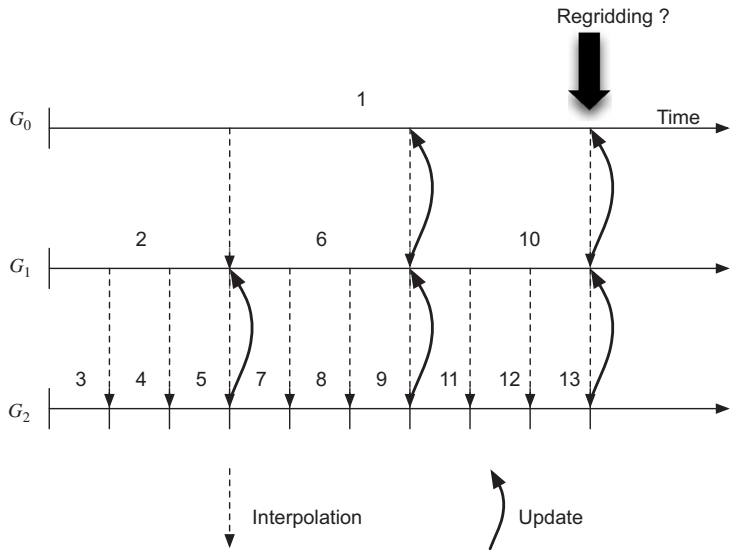

Fig. 2. Integration algorithm (top) and an example (bottom) with a time refinement factor of 3 and three grids G_0, G_1 and G_2. Numbers on chart refer to integration order.

3.2. Overview of the process

To introduce BOSAMR within an existing numerical model, a number of tasks must be carried out. The AGRIF software is composed of two parts. In addition to the model-independent parts of the algorithm (time integration and clustering algorithms, interpolation and update schemes...), the new code must deal with dynamic memory allocation, the instantiation of the previously described pointers, etc. This is the model-dependent part since it depends on the variables of the original model.

The model-independent parts are implemented in a library of Fortran90 modules. AGRIF also includes an external program (a converter written in the C and Yacc/Lex languages) that analyzes a user's written configuration file and produces the model-dependent part of the BOSAMR. The information provided by the user in this configuration file concerns global BOSAMR parameters (mesh refinement factors, maximum number of levels of the grid hierarchy, regridding interval, ...), grid variables, and grid operations (interpolations/updates) on these variables.

Specific keywords exist for each of these operations and for each keyword that refers to an interpolation or update operation, user callable Fortran routines will be produced by the converter. For full information, the reader is referred to the user's manual.

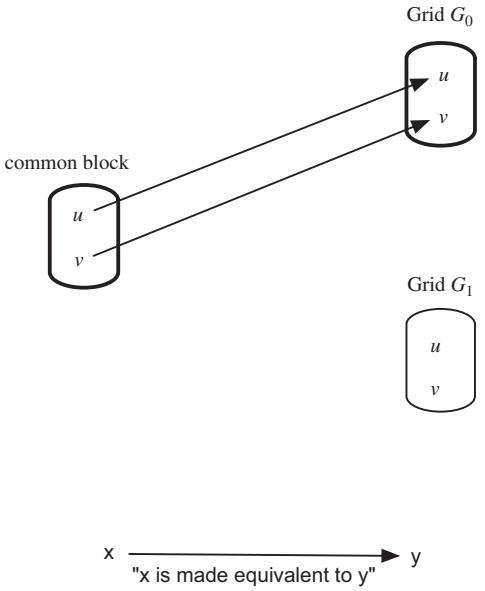

Fig. 3. Pointers instantiation: instantiation process is represented by arrows, once it is completed, accessing u and v in common block is equivalent to accessing corresponding variables on grid G_0.

The overall process of compilation is depicted in Fig. 4. In the original model, the main modification is to add a call to the AGRIF procedure named Agrif_Step which calls the time integration algorithm of the grid hierarchy and proceeds to the regridding step at regular intervals.

3.3. Main features of AGRIF

The AGRIF software is distributed under the GNU GPL licence and includes a number of examples to illustrate its main features. These are:

- 1D, 2D, 3D (adaptive) mesh refinement;
- treatment of staggered grids;
- treatment of masked fields;
- use of grids with predefined fixed location; and
- MPI parallelization.

More complex examples in the context of ocean modelling can be found in Blayo and Debreu (1999) and Debreu et al. (2005).

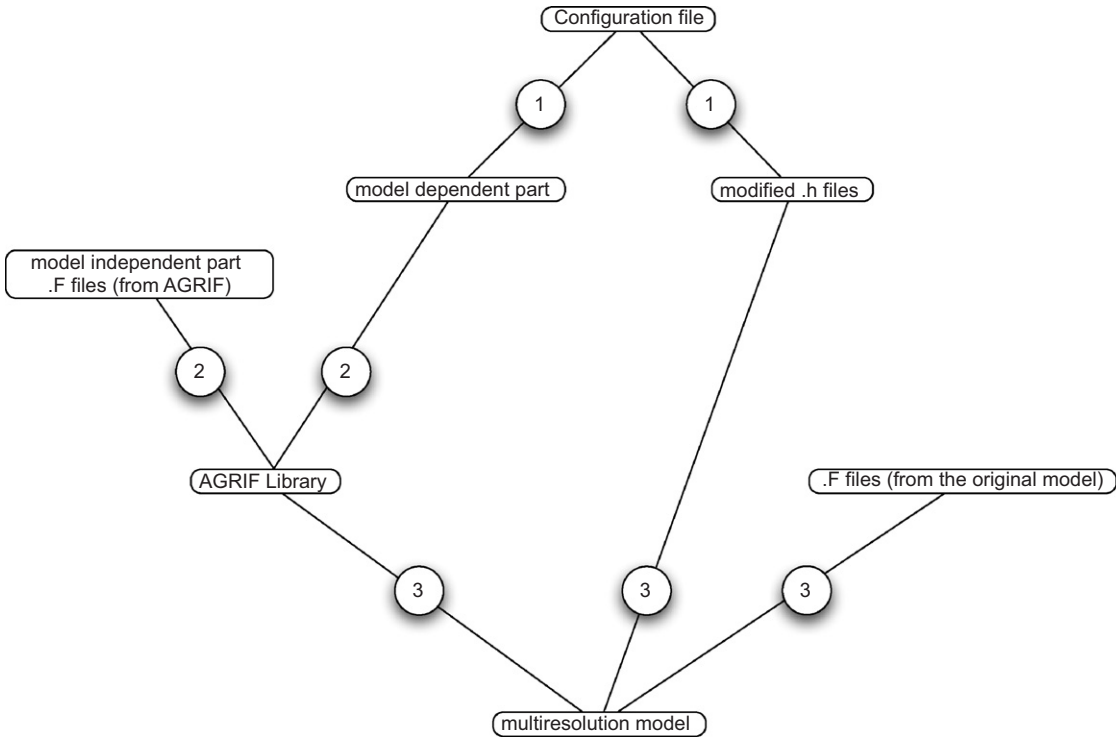

Fig. 4. Overview of compilation process: (1) a converter creates model-dependent part (dynamic allocation of model variables, pointers instantiation) and modifies original declaration files (changing statically declared variables to pointers); (2) model-dependent and model-independent parts are compiled to produce a library; (3) this library is used in compilation of original Fortran files and modified declaration files from step 1 to produce a multiresolution code. In this multiresolution code, interpolation/update procedures generated through step 1, according to configuration file, can be used.

Acknowledgments

We are grateful for financial support from the French Navy under Contract EPSHOM 30/97. IDOPT is a joint CNRS/University of Grenoble/ Institut National Polytechnique de Grenoble/ INRIA project. The authors are also grateful for the comments of a reviewer which lead to substantial improvements in the paper.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at doi:10.1016/j.cageo.2007.01.009.

References

Berger, M., Colella, P., 1989. Local adaptive mesh refinement for shock hydrodynamics. Journal of Computational Physics 82, 64–84.

Berger, M., Oliger, J., 1984. Adaptive mesh refinement for hyperbolic partial differential equations. Journal of Computational Physics 53, 484–512.

Berger, M., Rigoutsos, I., 1992. An algorithm for point clustering and grid generation. IEEE Transactions on Systems, Man and Cybernetics 21, 1278–1286.

Blayo, E., Debreu, L., 1999. Adaptive mesh refinement for finite difference ocean models: first experiments. Journal of Physical Oceanography 29, 1239–1250.

Debreu, L., Blayo, E., Barnier, B., 2005. A general adaptive multiresolution approach to ocean modelling: experiments in a primitive equation of the north Atlantic. In: Plewa, T., Linde, T., Weirs, V.G. (Eds.), Adaptive Mesh Refinement—Theory and Applications. Lecture Notes in Computational Science and Engineering, vol. 41. Springer, Berlin, pp. 303–314.