CAMPFIRE

CAMPFIRE: ADAPTIVE, MULTILEVEL, PARALLEL FULLY IMPLICIT RESEARCH ENGINE

Christopher Goodyer
School of Computing
University of Leeds
C.E.Goodyer@leeds.ac.uk

Contents

1 Introduction

Campfire is a tool to facilitate coupling non-linear multigrid, parallel execution on a properly load balanced adaptively refined grid, using the open source library PARAMESH [1, 2, 3]. The intention is to provide a user-friendly interface layer where only the application-specific routines are exposed to the user by default. Obviously the source to the library is also provided and it is envisaged that there may be future scenarios where applications may need to extend the functionality of algorithm.

PARAMESH, as used here, is an adaptive mesh refinement code which works on individual square/cubic blocks of data. These data blocks therefore can be spread across processors allowing parallelism. This is accomplished using MPI. Since each block is independent in memory from its neighbours they all have a layer of *guard cells* around the outside which can be filled with the appropriate neighbouring values. PARAMESH includes its adaptivity through quad-tree/oct-tree uniform refinement of blocks.

By default PARAMESH includes a linear multigrid solver in its code-base, although this not used by default. Campfire extends this idea to be an FAS multigrid in order to solve non-linear equations. The mesh refinement functionality in PARAMESH allows the multigrid implementation to therefore be an adaptive mesh multigrid, sometimes coined MLAT.

The use of multigrid in Campfire inherently needs a smoother to be applied at each level. A Jacobi pointwise finite difference smoother is used, with an approximate solve performed on the coarsest grid level.

This document is structured as follows. In Section 2 more details of the genesis of this software are given, along with information on how it operates. The changes from the default PARAMESH behaviour are also summarised.

Instructions on how to obtain, and compile the code is given in 3 with the additional information for running cases given in Section 4.

Information for developers follows. In Section 5 guidelines on extending existing models is given, and Section 6 explains how to write a new application.

Contained in Campfire are a collection of tools intended to aid visualization of outputs. These are described in Section 7. There are also a growing number of documents included with the software in the `DOCS` directory and the suggestions for how to add what to these are given in Section 8.

Finally the document concludes with a full description of the API (Section 9) and kknown bugs and features needed to be added (Section 10).

2 History and Implementation

Following the development of a 2-d fully implicit adaptive multigrid code for phase field problems [5, 6, 7], now released as open source software from [8] attention turned to the more interesting, but considerably more computationally challenging 3-d problems. The need for multigrid, adaptive meshing and implicit timestepping had been clearly demonstrated, but in order to get to mesh resolutions that were sufficiently fine to be realistic,

but sufficiently large domains to be useful then a move to a parallel implementation was imperative. As such, PARAMESH was chsen as a suitable software library providing a necessary framework in which to build these applications.

The PARAMESH software used in this work was developed at the NASA Goddard Space Flight Center and Drexel University under NASA's HPCC and ESTO/CT projects and under grant NNG04GP79G from the NASA/AISR project.

The original development of what is today the Campfire software was done by Jan Rosam and James Green. This resulted in a software basis that forms much of the core code present today in Campfire. In particular many of the components of both the MG-SRC and PF-SRC directories come from their work. Chris Goodyer took their software, corrected many issues and has repackaged it as given here. The major extensions are as described in the following paragraphs.

This idea of adaptive meshes for multigrid is one of the most important areas where the original PARAMESH library has had to be modified. For efficient parallelism there needs to be equal work distribution per processors on every grid. Unfortunately the default PARAMESH implementation has a load balancing of all the blocks in the simulation, irregardless of which grid is being worked on. A new algorithm was devised, throwing away the Morton ordering that PARAMESH has at its heart, and ordering the blocks per processor based on which grid they are on. In order to load balance the first and last blocks on that processor are shuffled left or right if a processor has too many blocks.

Whilst apative meshing is included in the default mesh it requires that the domain remains fixed throughout the computation. For most simulations this is entirely sensible. However for the phase field applictions that this software suite was designed for this is not quite so useful. The simulation is about growing a dendrite from a small seed. Fixing the domain means that the total mesh requirements of the final solution need to be known before starting the simulation. In order to combat this requirement a mesh growing strategy has been introduced. The basic idea is that as the solution grows towards a far-field boundary condition extra blocks are added on the coarsest level to allow the solution to keep growing. Advantages of this approach are that the simulation can be started on a very small domain and be allowed to adapt as necessary, and that this adaptation need no longer be constrained to square domains. The introduction of non-convex corners meant that further changes were necessary in the PARAMESH library, as did the loading of checkpoint files with different domain sizes than the initial domain specified in the setup procedures.

Scalability and the pf_ routines

3 Installation and Compilation

svn checkouts of paramesh and Campfire

download of appropriate hdf5

environment variables

```
setenv HDF5_DIR /usr/not-backed-up/PHASEFIELD/hdf5-1.6.10
setenv PARAMESH_DIR /usr/not-backed-up/PHASEFIELD/paramesh
```

4 Running cases

modifications – application case and generic parameters
- make
- amr_runtime_parameters:
- block counts
- ndim
- nxb
- nvar
- nguard
- output_dir
- mpirun

5 Development of existing applications

Hopefully all changes should be incremental
- Adding new variables will mean increasing nvar_work as well as nvar
- Uploading tested changes back to svn.

6 Producing a new application

Assumptions made for PARAMESH in Campfire, e.g. not spherical, only cell centred
- Assumptions made in Campfire, e.g. BCs
- Copy SKELETON-SRC directory
- Rename source files appropriately and modify Makefile
- Expand all the needed stub functions
- Use a user module to pass parameters
- Uploading changes back to svn.

7 Visualization tools

Output provided at two frequencies by Campfire: every timestep and every output_frequency timesteps
- Text output controlled using `verbosity` variable in `generic_parameters`. Use it.

7.1 Using ViSIT with Chombo output

HDF5 output.

7.2 Extractor

Generates isosurfaces, axial-plane profiles,

7.3 Tip velocity and averaging using RealStripper

7.4 GraphViz

8 Documentation

9 API

The following functions are all called from within Campfire and need to be provided in the user code. Many can just be stubs if not needed. Obviously a domain needs initialisation and the user needs to provide an appropriate smoother and defect calculation (which will be very similar).

Initialisation:
`app_initial_soln_blk` : (`lb`)
`app_parameter_set` : ( )
`app_domain_setup` : (`pid`, `noprocs`)
Multigrid smoother and defect calculation:
`app_mg_smooth` : (`pid`, `level`)
`app_mg_get_defect_var` : (`lb`, `max_defect`, `rms`, `npts`)
Input/output:
`app_read_checkpoint` : (`pid`, `noprocs`)
`app_pretimeloop_output` : (`pid`, `noprocs`)
`app_close_files` : (`pid`, `noprocs`)
`app_test_refinement` : ( )
`app_output_perstep` : (`pid`, `noprocs`, `stepno`)
`app_output_occasional` : (`pid`, `noprocs`, `stepno`)

10 Known Bugs and To Do list

- If mesh growing code is used then in parallel it seems to struggle finding coarsest neighbour initially if only a single block is used on the coarsest level.
- Calls to `pf_prolong` and `pf_restrict` do not work correctly, so stick with `amr_prolong` and `amr_restrict` for now.

References

[1] Peter MacNeice, Kevin M. Olson, Clark Mobarry, Rosalinda deFainchtein and Charles Packer, "PARAMESH : A parallel adaptive mesh refinement community toolkit.", Computer Physics Communications, vol. 126, p.330-354, (2000).

[2] Kevin Olson and Peter MacNeice, 2005, "An Over of the PARAMESH AMR Software and Some of Its Applications",in Adaptive Mesh Refinement-Theory and Applications, Proceedings of the Chicago Workshop on Adaptive Mesh Refinement

Methods, Series: Lecture Notes in Computational Science and Engineering, vol. 41, eds. T. Plewa, T. Linde, and G. Weirs (Berlin: Springer).

[3] Kevin Olson, 2006, "PARAMESH: A Parallel Adaptive Grid Tool", in Parallel Computational Fluid Dynamics 2005: Theory and Applications: Proceedings of the Parallel CFD Conference, College Park, MD, U.S.A., eds. A. Deane, A. Ecer, G. Brenner, D. Emerson, J. McDonough, J. Periaux, N. Satofuka, and D. Tromeur-Dervout (Elsevier).

[4] PARAMESH source code : `http://sourceforge.net/projects/paramesh/`

[5] Mullis A M, Rosam, J and Jimack, P K, Solute Trapping and the effects of Anti-Trapping Currents on Phase-Field Models of Coupled Thermo-Solutal Solidification J. Cryst. Growth, vol.312, pp.1891–1897, 2010.

[6] Rosam, J, Jimack, P K and Mullis, A M, Quantitative Phase-Field Modelling of Solidification at High Lewis Number . Physical Review E, vol.79, article 030601(R), 2009.

[7] Rosam, J, Jimack, P K and Mullis, A M, An Adaptive, Fully Implicit, Multigrid Phase-Field Model for the Quantitative Simulation of Non-isothermal Binary Alloy Solidification. Acta Materialia, vol.56, pp.4559-4569, 2008.

[8] PhAIM-2d : `http://www.digital.leeds.ac.uk/software`

[9] Green, J, Jimack, P K, Mullis A M and Rosam, J, An Adaptive, Multilevel Scheme for the Implicit Solution of Three-Dimensional Phase-Field Equations Numerical Methods for PDEs, vol.27, pp.106–120, 2011.

[10] Green, J R, Jimack, P K, Mullis, A M and Rosam, J, Towards a Three-Dimensional Parallel, Adaptive, Multilevel Solver for the Solution of Nonlinear, Time-Dependent, Phase-Change Problems. In Parallel, Distributed and Grid Computing for Engineering, eds. B.H.V. Topping and P. Ivanyi (Saxe-Coburg Publications, UK), pp.251–274, 2009.