Multifluid Software Testing Plan

Applied Numerical Algorithms Group
NERSC Division
Lawrence Berkeley National Laboratory
Berkeley, CA

July 8, 2003

Contents

Chapter 1

Scope

The multifluid code will build heavily on the MFChombo [1] infrastructure, which in turn will rely on the EBChombo [3] and Chombo [2] software. The software test plan outlined in this document will focus on the functionality developed to implement the algorithm outlined in the "Multifluid Algorithm Specification" [5] document; since MFChombo, EBChombo and Chombo have their own software test plans, it is not necessary to provide for testing the functionality of the libraries themselves. Note, however, that since the software developed for the multifluid code will use the MFChombo functionality so extensively, changes and bugs in the libraries will tend to have effects on the testing results. Multifluid code developers are kept abreast of developments in the libraries through CVS notification (which sends e-mail whenever a change is made in the CVS version-control repositories), and through the ChomboUsers e-mail list.

1.1 System Overview

The multifluid software will implement an algorithm for computing fluid dynamics for multiple inmiscible fluids with surface tension [5]. This software will be built using the MFChombo software, which implements basic support for computations in a multifluid environment using an embedded interface description of the multifluid interface. The MFChombo software is built upon the Chombo and EBChombo software libraries.

Chapter 2

Reference Documents

We will refer to the multifluid algorithm described in [5] and the MFChombo software design document [1]. Also, the test plan for the AMRINS code [4] is used as a starting point for this test plan.

Chapter 3

Software Test Environment

The multifluid software, linked to the MFChombo and Chombo software libraries will be tested. As new functionality is added and functionality is improved, testing will continue. It is expected that a given time, the multifluid code will be in sync with the current state of the MFChombo and Chombo libraries.

This software is primarily intended for use on UNIX/Linux-based systems. In general, the makefiles used in both Chombo and AMRINS require GNU make (gmake). The software itself is designed to be run from a shell, with an inputs file providing run-specific inputs. For data output, the software uses hdf5, so the system must have hdf5-1.4.1 installed. The Chombo and AMRINS software is written in C++ and Fortran77, so working C++ and F77 compilers must be available. We generally use the GNU compiler: both gcc 2.95 and 3.1 have been successfully used to compile this code. In addition, the Chombo Fortran preprocessor uses PERL. If ChomboVis will be used to examine results, then it must be installed as well. ChomboVis additionally requires Python and VTK.

We will test the multifluid code in a variety of environments, with a variety of compilers. Table 3 lists the platforms and compilers we have successfully compiled and run other Chombo codes:

Testing is done by ANAG personnel, although collaborators have been useful for finding unintended functionality, primarily in the Chombo libraries themselves.

Platform	OS	C++ Compiler	Fortran Compiler
CRAY T3E	unicos	KCC 3.3d	Cray Fortran 3.5.0.4
IBM SP	AIX	KCC 4.0f, xlC 5.0.2.0	IBM XL Fortran 7.1.1.0
Pentium/AMD	Linux	gcc 2.95.3+,	g77 2.95.3+, PGI Fortran 3.3-2
		Intel C++ 6.0	Intel Fortran 5.0.1
Compaq	OSF	gcc 3.1	Compaq f77 X5.4A-1684-46B5P
Compaq	Linux	gcc 2.95.3	g77 2.95.3
SGI	IRIX	MIPS Pro CC 7.3.1.2m, gcc 2.95.3	MIPS Pro f90 7.3.1.2m

Table 3.1: Platforms and compilers on which the Chombo codes have been tested

Chapter 4

Test Identification

4.1 General Information

The testing for the multifluid libraries will mostly be a set of simple unit tests to verify functionality.

For much of the functionality which will be developed for this work, the best way to test whether components are functioning properly is often to do a convergence study. For example, in the case of an operator such as a gradient or a Laplacian, a field is initialized on a series of meshes, each a factor of 2 finer than the last. The operator is applied to the field. If the operator is properly implemented, the result should converge at second-order rates to an analytic solution.

4.1.1 Test Level

In general, most of the testing outlined in this document will be component testing. System-level testing will also be carried out on the entire multifluid codes once they have been fully developed. It is expected that integration testing is not necessary at this time, because of the small size of the design team.

4.1.2 Test Classes

In general, testing will be structured to evaluate correctness of the code. It is anticipated that since the future phases of code development will be focused on performance enhancement, performance of the code will then be monitored closely, so routine performance testing should be unnecessary, while testing for correctness will be important as changes are made to speed up the code.

4.2 Planned Tests

In this section, we outline the tests planned for the multifluid software, broken down by functional algorithm component. Many of the tests will be based on the similar unit tests developed for the `AMRINS` software [4], with extensions to test functionality unique to the `multifluid` case. All testing codes will be written in C++.

4.2.1 Test 1 – Linear Operators

The multifluid divergence, gradient, and Laplacian operators developed for the multifluid projections will be tested by doing convergence tests. The gradient and Laplacian operators will be tested by defining a field on a set of progressively finer grids, applying the operator to the field, and then ensuring that the result converges to the analytic result as expected. For the divergence operators, a similar procedure is followed, except with a vector field instead of a single field variable.

4.2.2 Test 2 – Convergence of Multifluid Projection Operators

Both face-centered and cell-centered projection operators will be implemented for the multifluid algorithm. Tests for each of these operators are similar.

Test 2a – Face-centered Projection Operator Convergence

A face-centered velocity field is initialized on a multifluid domain, the face-centered projection is applied, and convergence of the result is checked.

Test 2b – Cell-centered projection operator convergence

A cell-centered velocity field is initialized on a multifluid domain, the cell-centered projection is applied, and convergence of the result is checked.

4.2.3 Test 3 – Fixed-boundary diffusion solver

The heat equation with a simple moving interface (initially a plane, then one with curvature) will be solved on a set of test grids using the fixed-boundary approximation to ensure that the basic algorithm and implementation of the fixed-boundary approach is correct and converges appropriately.

4.2.4 Test 4 – Upwind advection

A bubble will be advected in a simple flow field (first unidirectional flow without shear, then unidirectional flow with shear, then a solid-body rotation) to ensure that the upwind

advection scheme has been implemented correctly and demonstrates the proper convergence as the mesh is refined.

4.2.5 Test 5 – Fixed-boundary projection test

A bubble in an incompressible flow field will be advanced to ensure that the projection method for a moving boundary is implemented correctly and demonstrates proper convergence.

4.2.6 Test 6 – Surface Tension Test

In this test, a bubble is initialized to a non-circular (spherical) shape in a viscous fluid with no velocity. Surface tension effects should cause the bubble to evolve into a spherical shape. This will partially test the surface tension model used in this code.

4.2.7 Test 7 – Multifluid system test

A more complicated multifluid test problem will be identified and used to ensure that the proper convergence for the multifluid system test is achieved.

4.2.8 Test 8 – Multifluid AMR test

To test the AMR algorithm, the test problem used in Test 5 will be used to ensure that the AMR incompressible multifluid implementation achieves fine-grid accuracy and convergence rates similar to the non-AMR (single-grid) example.

4.2.9 Test 9 – Multifluid system regression test

The benchmark multifluid test problem (most likely that used in Test 7) will be used as a regression test. Diagnostic variables such as total mass, kinetic energy, and max(divergence) will be reported at the end of the run. Changes in these diagnostic quantities will indicate changes which will need to be investigated.

Chapter 5

Test Schedules

Once a capability in the code has been verified by the appropriate test, we plan to use these tests as regression tests. We plan to apply the entire test suite once each month to ensure that no unintended changes are introduced, and we also will re-run the test suite after bugs are found and corrected to ensure that new bugs are not introduced.

The multifluid system regression test (Test 7) will be done weekly for serial runs, and monthly for the suite of parallel runs, and also after bug fixes and library changes to lessen the possibility of unintended changes in the code.

Also, acceptance tests will be run as stakeholders take possession of the software.

Chapter 6

Bug Tracking

The multifluid code developers (and the `MFChombo` and `Chombo` developers) use the ttpro system for bug tracking. When a bug or unexpected behavior in the code is identified, a description is entered in the ANAG ttpro database. As the bug is investigated and fixed, the description is updated and expanded. Once a bug has been fixed, the bug report is "closed" in ttpro, but it remains in the database for future reference if needed. Also, after a bug fix, the regression test (Test # 7) is re-run to ensure that no unanticipated effects have been added.

Chapter 7

Requirements Traceability

The requirements traceability matrix is presented in Figure 7.1. The first column, "Alg Spec No", connects the entry in matrix with the relevant section of the "Multifluid Algorithm Specification" document [5]. A parenthetical number refers to a specific equation in [5].

Alg Spec No.	Req Statement	S/W module	Test Spec	Test Case #	Verification	Mod. Field
5.2-3	Linear operators	Linear Operator functions	multifluid Test Plan	1	not verified	
5.2 (46)	Face-Centered Projection	MFProjector: MacProject	multifluid test plan	2a	not verified	
5.3 (47)	Cell-centered Projection	MFProjector: CCProject	multifluid Test Plan	2b	not verified	
3.2-3.3	Diffusion solver	MFDiffusion: computeDiffusion	multifluid Test Plan	3	not verified	
6.1	upwind advection	MFUpwindTrace	multifluid Test Plan	4	not verified	
3.4	moving-interface projection	MFProjection CCproject	multifluid Test Plan	5	not verified	
	surface tension		multifluid Test Plan	6	not verified	
	multifluid system test	multifluid code	multifluid Test Plan	7	not verified	
	multifluid system regression test	multifluid code	multifluid Test Plan	7	not verified	

Table 7.1: Requirements Traceability Matrix

Bibliography

[1] P. Colella, D. T. Graves, T. J. Ligocki, D. Martin, D. B. Serafini, and B. Van Straalen. MFChombo Software Package for Cartesian Grid, Multifluid Applications. unpublished, 2003.

[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, D. Modiano, D. B. Serafini, and B. Van Straalen. EBChombo Software Package for Cartesian Grid, Embedded Boundary Applications. unpublished, 2001.

[4] Applied Numerical Algorithms Group. Incompressible Navier-Stokes software testing plan. available at http://davis.lbl.gov/NASA, 2002.

[5] Dan Martin and Phil Colella. Multifluid algorithm specification. available at http://davis.lbl.gov/NASA, 2003.