Orbital Integrator System Manual

Benjamin Sprague

This manual is intended to describe the functionality of the orbital integrator system.

Copyright © 2006 Benjamin Sprague

Permission is granted to copy, distribute, and/or modify this document as long as no claims or damages are made to the author. It is to be accepted as-is, with no liability.

Table of Contents

1 Getting Started

1.1 Building the executables

We designed this orbital integrator to have a very simple scheme for building executables. You can use the default GNU `make` program to build both MPI and non-MPI versions of the integrator, as well as to make the initial conditions generator executable.

To use the makefile scheme that we designed for this orbital integrator, you must first set up the makefile, and then issue '`make`' commands to create the executables.

1.1.1 Makefile setup

The main makefile is located in the root directory of the orbital integrator system, and is named '`Makefile`'. This file must be modified to tell the build system what compiler and linker is to be used for both the MPI and non-MPI build setups. Note that the initial conditions generator is built using the non-MPI settings, so if you want to generate ICs automatically, you must provide non-MPI settings even if you only build the main orbital integrator as an MPI project.

The MPI variables *MPI_FORTC* and *MPI_LINKER* are typically set to the '`mpif90`' script provided with the MPICH distribution. The non-MPI variables, *FORTC* and *LINKER* are set to a Fortran 90 compatable complier. The makefiles are currently designed for the Intel fortran compiler, and may need additional modification for use with other compilers.

1.1.2 Compiling the executables

To build the executables using GNU `make`, first navigate to the root directory of the orbital integrator system. The build options are as follows:

- To make the basic non-MPI integrator, simply type '`make`'.
- To build the MPI-enabled integrator, type '`make mpi`'.
- To create the initial conditions generator, use the command '`make ic_generator`'.
- To erase all of the executables and intermediate files, use the command '`make clean`'

All of these commands place all intermediate build files within the '`build/`' directory in order to keep the source directories as clean as possible. Also, the final executables are placed within the main orbital generator system directory.

Also notice that the MPI and non-MPI integrators share the same executable and most of the same intermediate build files. In most cases, this eliminates the need for a full rebuild of intermediate files in between builds of the non-MPI and MPI version. This significantly speeds up builds when switching between versions, as well as when building the IC generator.

This fortunate situation is the case if the MPICH build command '`mpif90`' is set up to use the same base compiler as the non-MPI compiler in the integrator makefile. Most of the time, this is how an MPICH system is installed. If you are not sure that your system is set up in this way, you may need to remove all of the intermediate files using '`make` `cleanobj`' in between switching from the MPI to the non-MPI builds. Note also that the '`ic_generator`' executable is actually a non-MPI build, and in such a case would need to have the object files cleared in order to build it after an MPI build of the main system.

1.2 Running the integrator

The first step in running the integrator is creating an input file. See the input files in the '`examples/input/base`' directory for an example of how the input files are formatted. They are read using a FORTRAN NAMELIST scheme, so they must be formatted in a similar fashion to the example input files.

Step two is to generate the initial conditions. To create the initial conditions, simply run the command '`./IC_generator -i <input file(s)>`'. This command will run the initial conditions generator and place the file containing the initial conditions in the location specified by the input file. Notice that you may specify multiple input files by listing them separately, or by using standard unix wildcard characters. The initial conditions generator will simply generate each set of initial conditions in sequence, and exit when it has completed all initial conditions files specified.

The final step is to execute the orbital integrator program. To execute the program on a single processor, use the command '`./orbit_integrator -i <input file(s)>`'. The output from the program will be placed in the location specified in the input file.

For an MPI execution, use the command '`mpiexec -n <# processors> ./orbit_integrator -i <input file(s)>`'. This will cause the integrator to use as many processors as requested for each input file, greatly improving the execution time. Note that during execution, each thread of the program uses a temporary file with the suffix '`.threadXXXX`', which are then combined into the final output file when the program is completed. This method requires that the output files be stored in a shared storage location that is visible to each thread of execution, so that the files can be collected at the end of the simulation run.

2 Structure Overview

3 Code Organization

Index

(Index is nonexistent)