HD3D

HD3D is a pseudospectral three-dimensional periodic hydrodynamic/magnetohydrodynamic/ Hall-MHD turbulence model. HD3D numerically solves the incompressible Navier-Stokes equations in 3 dimensions with periodic boundary conditions. A pseudo-spectral method is used to compute spatial derivatives, while adjustable order Runge-Kutta method is used to evolve the system in the time domain. This benchmark does a free decaying simulation of Taylor-Green vortices. Building HD3D requires a fortran90 compiler, MPI libraries, and the FFTW libraries (versions 2.1.5 or earlier). FFTW 2.1.5 is included in this distribution of HD3D.

The HD3D benchmark can be downloaded from http://web.ncar.teragrid.org/~bmayer/benchmarks/hd3D_HT_patched.tar.gz.

1 Procedure

Before building HD3D the FFTW libraries must be built and installed. FFTW 2.1.5 is included in the **hd3D/fftw-2.1.5** directory. To build the FFTW libraries, **cd** into the **fftw-2.1.5** directory and follow these steps:

1. Set environment variables to control compilation options as needed. Some frequently used environment variables are:

CC C compiler

CFLAGS flags to pass to C compiler

F77 Fortran compiler

FFLAGS flags to pass to Fortran compiler

LDFLAGS flags to pass to linker

ARFLAGS flags to pass to ar

1. Type **./configure --enable-type-prefix --enable-float ---**prefix=*PATH;* where *PATH* is a user-specified installation directory (as noted below).
2. Type **make**
3. Type **make install**

By default the libraries are installed in **/usr/local/lib**, **/usr/local/man**, etc. You can specify an installation prefix other than **/usr/local** (e.g. the user's home directory) by giving **configure** the option **-prefix=***PATH*.

After FFTW has been installed follow these steps to build the **hd3D** executable.

1. Change to the **hd3D/source_mpi** subdirectory.
2. Edit the file **pseudospec3D_mod.f90**, if necessary, to set the model resolution. The following line sets the number of grid points in each direction, and the value of **n** should be set to 256 for the standard benchmark case.

INTEGER :: n = 256

1. Edit the **fftp_mod.f90** file and set the lines:

INTEGER, PARAMETER :: ikind = 8

INTEGER, PARAMETER :: csize = 8

The parameter **ikind** should be set to the size of a C pointer (4 for 32-bit architectures or 8 for 64-bit architectures). The parameter **csize** is used by the parallel FFT to do cache-friendly transpositions. The optimal value for a particular machine can be determined through benchmarking, but rule of thumb values are 8 if the L1 cache is smaller or equal to 64 kB, and 16 if the L1 cache is larger than 64 kB.

1. Edit the **Makefile** to set compiler options and library paths. The **Makefile** contains a number of example sets of compiler options for various architectures. The **Makefile** also contains an option to help resolve underscoring incompatibilities between code compiled with different compilers. If the variable 'UNDER' is set to 1 in the makefile, the makefile runs a PERL script to add underscores as needed before compiling. The file **external** lists the names of external functions and subroutines that need an extra underscore.

WARNING: if you use 'UNDER=1' and compilation fails, remember to do **make dist** before compiling again or changing the source files. Failing to do so will result in extra underscores being added to the source files every time a **make** is attempted.

1. Type **make hd3D** to build the **hd3D** executable.

10. Input files for two run types are included, a consistency run and a benchmark run. After the executable is built the consistency run should be performed to validate the correctness of the port. For more information on performing a consistency run see the validation section below.

11. After the consistency run has been successfully completed the benchmark runs can be performed. To perform the benchmark runs copy the **hd3D** executable into the **benchmark_run** subdirectory and launch the benchmark from this location. The **hd3D** executable will read the input files **abc.txt**, **para meter.txt**, and **status.txt** from this directory, and then perform a benchmark run. The benchmark should be launched using the **mpirun** command (**mpirun -np 16 ./hd3d**) and run on one to the maximum number of system processors. Each time the benchmark is run a line containing the number of processors used and the average time required to complete a single time step is appended to the file **benchmark.txt**.

2 Validation

To perform a consistency run copy the **hd3D** executable into the **consistency_run** subdirectory and run the executable from this location. The code will read the input files **abc.txt**, **parameter.txt**, and **status.txt** from this directory and then perform the consistency run. The consistency run will write a large number of binary files to the **binfiles** subdirectory and will write three text files (**balance.txt**, **kspectrum.txt**, **helicity.txt**) into the **consistency_run** directory. A Perl script named **validate.pl** is included in the **consistency_run** directory. The **validate.pl** script reads the **balance.txt** file produced by a consistency run and determines if the results are numerically valid. An example of the output from **validate.pl** on a valid **balance.txt** file are shown below.

```
% ./validate.pl

N: 100

Results are valid.      [ pass ]

Max diff: 3.98762853921365e-05
```

The **validate.pl** output should indicate that the results are valid; the value of "Max diff" need not match the above. Note that each time a consistency run is performed if a **balance.txt** file already exists then new values are appended to the end of the file, leaving the old values intact. For this reason it is important to remove the **balance.txt**, **helicity.txt** and **kspectrum.001.txt** files before each new consistency run.

3 Data

A consistency run should be completed on any number of processors greater than or equal to 8. The benchmark case should be run on one to the maximum number of system processors. The file **benchmark.txt** containing the timings should be returned.

Results should be recorded in the benchmark spreadsheet.