User Guide Part 7: Easily-Made Mistakes / Troubleshooting

Author: John Hayes

"I drank *what*?" -- Socrates
"Doh!!" -- Homer Simpson

Herein I offer a list of mistakes, all of which I have made (many of them repeatedly), and some of which you will undoubtedly make sooner or later. For example:

Conflicting Processor Number Specification

The product ntiles(1)*ntiles(2)*ntiles(3), taken from the specified values in the MPITOP namelist in zmp_inp, must equal the number of processors you request in your job submission script or command line (poe, mpirun, etc.).

Conflicts Between Array Sizes and MPI Topology

Referring to the namelists ARRAYCONF, MPITOP, and GGEN(1-3), in zmp_inp, remember that:

ntiles(1)*I_ZONES must be greater than or equal to NBL in GGEN1

ntiles(2)*J_ZONES must be greater than or equal to NBL in GGEN2

ntiles(3)*K_ZONES must be greater than or equal to NBL in GGEN3

An Insufficient Number of Zones Along a Particular Axis

Because of the strategy used to overlap communication and computation in a multi-D/multi-process calculation, there are required minimum numbers of zones **PER PROCESSOR** along each axis made active by the dimensionality of the problem. In 1D, this minimum number will surely not be an issue. In 2 or 3 dimensions, however, we have that:

2D:

- I_ZONES must be at least 5
- J_ZONES must be at least 8

3D:

- I_ZONES must be at least 5

- J_ZONES must be at least 8
- K_ZONES must be at least 5

Periodic Boundary Condition Conflicts

The MPITOP namelist in zmp_inp contains a 3-element LOGICAL array called PERIODIC. The values of these elements (".true." or ".false.") tell MPI whether periodic boundary conditions are used along any of the domain axes. These values must be consistent with the integer values specified in the hydro BC namelists (IIB, OIB, IJB, etc.).

An Inconsistent Value of LDIMEN

The integer LDIMEN in zmp_inp namelist GEOMCONF tells ZEUS-MP whether a 1, 2, or 3-dimensional problem is being selected. If you select LDIMEN=2, then the value of NBL in GGEN3 had better equal 1.

Inconsistent Gravity Control Switches

Some rules for gravity:

1. XGRAV must equal .TRUE. for any other gravity switch to be active.
2. XSPHGRV can NOT be used if LDIMEN = 3.
3. XSPHGRV MUST be used (assuming XGRAV=.TRUE.) if LDIMEN = 1.

I/O Control

The final namelist in zmp_inp, IOCON, contains time counter integers used to determine how often various types of output files are dumped. When a run is stopped and resumed from a restart file, it is critical that -- using HDF files as an example -- that thdf be reset so that it is equal to or greater than the starting "time" value of the new run.

Some Trouble-Shooting Tips for Naive Users

ALERT: "-O3 -qhot" on an IBM compiler may break the MHD algorithm. I have know idea why, and this may be machine dependent. In any case, be very wary of the "qhot" option.

The example problems documented on this page should work "out of the box" without any fuss, particularly if you attempt them on an IBM Power3 or Power4 architecture (but note the red sentence immediately above). Grizzled code warriors can ignore this section, but users who are relatively new to computational work (welcome to Hell, incidently) may benefit from the following strategy:

1. Select the "blast" test problem (-DPROBLEM=blast in ZMP_CPP line in the Makefile).
2. Turn off ALL optimization in the compiler options in the ZMP_OPTS compiler environment variable.
3. Again in ZMP_OPTS, turn on EVERY exception trap offered by the compiler on your computer (i.e. uninitialized variables, floating exceptions, array bounds checking...)

4. Run this problem as "hydro-only" -- i.e. set XMHD=.FALSE. in zmp_inp. Your goal is to run the most trivial problem ever conceived by Man.
5. Run the problem in 3D cartesian geometry on ONE processor, at 32**3 resolution. Even with debugging traps activated, it won't take that long.
6. If you can run the problem on one processor, try running it on two processors, decomposing along the 1 (X) axis. Then run it on four, decomposing across X and Y. If you made it this far, decompose across all 3 axes. Weird MPI problems can often be ferreted out this way.

Experience has shown that you can sniff out a lot of mistakes with this strategy, including ones that you didn't make (like building asinine precision conventions into the fortran compiler).