Imperial College
London

What to do if your simulation breaks?

Fluidity training workshop

Applied Modelling and Computation Group

Alexandros Avdis

Department of Earth Science and Engineering, Imperial College London

11-13 November 2015

An overview to debugging your failing run.

You will encounter failures while running code.

You have already encountered failing software: Crashes!

Keep calm and carry out the following steps:

- Collect information.
- Interpret information: Think of a cause of the error.
- Treat & test.
- Repeat, and remember simulation is an iterative process.

Collect information.

Collect and go through even if you do not understand all of it

The model & OS output is a primary source of information:

- log file (Fluidity options "`-l -v3`")
- error file (including backtrace)
- pbs output (if running on a HPC)
- stat file
- vtus

Treat is as a puzzle/crossword: Go through the information again

Repeat & Remember: Simulation is an iterative process.

Further sources of useful information

You can re-run a simulation in order to obtain more information:

- Re-run simulation with extra diagnostic fields.
- Increase dump-rate.
- Re-run simulation using binary compiled with debugging flags.
- pbs output (if running on a HPC).
- “matrixdump” and “core” files.

“matrixdump” and “core” files are here included for completeness. However, they large files, do not send to individuals/mailing lists. They are very useful when combined with debuggers.

Interpret information

Going through the information will give you the time to think of what may have gone wrong.

Common causes of failing simulations:

- Bad environment variables, missing/out-of-date dependent software or modules
- Bad options set-up:
  - Option errors
  - Faulty Python scripts
  - Under/over/badly constrained initial and/or boundary conditions
  - Numerical instability
  - Partitioning, adaptivity, field projection
- An error in software, a “bug”.
  - When certain of a bug, contact the development team.

Bad environment variables

What to check:

- Check environment variables, in particular PYTHONPATH: type "`echo $PYTHONPATH`" into your terminal window
- Check for absent dynamic libraries:
  - Navigate to the fluidity `bin` directory
  - Issue `$ ldd fluidity` at the prompt, it will list all dynamic libraries used by the binary.
  - There should be no missing libraries.

Bad options set-up

Option errors:

- The error message might contain advice on what to correct. Try again.
- Use the `diff` and `meld` commands to compare against working `.flml` files.
- Regress to another, working set-up

Regress

Back-up your `.flml` file and progressively remove fields:

- Remove adaptivity.
- Remove any turbulence models and prescribe viscosity (as above)
- Remove parameterisations
- Reduce the number of spatial dimensions
- Simplify the geometry
- Smooth BCs with discontinuities in space or time
- If you have a viscosity field, then set that to a value that gives you a Reynolds Number $\simeq 1000$
- Progressively remove prognostic fields

Then, add complexity back in ONE STEP AT A TIME

Numerical instability

The cause of numerical instability can be tricky to pin-point: The symptom described within an error message may not directly relate the underlying cause of failure.

Check the various factors that can cause numerical instability:

- Timestep, Courant Number, and temporal discretisation
- Spatial discretisation and element pair
- Mesh quality, Condition Number
- Mesh resolution, interpolation error, Grid Reynolds Number
- Field stabilisation, upwinding/slope-limiting
- Strong/weak boundary conditions
- Preconditioner, iterative solver, convergence criteria

Monitoring the progress of a simulation

If you are suspicious that your simulation may crash and you want to monitor its progress, the following commands can be useful:

`statplot *.stat` (press "r" to refresh)

```
tail -f fluidity.log-0
grep ''n/o iterations'' fluidity.log-0
grep reason fluidity.log-0
```

For the last three to work, run with a verbosity of two or more, i.e.
`./fluidity -v2 -l my.flml`

A bug

If you believe there might be a bug in the code then your first step should be to contact the development team and submit a bug report.

The following third-party software can then be useful for slowly stepping through the simulation solve to determine the root cause:

- Valgrind (checking for memory issues)
- GDB (a free Open Source de-bugger)
- DDT (a commercial alternative)

Similarly “vtune” can be useful in determining which lines of code are taking up the most time in the solve.

HELP!!!

Consult the documentation:

- Fluidity wiki
- The troubleshooting section of the Fluidity manual.

Fluidity has a very eager and responsive development community. We can be contacted either through the mailing list or through IRC (`# amcg`).

It would be helpful for you to include information about:

- What simulation are you trying to run?
- What equations are you solving?
- What discretisation are you using?
- What similar simulations do you have that *did* work?

Questions?

AMCG:
http://amcg.ese.ic.ac.uk/

Fluidity:
http://fluidityproject.github.io/

Fluidity code on GitHub:
https://github.com/FluidityProject/fluidity
https://github.com/FluidityProject

Fluidity wiki:
https://github.com/FluidityProject/fluidity/wiki