Fortran

Page maintainer: Gijs van den Oord @goord

Disclaimer: In general the Netherlands eScience Center does not recommend using Fortran. However, in some cases it is the only viable option, for instance if a project builds upon existing code written in this language. This section will be restricted to Fortran90, which captures majority of Fortran source code.

The second use case may be extremely performance-critical dense numerical compute workloads, with no existing alternative. In this case it is recommended to keep the Fortran part of the application minimal, using a high-level language like Python for program control flow, IO, and user interface.

Recommended sources of information

- Fortran90 official documentation
- Fortran wiki
- Fortran90 handbook

Compilers

- **gfortran**: the official GNU Fortran compiler and part of the gcc compiler suite.
- **ifort**: the Intel Fortran compiler, widely used in academia and industry because of its superior performance, but unfortunately this is commercial software so not recommended. The same holds for the Portland compiler **pgfortran**

Debuggers and diagnostic tools

There exist many commercial performance profiling tools by Intel and the Portland Group which we shall not discuss here. Most important freely available alternatives are * **gdb**: the GNU debugger, part of the gcc compiler suite. Use the **-g** option to compile with debugging symbols. * **gprof**: the GNU profiler, part of gcc too. Use the **-p** option to compile with profiling enabled. * **valgrind**: to detect memory leaks.