PX913-15 Introduction to Scientific Software Development

22/23

Department
Physics
Level
Taught Postgraduate Level
Module leader
David Quigley
Credit value
15
Assessment
100% coursework
Study location
University of Warwick main campus, Coventry

Description

Introductory description

N/A.

Module web page

Module aims

To equip students with a solid knowledge of the tools and techniques used in modern scientific software development and the associated tools used to work with the data produced.

Outline syllabus

This is an indicative module outline only to give an indication of the sort of topics that may be covered. Actual sessions held may differ.

Real computers (1 Lecture + 1 x 2hr workshop) - Describe the parts of a computer (memory, CPU disk etc). Introduction to representation of numbers in a computer. High and low-level languages, their purposes and suitability. Intro to the Linux command line, compiling existing code, compiling vs linking, common command line flags to compilers.

Version control (1 Lecture + 1 x 2hr workshops) - Practical use of git for version control, in

particular add, commit, log, push and pull. Collaborative working using git: merging and conflicts.

Programming in C or Fortran (4 lectures + 4 x 2hr workshops) - Boilerplate and support code. Recap of loops, conditionals and functions. Arrays and pointers (in C). Arrays and array intrinsics (in Fortran). Use of and linking of library code.

Codes in the real world (1 lecture + 1 x 2hr workshop) - Output files, code control, testing and verification principles.

Debugging and profiling (1 lecture + 1 x 2hr workshop) - Use of standard debugging tools, approaches to finding bugs. Brief introduction to profiling and optimisation strategies.

Introduction to miniproject (1 lecture) - Description of the projects, how to work with others in software development, expectations and requirements.

Consolidation (1 x 2hr workshop) - When starting with the miniproject one final workshop to help people with any questions they have about any of the tools or techniques that they will need.

Python data visualisation (1 lecture + 1 x 2hr workshop) - Introduction to python for loading data from file and plotting data in Matplotlib, and Mayavi or similar.

Learning outcomes

By the end of the module, students should be able to:

- Understand the basic internals of a computer.
- Be able to use version control for reproducible research.
- Be able to write numerical codes in Fortran or C.
- Understand the common parts of computer code (loops, conditionals, pointers, arrays etc.).
- Be able to compile code and manage libraries.
- Be able to understand the operation of, and make simple modifications to Fortran or C code.
- Acquire basic experience with symbolic debuggers.
- To work alone and in small groups to produce a piece of working code using best principles.
- To have used common tools for data analysis and visualization in Python.

Indicative reading list

Introduction to programming with Fortran : with coverage of Fortran 90, 95, 2003, 2008 and 77, Ian D. Chivers, Jane Sleightholme

The C programming language Kernighan, Brian W.; Ritchie, Dennis M

Introducing Unix and Linux, Joy, Mike; Jarvis, Stephen; Luck, Michael

https://www.cprogramming.com/ website

Python data visualization cookbook, Igor Milovanović

Subject specific skills

To equip students with a solid knowledge of the tools and techniques used in modern scientific

software development and the associated tools used to work with the data produced

understand the basic internals of a computer

be able to use version control for reproducible research

be able to write numerical codes in Fortran or C

understand the common parts of computer code (loops, conditionals, pointers, arrays etc.)

be able to compile code and manage libraries

be able to understand the operation of, and make simple modifications to Fortran or C code

acquire basic experience with symbolic debuggers

to work alone and in small groups to produce a piece of working code using best principles

to have used common tools for data analysis and visualization in Python

Transferable skills

Coding, debugging, project management, data analysis

Study

Study time

Type	Required
Lectures	10 sessions of 1 hour (33%)
Practical classes	10 sessions of 2 hours (67%)
Total	30 hours

Private study description

Coding practice, further reading.

Costs

No further costs have been identified for this module.

Assessment

You do not need to pass all assessment components to pass the module.

Assessment group A1

	Weighting	Study time
Moodle Quizzes	30%	16 hours

Conceptual understanding of computer architecture and underlying data representation. Tests understanding of the concepts of memory hierarchy and numerical precision.

Practical understanding of the “git” version control system.

Ability to interpret syntax of either Fortran2003 or ANSI C (C89).

Critical evaluation of approaches to writing output files and visualising data. They should demonstrate independent judgement in selecting in-code data reduction versus output size and the practical consequences of these choices.

Individual Programming Assignments	45%	16 hours

1. Students will design and write a short code in either Fortran or C to perform a specified numerical task. Examples would be numerical integration using Simpson’s rule of a specified function, simple iterative refinement of a root using Newton-Raphson method etc. Algorithms needed would be provided in mathematical form but the student would have to interpret this as code. Assessment would be of the submitted code. Duration 1 hour.
2. Students should write a code in C or Fortran making use of arrays to solve a typical numerical problem. Example problems include a simple 1D cellular automaton model etc. All algorithms would be provided, with the student converting this to code. Assessment would be of the submitted code and a simple 1 page description of how the student approached converting the algorithm to software. More weight will be given to the style and design of the code than in Assignment 1, but will still be only a small part of the overall mark. Duration 4 hours.
3. Students will write a simple 2D array based code in C or Fortran linking to the NetCDF (or similar) IO library. The code will have either no core algorithm or an implementation of the algorithm will be provided. The student will also submit a short document describing how to compile their provided source code on the SCRTP system (including descriptions of modules needed etc) and providing a visualisation of the output data in Python (using e.g. Matplotlib, VTK or Mayavi) together with the python code that was produced to visualise the data. Duration 4 hours.

Microproject	25%	20 hours

Students will work in pairs to generate a program in either C or Fortran solving a real but simple numerical problem. The program will be created under git version control and both students must have commits to the repository at the end of the project. The program will take input from a user, iterate through an algorithm and write one or more output files. The algorithm would once again be described in detail but implementation would be up to the students. Example problems would be the Ising model for spins (with details given to the students for dealing with random numbers), or solving the heat equation using Jacobi or Gauss-Seidel iteration. The project would be expected to be accompanied with a Python script to automake visualisation of output files. This would take the last 2 weeks of the course’s workshops and time over the subsequent holiday

	Weighting	Study time
period. Submitted and marked at the start of the following term. Duration 5 Hours.		

Feedback on assessment

Online quizzes will use instant feedback in Moodle/Quizbuilder or similar.\r\n\r\nProgramming assignments will be subject to individual electronic feedback. Earlier assignments will receive high marks if the submitted code is syntactically valid and capable of producing correct output. However, feedback will be provided on style, design and efficiency. Marks will be lost in later assignments if this feedback is not incorporated.\r\n\r\nFeedback on the microproject will be given via group meetings (one per project pair) with a member of the SCRTP research software engineering team.\r\n\r\n\r\n

---

Availability

Courses

This module is Core for:

- Year 1 of TPXA-F344 Postgraduate Taught Modelling of Heterogeneous Systems
- Year 1 of TPXA-F345 Postgraduate Taught Modelling of Heterogeneous Systems (PGDip)