Course ID 025032

Scientific Computing *MATH 49111*

Credit rating 15
ECTS credits 7.5

Unit coordinator: Christopher Johnson

Semester 1

School of Mathematics
Undergraduate ***Level 4***

FHEQ level ' Masters/Integrated Masters P4'

Marketing course unit overview
This course covers the techniques required to develop C++ programs that solve mathematical and scientific problems.
As well as covering the rudiments of C++ (which will be taught with no assumed prior knowledge) the course will also outline the basic techniques used in scientific programming, such as discretisation of equations, numerical error and code validation.
The material is examined primarily through two programming projects, chosen from a list of mathematical topics, which investigate particular algorithms or techniques in more depth. The projects will be assessed by a written report and a demonstration/oral description of the code.
Much of this course is taught in practical computer labs, which limits the number of places available.

Course unit overview
This course covers the techniques required to develop C++ programs that solve mathematical and scientific problems.
As well as covering the rudiments of C++ (which will be taught with no assumed prior knowledge) the course will also outline the basic techniques used in scientific programming, such as discretisation of equations, numerical error and code validation.
The material is examined primarily through two programming projects, chosen from a list of mathematical topics, which investigate particular algorithms or techniques in more depth. The projects will be assessed by a written report and a demonstration/oral description of the code.
Much of this course is taught in practical computer labs, which limits the number of places available.

Aims
To develop the knowledge required to solve mathematical and scientific problems by writing computer programs in C++.

Learning outcomes
On successful completion of this module, students will be able to formulate and write C++ programs to solve mathematical problems. They will understand how to:
- choose appropriate algorithms for the problem,

- develop a suitable program architecture and implement it in C++,
- debug the code and validate its results in the context of the mathematical problem.

Syllabus

Introduction to C++ programming language:

- statements, expressions, control flow, functions, types

- standard C++ library: streams and file i/o, strings, containers, algorithms

- use of external libraries

Code structure and object-oriented programming:

- methods, member data, constructors, destructors, access specifiers

- inheritance, virtual methods and run-time polymorphism.

- operator overloading

Fundamental concepts and techniques:

- numerical error

- discretisation

- writing efficient code (algorithm complexity, optimisation, parallelism)

- communication and visualisation of numerical results

- common algorithms (covered in coursework, and in lectures as time permits) such as numerical linear algebra, root finding, quadrature, sorting, BVPs, PDEs

Debugging and validation:

- error handling

- testing and test-driven development

- debugging

- validation of numerical results

Assessment methods

Weekly courseworks: 10%
Project 1: 40%
Project 2: 50%

Feedback methods
Tutorials will provide an opportunity for students' work to be discussed and provide feedback on their understanding.

Requisites
Students are not permitted to take, for credit, MATH49111 in an undergraduate programme and then MATH69111 in a postgraduate programme at the University of Manchester, as the courses are identical.

Available as free choice? N

Recommended reading
- S.B. Lippman, J. Lajoie, B. Moo. C++ Primer (Fourth edition). Addison Wesley, 2005. (Available as an e-book from the university library)
- W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery. Numerical Recipes: The Art of Scientific Computing (Third edition). Cambridge University Press, 2007.
- B. Stroustrup. The C++ Programming Language (Third edition). Addison-Wesley, 1997
- S. Meyers. Effective C++: 55 specific ways to improve your programs and designs (Third edition). Addison-Wesley, 2005.
- D. Yang. C++ and object-oriented numeric computing for scientists and engineers. Springer, 2000.

Scheduled activity hours

Lectures	11
Practical classes & workshops	33

Independent study hours 106 hours

Version Nbr 008.0.0