MST374 Computational applied mathematics

Presentation pattern

October to June

Module description

The application of numerical methods to mathematical problems is an important skill in applied mathematics. MST374 teaches the theory and the computer programming methods needed to find numerical solutions to mathematical problems encountered in applied mathematics, in the physical, biological and social sciences, as well as in data science.

Topics covered include numerical solutions to systems of linear and nonlinear equations, the convergence of iterative schemes, linear and nonlinear optimisation, numerical solutions to differential equations and data analysis.

MST374 uses Python to demonstrate the methods taught. No previous knowledge of this language is assumed for students though some accommodation for students with prior Python experience has been made. Students may be following one of a range of different qualifications, including degrees in Mathematics, Physics, Engineering or Data Science. A reasonable understanding of linear algebra and multi-variable calculus, as taught in MST210/MST224, is assumed.

The module uses continuous assessment only, with no written examination, since the main focus is to assess computational skills.

Person specification

This section be read in conjunction with the generic person specification for an Associate Lecturer at The Open University and you should address these requirements in any application.

As well as meeting all the requirements set out in the generic person specification, it is essential that you should:

- have a good first degree with significant computational applied mathematics content;
- have experience of successful degree-level teaching preferably of numerical computing, numerical analysis and operational research or other areas of computational applied mathematics;
- demonstrate an understanding of the main module concepts and methods, and the ability and willingness to quickly develop an in-depth understanding of the remaining module content;

- have expertise, or demonstrate the ability and willingness to quickly develop expertise, in using a programming language such as Python for numerical computation, especially within a web-based interactive platform such as Jupyter Notebooks;
- have experience of teaching online or an understanding of the pedagogical issues surrounding online teaching.

It would also be desirable to have:

- a higher degree or PhD (or equivalent);
- experience of using Python and Jupyter Notebooks, ideally in an applied mathematics/numerical computation context;
- experience of teaching mathematics at this level to adults or to students from a broad range of educational backgrounds;
- a teaching qualification in a relevant subject or HEA accreditation.

Additional information

As students on this module are required to submit their Tutor Marked Assignments (TMAs) electronically via the University's online TMA service, you will be required to mark and provide feedback on TMAs submitted electronically and to return the marked work as an electronic file, in the prescribed form, to the online TMA service. If you are invited for an interview and the latter involves an electronic marking exercise, some guidance will be given for this. Further information and advice will be available should you be appointed to the role.

The exact nature of e-learning facilities and University systems for monitoring student progress and handling TMAs will evolve in future, and you will need to be prepared to adapt accordingly.

Module related details

Credits awarded to the student for the successful completion of a module:	30
Number of assignments (TMAs) submitted by the student:	4
Method of submission for assignments:	2: electronic only
Level of ICT requirements:	Web focussed
Number of students likely to be in a standard group:	20
Estimated number of hours per teaching week:	3.5

The module is arranged in 10 Units.

Unit 1: Getting started

Students will start with an introduction to Python in the context of solving equations of one variable using various iterative methods such as simple iteration, bisection methods and the Newton-Raphson method. You'll also learn about the convergence of simple iterative schemes.

Unit 2: Interpolation

This unit introduces practical root-finding, Lagrange interpolation, least-squares curve fitting and splines.

Unit 3: Systems of linear equations

Starts with the solution of linear equations by LU decomposition and then discusses ill-conditioning and applications in finding eigenvalues and least-squares regression analysis.

Unit 4: Data analysis

In this unit students will learn important methods for analysing big data, including singular value decomposition (SVD), principal component analysis (PCA), independent component analysis (ICA), and multidimensional scaling and k-means.

Unit 5: Linear programming

This unit mainly covers the simplex method and includes graphical formulations, the two-phase simplex method, duality and sensitivity analysis.

Unit 6: Systems of nonlinear equations

In this unit students will learn the Newton-Raphson method for multivariate problems and quasi-Newton methods, such as Broyden's method. The unit also includes a further discussion of the convergence of simple iterative schemes.

Unit 7: Nonlinear optimization

This unit starts with minimizing functions of one variable before moving on to multivariate problems that include unconstrained minimization and constrained minimization with equality and inequality constraints.

Unit 8: Differentiation, integration and ordinary differential equations

This unit covers numerical differentiation and numerical integration using Newton-Cotes formulae such as the trapezium and Simpson method. Initial value problems are solved using Euler and Runge-Kutta methods and boundary value and eigenvalue problems are solved using shooting methods.

Unit 9: Random processes

This unit introduces basic theory of random variables, including random walks and Markov chains. Monte Carlo integration is discussed and the unit finishes with the numerical solution of stochastic differential equations.

Unit 10: Case studies

The final unit contains a series of case studies which consolidate ideas presented in the previous units and provide background towards the final end-of-module TMA.