

§4 Course review

MA385 – Numerical Analysis 1

28 November, 2019



< < annotated notes > >

The end...

MA385 is four (or five) main components

- 0 Preliminaries: Taylor's Theorem.
- 1 Solving nonlinear equations.
- 2 Solving initial value differential equations.
- 3 Linear algebra (linear systems of equations, norms and eigenvalues)
- 4 Implementation and investigation of numerical method in Matlab.

Assessment is based on

- two assignments, each worth 10%;
- three labs (weighted 3%, 3% and 4%), worth 10%;
- and the class test, worth 10%;
- and final exam contributes the remaining **60%** of your MA385 grade.

It exam has **4** questions, each worth 20 marks. The number of marks for each section is indicated.

Answer 3 correctly for full marks.

There is be one question on each of the topics (1)–(3), and one on MATLAB.

1. Solving Nonlinear Equations.

(4/16)

Given $f : \mathbb{R} \rightarrow \mathbb{R}$, find $\tau \in [a, b]$ such that $f(\tau) = 0$.

We now know how to solve this

- using interval **bisection**;
- the **Secant** method;
- **Newton's** method;
- **fixed point** iteration.

We also know:

- a sufficient (but not necessary) condition for a solution to exist;
- what the *order of convergence* of a method is;
- for each method, how to prove it converges (subject to certain assumptions); *(not for Secant)*.
- how to determine the order of convergence experimentally;
- the Fixed Point and Contraction Mapping Theorems.

Don't leave home without....

- knowing about Taylor polynomials and remainders;
- being able to state, motivate, and use these methods;
- knowing the terminology (order of convergence, fixed points, contraction, etc.);
- knowing which method is best in which situation;
- being able to prove convergence.

What you don't have to know

- precise assumptions needed for convergence of Newton's and Secant method (will be given in the exam if needed);
- anything about the Black-Scholes equation, Julia sets and roots of unity (won't come up).

Solve the differential equation $y(t_0) = y_0$, $\frac{dy}{dt} = f(t, y)$ $t > t_0$.

We now know:

- what is meant by a **Lipschitz condition**, and how to check that one is satisfied;
- how to derive and use **Euler's** method;
- the definitions of the **global** and **truncation** errors, **consistency** and convergence;
- the conditions on the RK-2 method's parameters for it to be 2nd-order;
- how to show a given RK method has the correct order of convergence for a simple linear problem;
- ~~■ we can summarise an RK method as a tableau;~~

2. Initial Value Problems

(8/16)

- how to apply Euler's method to a system of IVPs;
- how to write a high-order problem as a system of 1st-order IVPs.

Don't leave home without...

- being able to state and use Euler's Method.
- being able to show how it, and a formula for its truncation error, can be derived from a Taylor series;
- for any **one-step** method, knowing how to show it is **consistent**, and how it relates to a Taylor series.

What you don't have to know:

- Picard's Theorem (will be given if needed);
- the definition of the truncation error for an arbitrary one-step method (will be given if needed);
- formulae for any RK-2, RK-3 or RK-4 method (will be given if needed);;
- implicit methods;
- systems and higher-order problems;
- finite difference methods for PDEs (i.e., the heat equation).

Solve the linear system of equations $A\mathbf{x} = \mathbf{b}$ where $A \in \mathbb{R}^{n \times n}$

We now know...

- how to relate systems of equations to matrix-vector equations;
- a good way and a bad way to compute $\det(A)$;
- that Gaussian Elimination and row reduction is effectively the same as ***LU-factorisation***;
- all about **triangular** matrices;
- how construct the *LU*-factorisation of A , and prove it exists;
- how to use *LU*-factorisation and back-substitution to solve $A\mathbf{x} = \mathbf{b}$;
- that the computational cost is $\mathcal{O}(n^3)$;
- all about vector and matrix norms;

- how to derive useful formulae for $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$; how to show they are **consistent**.
- how the **condition number**, $\kappa(A)$, relates to errors in solving $Ax = b$;
- **Gerschgorin's** theorems for estimating eigenvalues, and can prove the first one.

Don't leave home without being able to

leading
1

- explain triangular matrices, matrix partitioning, principal sub-matrices;
- establish properties of the product and inverse of triangular matrices;
- show the existence of the LU -factorisation;
- derive (with justification) the formulae for L and U ;
- write down the LU -factorisation of a given matrix;
- use the LU -factorisation to solve a linear system;
- explain vector and matrix norms, and consistency of matrix norms;
- relate the vector norms $\|x\|_1$, $\|x\|_2$ and $\|x\|_\infty$ to each other.
- prove the formulae for $\|A\|_1$, ~~$\|A\|_2$~~ , and $\|A\|_\infty$.
- state, prove and apply Gerschgorin's 1st Theorem.

- state and apply Gerschgorin's 2nd Theorem.

What you don't have to know

- Permutations/pivoting.
- Computer representation of numbers

We now know

- The fundamentals of MATLAB programming: defining vectors and matrices; basic arithmetic, including element-by-element calculations; plotting, including log-log plots; iterating using `for` and `while` loops; conditionals (`if`), using function handles; output with `fprintf`;
- How to implement key numerical algorithms such as interval bisection, Newton's method, the secant method, Euler's method, (explicit) Runge-Kutta methods.
- How to interpret the results generated by test problems to establish (experimentally) rates of convergence of certain methods.

What you will be asked to do in the question on Implementation/MATLAB *will* include how to implement (as a Matlab code-snippet) **one** of the following algorithms

- Interval bisection;
- The Secant Method
- Newton's Method;
- Euler's Method
- A specified RK-2 or RK-3 method

You may also be expected to explain/interpret the output generated by a MATLAB program.

But not: MATLAB implementation of LU-Factorisation and related topics.

AMA!