Review Guide for Midterm Exam on Friday, 11 October 2024

The in-class Midterm Exam is *closed book* and *closed notes.* No internet, calculator, or electronics are allowed. Please bring only a writing implement. I will aim for an exam which takes 45 minutes to do, but you will have 65 minutes if start right on time. If you come prepared it will be easy.

I encourage you to work with other students on this Review Guide. Please read the relevant parts of the textbook[1] and identify the items you *don't* understand as you get to them.

Problem types. Problems will be in these categories, *with examples in italics*:

- apply an algorithm/method in a simple concrete case,
  Do two steps of bisection on this problem.
- state a theorem or definition,
  State Taylor's Theorem. (I will not ask you to prove *theorems. The two theorems you should memorize are listed below.)*
- write a short pseudocode or MATLAB code to state an algorithm,
  Write Newton's method as a MATLAB *code or pseudocode.*
- explain/show in words, and
  Why is one of these methods better than another, when applied to this example? (Write in complete sentences.)
- derive an algorithm.
  Derive Newton's method. Also draw a sketch which illustrates one step.

Sections. See these textbook sections which we covered in lecture and homework:
2.1–2.10, 4.1–4.5, 5.2–5.4, 5.7, 7.1, 7.2.1–7.2.3, 7.3
Also read the Chapter introductions for Chapters 2, 4, 5, and 7. You can omit Chapter 3 and 6 entirely, for now, but rereading Chapter 1 is not a bad idea. Reading sections 5.1, 5.5, 5.6 is unnecessary but might be interesting.

Definitions. Please be able to use these words correctly and/or write a definition when requested.

- the *absolute error* of $\hat{y}$, a computed quantity, versus the exact value y is $|\hat{y} - y|$
- the *relative error* of $\hat{y}$ versus the nonzero exact value y is $|\hat{y} - y|/|y|$
- *bracket* of a root (defined in class; used for bisection)
- *fixed point* and *fixed point iteration* (section 4.5)
- *floating-point representation*, *machine precision*, *single precision*, *double precision*, *overflow*, and *underflow* (Chapter 5)

[1] Greenbaum & Chartier, *Numerical Methods: ...*, Princeton University Press 2012.

Theorems. You should understand the statements of these theorems, and be able to apply them in particular cases. I will not ask you for the proofs.

- Intermediate Value Theorem (Thm 4.1.1) MEMORIZE
- Taylor's theorem with remainder (Thm 4.2.1) MEMORIZE
- Newton's method converges quadratically theorem (Thm 4.3.1)
- fixed point convergence theorem (Thm 4.5.1)

Algorithms. Please recall these algorithms from memory, or re-derive them as needed.

- bisection method (section 4.1)
- Newton's method (section 4.3)
- secant method (section 4.4.3)
- Gaussian elimination to solve linear systems (section 7.2)
- forward substitution to solve lower-triangular systems (subsection 7.2.2)
- back substitution to solve upper-triangular systems (section 7.2)
- Gaussian elimination *as LU decomposition* (section 7.2)
- Gaussian elimination *with partial pivoting* (subsection 7.2.3)

Your three key concerns about algorithms should be:

(1) What problem does it solve?
(2) Can I run the algorithm by hand in small cases with nice/convenient numbers?
(3) How does it compare to the other algorithms which solve similar/same problems?

Concepts.

- anonymous functions in MATLAB (section 2.8)
- number of steps k for bisection to reduce interval size to 2δ (section 4.1, p. 78)
- Newton's iteration turns a root-finding problem $f(x_*) = 0$ into a fast-converging fixed-point iteration $x_{k+1} = \varphi(x_k)$ where $\varphi'(x_*) = 0$ at the fixed point (section 4.5)
- floating-point representation and IEEE double precision (sections 5.3 & 5.4)[2]
- row operations as left multiplication by lower-triangular matrices (section 7.2)
- counting operations (subsection 7.2.1)
- using a factorization $A = LU$ to solve $A\mathbf{x} = \mathbf{b}$ by two triangular solves (section 7.2)

[2] For a given floating-point system you should *understand* a description of the bit representation. You should *be able to find* the machine precision ϵ, the largest representable number, and the smallest positive (normal) representable number. Other more finicky questions will not be asked.