Review Guide for Final Exam
on Friday, 10 December 2021

F01 students: 10:15am–12:15pm | FXA students: proctored before 5pm

This exam is in-class (or proctored), closed-book, and 120 minutes long. No calculators or computers or phones are allowed. However, **you may bring one page of letter size paper, ONE-sided, with any notes you want on it**.

This exam covers the whole course, but material covered after the Midterm Exam will be more-heavily weighted.

If possible, work through this Guide with other students. Read the relevant parts of the textbook (Driscoll & Braun) and identify the first items you *don't* understand as you get to them. Talk about that! If you come prepared it will be easy.

Sections. Here are textbook sections which can appear on the Final Exam:

1.1–1.2	floating point, conditioning, and stability
2.1	polynomial interpolation as a linear system
2.2–2.6	solving linear systems using LU factorization (Gaussian elimination)
3.1–3.2	fitting data, overdetermined systems, normal equations
4.1–4.2	root-finding and fixed-point iteration
4.3–4.5	Newton's method (including for systems), secant method
5.1	polynomial interpolation again
5.2	piecewise-linear interpolation
5.6–5.7	numerical integration: trapezoid, Simpson's, adaptive
6.1–6.3	numerical solution of ODE IVPs (including systems), Euler's method
6.4	Runge-Kutta methods

Not on the Final Exam. Certain topics and sections *will not* appear on the Final Exam even though we touched them during the semester. Specifically: vector and matrix norms (section 2.7), the matrix condition number (section 2.8), cubic splines (section 5.3), and adaptive Runge-Kutta methods (section 6.5). For these topics either the coverage in class was too brief or the details are too complicated or un-memorable for an in-class exam.

How will I create the exam? I'll ask myself: "Did a question somewhat like that appear on homework?" and "Is it easy to do without a computer?" and "Will the answers be cluttered and hard to grade?" The answers should be: yes, yes, and no!

Categories of problems:

- apply an algorithm/method in a simple concrete case
  E.g. Do two steps of Euler on this ODE problem.
  E.g. Solve this 3×3 *linear system with LU factorization.*

- apply a theorem (no need to *prove* theorems)
  E.g. From the following theorem [Theorem 5.2.2 is stated], what spacing h is needed to get a certain accuracy from piecewise-linear interpolation?
- state a definition
  E.g. Define quadratic convergence *of a sequence.*
- write a short pseudocode or MATLAB code to state an algorithm
  E.g. Write the trapezoid rule as a MATLAB *code or pseudocode.*
- explain/show in words (write in complete sentences)
  E.g. Which of [these methods] is best for [this problem]? Explain.
- derive an algorithm
  E.g. Derive Newton's method. Also draw a sketch which illustrates one step.

Definitions. Be able to use words correctly and/or write the definition if requested.

- *machine epsilon* (section 1.1)
- *absolute accuracy* and *relative accuracy* (section 1.1, page 11)
- *relative condition number* of a problem (section 1.2, formula (1.2.6))
- *Vandermonde matrix* (section 2.1, formula (2.1.2))
- *big-O* notation (section 2.5, page 61)
- *linear least-squares problem* (section 3.1)
- *normal equations* (section 3.2, formula (3.2.1))
- *fixed point* ($x = g(x)$) and *fixed point iteration* (section 4.2, formula (4.2.1))
- *linear convergence* (section 4.2, formula (4.2.4))
- *quadratic convergence* (section 4.3, formula (4.3.5))
- *interpolation problem* (section 5.1)
- *cardinal function* (section 5.1)
- *hat function*, the cardinal functions of PL interpolation (section 5.2)
- *ordinary differential equation initial value problem* (ODEIVP) (section 6.1, eqn (6.1.1))
- *one-step method* (section 6.2, equation (6.23))
- *local truncation error* (section 6.2, equation (6.24))

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

- normal equations (section 3.2, Theorem 3.2.1)
- piecewise-linear error estimate (section 5.2, Theorem 5.2.2)
- existence and uniqueness theorem for ODEIVPs (section 6.1, Theorem 6.1.1)

Algorithms. Recall and/or explain these algorithms as needed. No need to memorize the Matlab; understand the algorithm!

- Horner's rule (Function 1.3.1)
- backward substitution for an upper-triangular system (Function 2.3.2)
- LU factorization (Function 2.4.1)
- Newton's method for a single equation (Function 4.3.1)
- secant method (Function 4.4.1)
- Newton's method for a system (Function 4.5.1)

- trapezoid rule (Function 5.6.1)
- adaptive (Simpson's) integration (Function 5.7.1)
- Euler method for single (scalar) ODEIVPs (Function 6.2.1)
- Euler method for systems of ODEIVPs (Function 6.3.1)
- improved Euler method for ODEIVPs (Function 6.4.1)

The three key concerns. Your key concerns about the above 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 the same problem?

Other concepts.

- anonymous functions in MATLAB (used for integration and ODEs)
- solve $A\mathbf{x} = \mathbf{b}$ by LU factorization, then forward-substitution, and then backward-substitution (section 2.4, page 57)
- what is the basic idea behind all numerical integration methods? (answer: replace the integrand by a polynomial and integrate that)
- how to write a first-order systems of ODEs from a second- or third-order ODE (section 6.3)
- sketch a direction field of an ODE
- sketch Euler method on the direction field of an ODE
- sketch improved Euler method on the direction field of an ODE