Even though the midterms in this class are open book, I highly recommend that you take some time to write up a sheet or two of notes for the exam. It will help you to review the material and organize your notes so that you don't spend the entire time in class scouring your notes, the slides, or the textbook. My best advice for studying is to review the homework and class activities, with a focus on understanding and interpreting the concepts at play. The following is not a comprehensive list.

Three main themes

- Complexity
- Convergence
- Conditioning

Two main problems

- Root-finding: $f(x) = 0$
- Solving a system of equations: $Ax = b$

Important skills and knowledge

- Linear algebra definitions, theorems, and interpretations. A few that we've reviewed include vector norms (different ways to measure distance), matrix norms, the Invertible Matrix Theorem, orthogonal projections, eigenvalues and eigenvectors, and positive definite symmetric matrices
- Use numerical experimentation to explore a concept
- For each of the three main themes, be able explain why, when, and how they are useful in algorithm evaluation, as well as different ways to measure quantities involved (e.g., forward error, backward error, relative forward error, relative backward error, error magnification, condition number)
- Use complexity analysis (big-O notation) to estimate running time of an algorithm, compare algorithms, identify the bottleneck in an algorithm
- Perform convergence analysis. Does an algorithm converge? What is the rate of convergence (linear, super-linear, quadratic, etc.)?
- Understand how computers store numbers in finite memory and the numerical errors that can arise due to that finite memory (binary to decimal, decimal to binary, IEEE floating point representation and arithmetic, machine epsilon, rounding errors, swamping)
- Recognize a model as a system of linear equations and write it down in $Ax = b$ form
- Compare and contrast methods for root-finding and linear equation solving. When will a given method find the right answer? Is it a direct or indirect method? When is it desirable (in terms of complexity, convergence, conditioning, etc.)? Is it especially well-suited to different variants of the problem (e.g., special structure in A, finding multiple different roots)? Here are some of the main methods we've studied:
  - Polynomial evaluation: Horner's method
  - Root-finding: bisection, Newton's, secant, IQI, Brent's
  - $Ax = b$: naive Gaussian elimination, LU, PA=LU, Cholesky

Programming skills

- Basic for/while loops, if-then statements
- Writing functions (including functions that return other functions)
- Basic plotting routines
- Manipulating matrices and vectors
  - Create and fill matrices
  - Access and change certain elements of a matrix
  - Arithmetic: transpose, matrix-matrix multiplication, matrix-vector multiplication
  - Access dimensions
  - Use sparse matrices