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.

Main themes

- Complexity
- Convergence
- Conditioning
- Function approximation / compression
- Orthogonality

Main problems

- Root-finding: $f(x) = 0$
- Solving a system of equations: $Ax = b$
- Interpolation
- Finding a least squares solution to $\min_x ||Ax - b||^2$ when the system of equations $Ax = b$ is overdetermined or when we want to fit a simpler model to approximate data points (rather than say interpolating with a high degree polynomial)

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, orthogonality, orthonormal matrices, orthogonal projections (onto lines or spaces), eigenvalues and eigenvectors, symmetric positive definite matrices, four fundamental spaces (column space, null space, row space, null space of $A^\top$), orthogonal complement of a subspace
- Use numerical experimentation to explore a concept
- For each of the 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, linear equation solving, interpolation, and least squares. 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, Jacobi, conjugate gradient
  - Interpolation: polynomial interpolation (via Vandermonde, Lagrange, or Newton's divided differences), Chebyshev interpolation, splines (also compare and contrast different endpoint conditions)
  - Least squares $\min_x ||Ax-b||^2$: solving the normal equations with Cholesky or conjugate gradient; QR decomposition
- Approximate a function by sampling points (e.g., evenly spaced, Chebyshev nodes) and interpolating a function through those points. Analyze the error between the function and its approximation using the different notions of distance (e.g., 1-norm, 2-norm, ∞-norm). Understand the Runge phenomenon and explain its role in function approximation.
- Understand and explain linear regression (or other least squares problems) as an orthogonal projection, including the role of the following:
  - The normal equations $A^\top Ax = A^\top b$
  - The pseudoinverse $A^\dagger = (A^\top A)^{-1}A^\top$
  - The orthogonal projection operator $P = A(A^\top A)^{-1}A^\top$
  - The optimal residuals, model coefficients, and fitted model values (predictions)
- Given a set of vectors, use the Gram-Schmidt process to find an orthogonal (or orthonormal) set of vectors with the same span. Compute and interpret the reduced and full QR decompositions of a matrix.
- Find the orthogonal projection of (i) one vector onto another vector, or (ii) one vector onto a subspace (using an orthogonal basis for the subspace). Find the distance between a vector and a space (by computing the residual).
- Interpret orthogonal projections geometrically. Draw abstract geometric pictures representing projections (even if the vector spaces can't be drawn like $\mathbb{R}^{23}$).

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