1. LAUNDRY LIST

- web site: http://www.stat.uchicago.edu/~lekheng/courses/309/
- notes: http://www.stat.uchicago.edu/~lekheng/courses/309/notes/
- books: http://www.stat.uchicago.edu/~lekheng/courses/309/books/
- last year’s notes: http://www.stat.uchicago.edu/~lekheng/courses/309f16/notes/
- no required textbook
- main references:
  - Trefethen and Bau
  - Watkins
  - Golub and Van Loan
  - Demmel
- facts about matrices:
  - Bernstein
- homework:
  - homework due beginning of class
  - collaboration allowed but must be declared
  - six assignments, lowest score will be dropped, accounting for 50% of grade
  - no late homework will be accepted
- exams:
  - two quizzes
  - in-class, closed-book, no cheat sheet, 1 hr 20 min
- grade: 50% homework, 50% exams
- office hours: Tue, 3:00pm–5:00pm, Jones 122B

2. NUMERICAL ANALYSIS

- numerical analysis: study of algorithms for continuous mathematics
- examples:
  - linear partial differential equation: given $c_\alpha$’s, find $f$
    \[
    \sum_{|\alpha| \leq n} c_\alpha(t) \frac{\partial^\alpha}{\partial t^\alpha} f(t) = 0 \tag{2.1}
    \]
  - Fredholm integral equation of the first kind: given $K$ and $g$, find $f$
    \[
    \int_\Omega K(s,t)f(t) \, dt = g(s) \tag{2.2}
    \]
  - linear eigenvalue problem: given $c_\alpha$’s, find $f$ and $\lambda$
    \[
    \sum_{|\alpha| \leq n} c_\alpha(t) \frac{\partial^\alpha}{\partial t^\alpha} f(t) = \lambda f(t) \tag{2.3}
    \]
Fredholm integral equation of the first kind: given \( K \) and \( g \), find \( f \) and \( \lambda \)

\[
g(s) + \lambda \int_{\Omega} K(s, t) f(t) \, dt = f(s)
\] (2.4)

nonlinear optimization: given \( f_0, \ldots, f_m \), find \( t_{\text{min}} \)

\[
\min f_0(t) \text{ subject to } f_1(t) \leq 0, \ldots, f_m(t) \leq 0
\] (2.5)

- many scientific and engineering problems can be formulated in one of these forms — the PDE or integral equations would be a mathematical formulation of physical principles like Newton’s second law or Maxwell equations or Schrödinger equation
- we can rarely solve these analytically, i.e., give a useful closed-form formula for the solution
- have to rely on computers, which can only deal with discrete problems
- discretization of (2.1) or (2.2), or Newton method applied to (2.5) yields

\[
Ax = b
\] (2.6)

- discretization of (2.3) or (2.4)

\[
Ax = \lambda x
\] (2.7)

- when we discretize, we have

\[
x = \begin{bmatrix} f(t_1) \\ f(t_2) \\ \vdots \\ f(t_n) \end{bmatrix}
\]

- solving for \( x \) gives us a sample of point values of \( f \), which is often enough for many purposes
- the larger \( n \) is, the more information we get about \( f \)
- the matrix \( A \) comes from discretization of the linear operator — differential operators in the case of (2.1) or (2.3) and integral operators in the case of (2.2) or (2.4)
- example: discretizing a 1-dimensional differential operator

\[
\frac{d^2}{dt^2} \xrightarrow{\text{discretize}} \begin{bmatrix} 2 & -1 \\ -1 & 2 & \ddots \\ \ddots & \ddots & -1 \\ -1 & 2 \end{bmatrix} \in \mathbb{R}^{n \times n}
\]

- example: discretizing a 2-dimensional differential operator

\[
\frac{\partial^2}{\partial t_1^2} + \frac{\partial^2}{\partial t_2^2} \xrightarrow{\text{discretize}} \begin{bmatrix} D & -I \\ -I & D & \ddots \\ \ddots & \ddots & \ddots & -I \\ -I & D \end{bmatrix} \in \mathbb{R}^{mn \times mn}, \text{ where } D = \begin{bmatrix} 4 & -1 \\ -1 & 4 & \ddots \\ \ddots & \ddots & \ddots & -1 \\ -1 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}
\]

- bottom line: many problems in science and engineering require that we solve (2.6) or (2.7)

3. Optimization

- suppose you want to solve an optimization problem

\[
\begin{align*}
\text{minimize} \quad & f(x) \\
\text{subject to} \quad & h_i(x) \leq 0, \quad i = 1, \ldots, p, \\
& Ax = b
\end{align*}
\]
one of the most widely used algorithms is interior point method (essentially Newton’s method adapted to a constrained optimization problem) which requires us to solve a linear system of the form

\[
\begin{bmatrix}
t \nabla^2 f(x_k) + \nabla^2 \varphi(x_k) & A^T \\
A & 0 \\
\end{bmatrix}
\begin{bmatrix}
\Delta x_k \\
\nu_k \\
\end{bmatrix}
=
\begin{bmatrix}
t \nabla f(x_k) + \nabla \varphi(x_k) \\
0 \\
\end{bmatrix}
\]

where \( \varphi \) is the so-called log barrier function that ‘traps’ the iterates \( x_k \) within the region defined by the constraints

at each iterate \( x_k \), we will have to solve such a linear system for \( \Delta x_k \) to obtain the next iterate \( x_{k+1} = x_k + \Delta x_k \)

so the computational cost of interior point methods is largely dominated by the cost of solving linear systems

4. MACHINE LEARNING

many modern problems are information theoretic in nature

– no differential or integral equations describing your solution \( f \)
– but a large test set of given data \( \{(x_i, f(x_i)) : i = 1, \ldots, n\} \) that allows you to guess your \( f \)

example: classification problems

– spam identification

\[ f : \text{emails} \to \{\text{spam, nonspam}\} \]

– image recognition

\[ f : \text{facial images} \to \{\text{male, female}\} \]

or more generally

\[ f : \text{handwritten digits} \to \{0,1,2,3,4,5,6,7,8,9\} \]

– there is no ‘Newton’s law’ type of rule to describe \( f \)

example: supervised learning for binary classification

\[ f : X \to \{-1, +1\} \]

– given training set \( \Omega = \{x_1, \ldots, x_n\} \subseteq X \), i.e., we already know the value \( f(x_i) = y_i \) for any \( x_i \in \Omega \)
– want to find \( f \), i.e., given some \( x \notin \Omega \), we want to predict the value \( f(x) \)
– let us use spam identification as an example, then for any e-mail \( x \in X \),

\[ f(x) = \begin{cases} 
-1 & \text{if } x \text{ is spam} \\
+1 & \text{if } x \text{ is not spam} 
\end{cases} \]

– we can encode an e-mail as a vector in \( \mathbb{R}^N \), for example, by counting word frequencies
– so if you like you may assume that \( X \subseteq \mathbb{R}^N \) where \( N \) is very large

one way to do this:

– assume that

\[ f(x) = \sum_{i=1}^{n} c_i K(x, x_i) \]

where \( K : X \times X \to \mathbb{R} \) is some suitable Mercer kernel

– if \( X \subseteq \mathbb{R}^N \) a common example is the Gaussian kernel

\[ K(x, y) = e^{-\|x-y\|^2/2\sigma^2} \]
– since we already know the value of \( f(x) \) for \( x \in \{x_1, \ldots, x_n\} \), we could in principle determine \( c_1, \ldots, c_n \) by plugging \( x_1, \ldots, x_n \) into (4.1) to get
\[
\begin{align*}
  f(x_1) &= c_1 K(x_1, x_1) + \cdots + c_n K(x_1, x_n) \\
  f(x_2) &= c_1 K(x_2, x_1) + \cdots + c_n K(x_2, x_n) \\
  \vdots &= \vdots \\
  f(x_n) &= c_1 K(x_n, x_1) + \cdots + c_n K(x_n, x_n)
\end{align*}
\]
or equivalently
\[
\begin{bmatrix}
K(x_1, x_1) & K(x_1, x_2) & \cdots & K(x_1, x_n) \\
K(x_2, x_1) & \ddots & \ddots & \vdots \\
\vdots & \ddots & \ddots & \vdots \\
K(x_n, x_1) & \cdots & K(x_n, x_n)
\end{bmatrix}
\begin{bmatrix}
c_1 \\
c_2 \\
\vdots \\
c_n
\end{bmatrix}
= \begin{bmatrix}
f(x_1) \\
f(x_2) \\
\vdots \\
f(x_n)
\end{bmatrix}
\tag{4.2}
\]
or
\[
K c = y
\]
– note that we know all values \( K(x_i, x_j) \) of the matrix and also the right-hand side \( f(x_i) \) as long as we have the training set \( \{(x_i, f(x_i)) : i = 1, \ldots, n\} \)
– so we end up with a linear system like (2.6) again
• in principle this is very nice but in practice it rarely works since (4.2) is unlikely to have a solution
• so what we often need to do is to solve linear systems (2.6) approximately, i.e., \( A x \approx b \) where ‘\( \approx \)’ is interpreted in some appropriate ways — we will look at some of these variants of (2.6) later
• the most common interpretation of \( A x \approx b \) is the least squares problem
\[
\min_{x \in \mathbb{R}^n} \| A x - b \|_2^2 = \min_{x_1, \ldots, x_n \in \mathbb{R}} \sum_{i=1}^{m} \sum_{j=1}^{n} (a_{ij} x_j - b_i)^2
\]
\[
\tag{4.3}
\]
• in the context of supervised learning, this is called empirical risk minimization, i.e., find \( c_1, \ldots, c_n \) so that
\[
\sum_{i=1}^{n} (y_i - f(x_i))^2
\]
\[
\tag{4.4}
\]
is minimized
• but (4.4) is often ill-posed (no unique solution) and so a common strategy is to do Tikhonov regularization and minimize instead
\[
\sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda \| f \|^2_K
\]
where \( \| \cdot \|_K \) is a special norm induced by the kernel \( K(x, y) \) (if you must know, it is called the reproducing kernel Hilbert space or RKHS norm)
• as we will see later in this course, this leads to a problem of the form
\[
(K + \lambda I) c = y
\]
which is again a linear system except that we will need to find \( \lambda \) separately (we will see how to do this)
• now once we have \( c_1, \ldots, c_n \), given any \( x \), we can find the value \( f(x) \)
• of course $f(x)$ would in general not be $\pm 1$ but we can design a rule of the form

$$f(x) \begin{cases} < 0 \Rightarrow x \text{ is spam} \\ > 0 \Rightarrow x \text{ is not spam} \end{cases}$$

• so we have built a spam filter

5. Solving Linear Systems

• most of the course will focus on solving linear systems (2.6) and its variants like least squares, regularized least squares, total least squares, etc
• the fundamental problem is

$$Ax = b$$

where we are given $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^{m}$ and we seek a solution $x \in \mathbb{C}^{n}$
• often we will work over $\mathbb{R}$ instead of $\mathbb{C}$ but these would be only fields of interest
• some of the stuff we say in this course will be false over arbitrary fields (e.g. $\mathbb{F}_2 = \{0, 1\}$ with mod 2 arithmetic)
• three important numbers associated to a matrix $A$ or a linear system $Ax = b$:
  - $m =$ number of rows = number of equations
  - $n =$ number of columns = number of variables
  - $r =$ rank($A$) = dim(im($A$)) = dim(colsp($A$)) = dim(rowsp($A$))
• $m, n, r$ tell us about existence and uniqueness of solution to $Ax = b$

• terminologies
  - $m = n$: $A$ is square matrix, $Ax = b$ is a square system, i.e. number of variables equals number of equations
  - $A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}$
  - $m > n$: $A$ is tall-and-thin matrix, $Ax = b$ is an overdetermined system, i.e., more equations than variables
  - $A = \begin{bmatrix} x & x \\ x & x \\ x & x \\ x & x \end{bmatrix}$
  - $m < n$: $A$ is short-and-fat matrix, $Ax = b$ is an underdetermined system, i.e., more variables than equations
  - $A = \begin{bmatrix} x & x & x & x \\ x & x & x & x \end{bmatrix}$
  - if $r = \min\{m, n\}$, we say $A$ is of full rank, otherwise we say $A$ is rank deficient
    - if $r = \min\{m, n\} = m$, we say $A$ is of full row rank
    - if $r = \min\{m, n\} = n$, we say $A$ is of full column rank
• question: what is the big deal about solving linear systems $Ax = b$? don’t we know all about this already?
• answer: we only know how to solve idealized versions of the problem, but not in realistic situations
  - what if there are rounding errors in the coefficient matrix $A$ or the right hand side $b$
    - or both
  - what if we want to solve it quicker than $O(n^3)$
  - what if $m$ and $n$ are large
– what if we want to do things in parallel on multicore processors
– what if we need to deal with a variant with constraints on the solution \( x \), or where \( Ax = b \) has no solution or no unique solution (as we saw in the machine learning example)

- linear systems are arguably the most widely solved problem in science and engineering
  - 70% of supercomputing time is spent on this
  - that’s why solution of linear system is used to benchmark supercomputers (cf. http://www.top500.org)

### 6. Top 10 Algorithms of the 20th Century
- a broader motivation for this course and its sequel next quarter is that matrix computations are behind some of the most important algorithms
- the three bold faced ones are algorithms in matrix computations
- the four italics ones are algorithms are variants or extensions of algorithms in matrix computations
- see http://www.stat.uchicago.edu/~lekheng/courses/309/top10/

1. Metropolis Algorithm for Monte Carlo
2. Simplex Method for Linear Programming
3. Krylov Subspace Iteration Methods
4. Decompositional Approach to Matrix Computations
5. Fortran Optimizing Compiler
6. QR Algorithm for Computing Eigenvalues
7. Quicksort Algorithm for Sorting
8. Fast Fourier Transform
9. Integer Relation Detection
10. Fast Multipole Method

### 7. Variants of \( Ax = b \)

- notations
  - \( x = [x_1, \ldots, x_n]^T \in \mathbb{R}^n \)
  - \( \|x\|_2 := \sqrt{x_1^2 + \cdots + x_n^2} \)
  - \( \|x\|_1 := |x_1| + \cdots + |x_n| \)
  - \( \|x\|_\infty := \max\{|x_1|, \ldots, |x_n|\} \)
  - \( \|x\|_0 := \text{nnz}(x) = \#\{i : x_i \neq 0\} \)
  - \( A = [a_{ij}]_{i,j=1}^{m,n} \in \mathbb{R}^{m \times n} \)

- \( \|A\|_F = \sqrt{\sum_{i,j=1}^{m,n} |a_{ij}|^2} \)

- we will discuss vector and matrix norms below
- note that \( \| \cdot \|_0 \) is not a norm

(1) linear regression or least squares problem: know \( A \) exactly but \( b \) is corrupted by error \( r \), i.e., \( Ax = b + r \), and we want an \( x \) that minimizes \( r \),

\[
\min\{\|r\|_2^2 : Ax = b + r\} = \min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2
\]  

(7.1)

Gauss–Markov theorem says that such an \( x \) is the maximum likelihood estimator if the error \( r \) is from a distribution that has zero mean and finite variance
(2) error-in-variables regression or total least squares problem: A and b are both corrupted by error E and r, i.e., \((A + E)x = b + r\), and we want an x that minimizes both E and r,
\[
\min \{\|E\|_F^2 + \|r\|_F^2 : (A + E)x = b + r\}
\]
(3) data least squares problem: A is corrupted by error E, i.e., \((A + E)x = b\), and we want an x that minimizes E,
\[
\min \{\|E\|_F^2 : (A + E)x = b\}
\]
(4) minimum norm least squares: want the minimum length solution to (7.1),
\[
\min \{\|x\|_2^2 : x \in \text{argmin} \|Ax - b\|_2^2\} = \min \{\|x\|_2^2 : A^TAx = A^Tb\} \tag{7.2}
\]
the solution \(x_*\) to (7.2) is unique and can in fact be used to define the Moore–Penrose pseudoinverse of \(A\):
\[
x_* = A^\dagger b
\]
(5) robust regression: replace 2-norm by 1-norm (more generally, the Huber loss function) in (7.1),
\[
\min \{\|r\|_1 : Ax = b + r\} = \min_{x \in \mathbb{R}^n} \|Ax - b\|_1
\]
great for reducing sensitivity to outliers
(6) ridge regression or regularized least squares
\[
\min_{x \in \mathbb{R}^n} \|Ax - b\|_2^2 + \|\Gamma x\|_2^2
\]
where \(\Gamma \in \mathbb{R}^{p \times n}\) is some other matrix — most commonly \(\Gamma = \lambda I\) or the finite-difference matrix
(7) sparse or structured linear systems: sparse means \(A\) has a lot of zeroes (sufficiently many that it pays to take advantage of the fact), structured means that \(A\) can be defined with fewer than the usual number of \(mn\) parameters. An example of a data sparse matrix is a Toeplitz matrix
\[
T = \begin{bmatrix}
a_0 & a_1 & a_2 & \cdots & a_{n-1} \\
a_{-1} & a_0 & a_2 & \cdots & \cdots \\
a_{-2} & a_{-1} & \ddots & \ddots & a_2 \\
& \ddots & \ddots & \ddots & \ddots \\
a_{-n+1} & a_{-2} & a_{-1} & a_0
\end{bmatrix} \in \mathbb{R}^{n \times n}
\]
i.e., \(a_{ij}\) depends only on \(|i - j|\), and \(T\) can be specified with just \(2n - 1\) parameters \(a_{-n+1}, \ldots, a_{n+1} \in \mathbb{R}\); a Toeplitz \(Tx = b\) can be solved in \(O(n \log^2 n)\) time as opposed to the usual \(O(n^3)\) time for general linear systems
(8) linear programming:
\[
\min \{c^T x : Ax \leq b\}
\]
note that \(c^T x = c_1 x_1 + \cdots + c_n x_n\) is a linear function; this is very important in economics
(9) quadratic programming: given \(A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, d \in \mathbb{R}^m\), want
\[
\min \left\{ \frac{1}{2} x^T Ax - c^T x : Bx = d \right\}
\]
this reduces to a linear system
\[
\begin{bmatrix}
A & B^T \\
B & 0
\end{bmatrix}
\begin{bmatrix}
x \\
\lambda
\end{bmatrix} =
\begin{bmatrix}
c \\
d
\end{bmatrix}
\]
(10) basis pursuit: if we want the sparsest solution to an underdetermined linear system, we ought to solve
\[
\min \{\|x\|_0 : Ax = b\}
\]
but this is NP-hard and so we look at a convex relaxation

$$\min\{\|x\|_1 : Ax = b\}$$

which can in fact be reduced to a linear programming problem

8. NORMS

- a norm is a real-valued function on a vector space (over $\mathbb{R}$ or $\mathbb{C}$), denoted $\| \cdot \| : V \to \mathbb{R}$ satisfying
  1. $\| x \| \geq 0$ for all $x \in V$
  2. $\| x \| = 0$ if and only if $x = 0$
  3. $\| \alpha x \| = |\alpha| \| x \|$ for all $\alpha \in \mathbb{C}$ and $x \in V$
  4. $\| x + y \| \leq \| x \| + \| y \|$ for any $x, y \in V$

- we will be interested in two specific choices of $V$
  - $V = \mathbb{R}^n$ or $\mathbb{C}^n$
  - $V = \mathbb{R}^{m \times n}$ or $\mathbb{C}^{m \times n}$

9. VECTOR NORMS

- if $V = \mathbb{C}^n$ or $V = \mathbb{R}^n$, we call a norm on $V$ a vector norm
- example: consider $\| \cdot \|_1 : \mathbb{C}^n \to \mathbb{R}$ defined by

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

for $x = [x_1, \ldots, x_n]^T \in \mathbb{C}^n$ and where $|x|$ denotes the modulus/absolute value of $x \in \mathbb{C}$

- check that this is a norm:
  1. clearly $\|x\|_1 \geq 0$
  2. the only way a sum nonnegative entries $\|x\|_1 = 0$ is if all entries $|x_i| = 0$ and so $x = [0, \ldots, 0]^T = 0$
  3. we have $\|\alpha x\|_1 = \sum_{i=1}^{n} |\alpha x_i| = |\alpha| \sum_{i=1}^{n} |x_i| = |\alpha| \|x\|_1$

since complex modulus satisfies $|\alpha x| = |\alpha||x|$ 

- using the triangle inequality for complex numbers, we obtain

$$\|x + y\|_1 = \sum_{i=1}^{n} |x_i + y_i| \leq \sum_{i=1}^{n} |x_i| + |y_i| \leq \|x\|_1 + \|y\|_1$$

- therefore the function defines a norm, called the 1-norm or Manhattan norm

- example: more generally, for $p \geq 1$ (can be any real number, not necessarily an integer), we define the $p$-norm $\|x\|_p$ by

$$\|x\|_p = (|x_1|^p + \cdots + |x_n|^p)^{1/p}$$

- most commonly used $p$-norms is the 2-norm or Euclidean norm:

$$\|x\|_2 = \left( \sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

- easy to see that for any $p$, we have

$$\left( \max_{i=1,\ldots,n} |x_i|^p \right)^{1/p} \leq \|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p} \leq \left( n \max_{i=1,\ldots,n} |x_i|^p \right)^{1/p}$$
– from which it follows that
\[ \max_{i=1,\ldots,n} |x_i| \leq \|x\|_p \leq n^{1/p} \max_{i=1,\ldots,n} |x_i| \]
– as \( p \to \infty \), we obtain the \textit{infinity norm}
\[ \|x\|_\infty = \lim_{p \to \infty} \|x\|_p = \max_{i=1,\ldots,n} |x_i| \]
which is also known as the \textit{Chebyshev norm}
– easy to verify that \( p \)-norms for any \( p \in [1, \infty] \) are indeed norms
• generalization of the \( p \)-norm is the \textit{weighted \( p \)-norm}, defined by
\[ \|x\|_{p,w} = \left( \sum_{i=1}^{n} w_i |x_i|^p \right)^{1/p} \]
– again it can be shown that this is a norm as long as the \textit{weights} \( w_i, i = 1, \ldots, n \), are strictly positive real numbers
• example: a vast generalization of all of the above is the \( A \)-norm or \textit{Mahalanobis norm}, defined in terms of a matrix \( A \) by
\[ \|x\|_A = (x^* Ax)^{1/2} = \left( \sum_{i,j=1}^{n} a_{ij} x_i x_j \right)^{1/2} \]
– this defines a norm provided that the matrix \( A \) is positive definite
– note that if \( W = \text{diag}(w) \), then
\[ \|x\|_W = \|x\|_{2,w} \]
• we now highlight some additional, and useful, relationships for a norm
– first of all, the triangle inequality generalizes directly to sums of more than two vectors:
\[ \|x + y + z\| \leq \|x + y\| + \|z\| \leq \|x\| + \|y\| + \|z\| \]
– more generally,
\[ \left\| \sum_{i=1}^{m} x_i \right\| \leq \sum_{i=1}^{m} \|x_i\| \]
– secondly, what can we say about the norm of the difference of two vectors? we know that \( \|x - y\| \leq \|x\| + \|y\| \) but we can obtain a more useful relationship as follows:
\[ \|x\| = \|(x - y) + y\| \leq \|x - y\| + \|y\| \]
we obtain
\[ \|x - y\| \geq \|x\| - \|y\| \]
– thirdly, from
\[ \|y\| = \|y - x + x\| \leq \|x - y\| + \|x\| \]
it follows that
\[ \|x - y\| \geq \|y\| - \|x\| \]
and therefore
\[ \|\|x\| - \|y\|\| \leq \|x - y\| \] (9.1)
– the inequality (9.1) yields a very important property of norms, namely, they are all (uniformly) continuous functions of the entries of their arguments