MACM 316 Spring 2025

Alexander Ng

Monday, April 14, 2025

0.1 Preface

Hi reader, the following is a collection of notes I took while taking SFU's MACM 316 (Numerical Analysis) in Spring 2025. Many of the notes are heavily based on the lecture notes given by Professor Steven Ruuth, which can be found in this repository at `./steve's notes`. I have tried to make the notes as readable as possible, and every file is compiled in a machine-readable format so you can easily stick them into an AI knowledge base to help you with your studies.

If you need help getting back up to speed on Linear Algebra, I highly recommend taking a look at An Infinitely Large Napkin by Venhance. It's a great book that covers all of the theory of mathematics that you will need to understand the material in this course. Overtime, I will be updating these notes to be better structured based on the chapters of the textbook that the course is structured around.

0.2 Preface

Many real-world problems stem from numerical analysis, particularly poor execution. Rounding errors, insufficient representation problems, and other such problems represent the significant impact of computation in the real world.

Check out the following resources for more information:

1. https://www-users.cse.umn.edu/ arnold/disasters/
2. https://web.ma.utexas.edu/users/arbogast/misc/disasters.html

0.3 Computer Arithmetic

We often want to work with the real number system, which consists of all integers, rational and irrational numbers

$$2, \sqrt{2}, e, \pi, 10^6, \text{ etc.}$$

Because we have a finite space limitation for numbers, **not all numbers can be represented exactly.** This can cause problems with arithmetic.

0.3.1 Bases

We typically use the decimal (base 10) system, e.g.

$$427.325 = 4 \times 10^2 + 2 \times 10^1 + 7 \times 10^0 + 3 \times 10^{-1} + \dots$$

However, when we work with a computer, we use the binary (base 2) system, e.g.

$$(1001.11101)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} + \dots$$

0.4 Base Conversion and Error

Because it is impossible to represent some finite decimal fractions in binary, we will (definitely) encounter **error** when converting from base-10 to base-2.

0.4.1 Example

Assume $\frac{1}{10} = (0.a_1 \dots a_n)_2$ where $a_i \in \{0,1\}$.

To convert, we can multiply by 2:

$$\frac{2}{10} = 0.2 = (a_1.a_2a_3\dots)_2$$

We take the integer part of both sides:

$$\begin{aligned} 0.2 &= a_1.a_2a_3\dots \\ 0 &= a_1 \end{aligned}$$

Now, we know that $a_1 = 0$. We can continue this process to get the next digit:

$$\begin{aligned} \frac{4}{10} &= 0.4 = a_2.a_3a_4\dots \\ &\implies a_2 = 0 \end{aligned}$$

Again:

$$\begin{aligned} \frac{8}{10} &= 0.8 = a_3.a_4a_5\dots \\ &\implies a_3 = 0 \end{aligned}$$

Once more:

$$\begin{aligned} \frac{16}{10} &= 1.6 = a_4.a_5a_6\dots \\ &\implies a_4 = 1 \end{aligned}$$

At this point, we know that $a_1 \dots a_3 = 0$ and $a_4 = 1$, all from taking the integer part of the fraction. Since we just returned a 1 from this process, we will subtract 1 from both sides and continue to the next digit:

$$\frac{16}{10} - 1 = \frac{6}{10} = 0.a_5a_6\dots$$

Multiply by 2 to get the next digit:

$$\frac{12}{10} = 1.2 = a_5.a_6a_7\dots$$
$$\implies a_5 = 1$$

Again subtract 1 from both sides:

$$\frac{12}{10} - 1 = \frac{2}{10}$$

Since we got back to $\frac{2}{10}$, which was our starting point, we know that every part of this process will repeat forever. Therefore, $\frac{1}{10}$ has an infinitely repeating binary representation. There is **no** way to represent $\frac{1}{10}$ in finite-representation binary.

$$\frac{1}{10} = 0.0001100110011\dots$$

0.5 Hypothetical Storage Scheme (32-bit)

We will use a hypothetical decimal computer since the concept is identical.

(By the way, this is almost exactly identical to IEEE-754 floating point representation, except that we are using a decimal representation instead of binary.)

Suppose we have the decimal number 423.7. Since we always want to represent numbers in **proper scientific notation**, we normalize the mantissa.

We write our number as follows:

$$423.7 = +0.4237 \times 10^{+3}$$

Notice how the + is relevant since we are going to explicitly represent the sign of the number.

We call the bits following from the decimal point (4273) the **mantissa**. We include 1 bit for the sign, which is 1 for positive numbers, 1 bit for the exponent sign, 7 bits for the exponent, and the remaining 23 bits for the mantissa. (See diagram below)

Because our storage format is finite, one of the biggest problems we will encounter is **bit overflow**. The maximum magnitude of our exponent (in

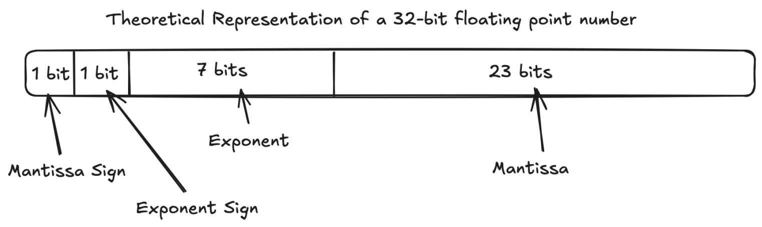

Figure 1: Hypothetical Storage Scheme (32-bit)

binary) is **127**, so our number can only range from 2^{-127} to 2^{+127}. Because our mantissa only has 23 bits of precision, the precision will decrease as our numbers get larger because we use exponentiation to represent the actual number.

Ex.: Consider the number $2^{25} = 33,554,432$. This number can be represented exactly in binary. However, the number $2^{25} + 1 = 33,554,433$ cannot be represented exactly, since it can't fit in 23 bits.

All numbers from $2^{25} - 1$ through $2^{25} + 2$ are represented with the same mantissa in binary. Only when you reach $2^{25} + 3$ does the mantissa change.

Another fun note is that within IEEE-754 floating point representation, the number of representable values within a given exponent is the same, regardless of the exponent. This may seem obvious, but it's interesting nonetheless. This fact comes from the fact that the number of bits in the mantissa is fixed, and the number of representable values is exactly $2 \times 2^{23} = 2^{24}$, since each positive value has a negative counterpart.

0.6 Floating Point Decimal Normalization

Can we write all real numbers in normalized scientific notation?

$$
\begin{aligned}
732.5051 &\to +0.7325051 \times 10^{+3} \\
-0.005612 &\to -0.5612 \times 10^{-2}
\end{aligned}
$$

For $x \in \mathbb{R}$, we can express it as:

$$x = \pm r \times 10^{\pm n}, \quad \text{where } \frac{1}{10} \leq r \leq 1.$$

In binary, we write:

$$x = \pm q \times 2^{\pm m}, \quad \text{where } \frac{1}{2} \leq q < 1.$$

Here, q is the mantissa and m is the integer exponent.

We limit r and q so that when $k < 1/\text{BASE}$, we can shift the decimal place and normalize the number further. When we have $1.x$, we rewrite it as $0.x \times \text{base}^1$.

0.7 Rounding or Chopping (Sources of Error)

Given $x = 0.a_1a_2 \dots a_na_{n+1} \dots a_m$ using m digits, rounding to n places follows:

- If $0 \leq a_{n+1} < 5$, then $x = 0.a_1a_2 \dots a_n$.
- If $5 \leq a_{n+1} \leq 9$, then $x = 0.a_1a_2 \dots (a_n + 1)$.

Example:

$$
\begin{aligned}
\text{round}(0.125) &= 0.13, \\
\text{round}(-0.125) &= -0.13.
\end{aligned}
$$

Instead of rounding, truncation follows:

$$x = 0.a_1a_2 \dots a_n.$$

Truncation introduces larger errors but is computationally cheaper than rounding.

0.8 Error

We define:

- Absolute error: $|p - p^*|$.
- Relative error: $\frac{|p-p^*|}{|p|}$.

Absolute error is used when magnitude matters, particularly for small values. Relative error is preferred when values differ in scale.

Example:

$$\text{Exact: } 0.1, \quad \text{Approximate: } 0.099,$$
$$\text{Relative Error: } \frac{|0.1 - 0.099|}{0.1} = 0.01.$$

t	5×10^{-t}	Is error within bound?
0	5	✓
1	0.5	✓
2	0.05	✓
3	0.005	$\times$

Since $0.01 < 5 \times 10^{-2}$ but not 5×10^{-3}, we have two significant digits.

0.9 Computations and Machine Representation

Let $\text{fl}(x)$ denote the machine representation of x. Computations on a machine follow:

$$\text{fl}(\text{fl}(x) + \text{fl}(y)).$$

Each step introduces an error.

Example:

$$p = 0.54617, \quad q = 0.54601,$$
$$r = p - q = 0.00016.$$

With 4-digit rounding,

$$p^* = 0.5462, \quad q^* = 0.5460,$$
$$r^* = p^* - q^* = -0.0002.$$

Relative error:

$$\frac{|r - r^*|}{|r|} = 0.25.$$

A high relative error results when subtracting close numbers.

0.10 Minimizing Error

Consider computing $f(x) = \frac{1-\cos x}{x^2}$ for $\bar{x} = 1.2 \times 10^{-5}$.

With 10-digit rounding:

$$\begin{aligned} c &= \mathrm{fl}(\cos \bar{x}) = 0.9999999999, \\ 1 - c &= 0.0000000001. \end{aligned}$$

This results in a large error.

Using $\cos x = 1 - 2\sin^2(x/2)$:

$$f(x) = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2.$$

This provides a more accurate computation.

Conclusion: Avoid subtracting close numbers. Use alternative representations like Taylor series or trigonometric identities.

0.11 Reducing Roundoff Error

One way to reduce roundoff error is to minimize the number of floating-point operations.

0.11.1 Polynomial Evaluation Using Nested Multiplication

Consider evaluating the polynomial:

$$f(z) = 1.01z^4 - 4.62z^3 - 3.11z^2 + 12.2z - 1.99.$$

We can rewrite this expression using nested multiplication:

$$\begin{aligned} f(z) &= (1.01z^3 - 4.62z^2 - 3.11z + 12.2)z - 1.99 \\ &= ((1.01z^2 - 4.62z - 3.11)z + 12.2)z - 1.99 \\ &= \dots \end{aligned}$$

By factoring out z as much as possible, we reduce the total number of floating-point operations, minimizing error accumulation.

0.12 Cancellation Errors

0.12.1 Quadratic Formula and Cancellation Errors

Consider solving the quadratic equation:

$$ax^2 + bx + c = 0.$$

Using the quadratic formula:

$$\begin{aligned} x_1 &= \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \\ x_2 &= \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \end{aligned}$$

Suppose $b = 600$, $a = c = 1$. The issue arises because $-b$ is close in magnitude to $+\sqrt{b^2 - 4ac}$, causing significant cancellation error in x_1.

0.12.2 Reformulating to Reduce Cancellation

We rationalize the numerator:

$$x_1 = \frac{(-b + \sqrt{b^2 - 4ac})}{2a} \times \frac{(-b - \sqrt{b^2 - 4ac})}{(-b - \sqrt{b^2 - 4ac})}.$$

This simplifies to:

$$x_1 = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})}.$$

Now, the cancellation error is eliminated. If $b = -600$, the same issue occurs with x_2, and we apply the same rationalization technique.

0.13 Review of Taylor Series

Taylor's theorem is fundamental for numerical approximations.

0.13.1 Definition of Taylor Series

Given a function $f(x)$ that is sufficiently smooth on $[a, b]$, we can approximate $f(x)$ with a Taylor polynomial $P_n(x)$:

$$P_n(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2!}(x - x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n.$$

Here, $x_0, x \in [a, b]$.

0.13.2 Conditions for Taylor Series Expansion

For $f(x)$ to have a valid Taylor series expansion:

- $f \in C^n[a, b]$ (i.e., f, f', f'', ..., f^n must be continuous).
- $f^{(n+1)}$ must exist on $[a, b]$.

0.13.3 Error in Taylor Approximation

The error in Taylor series approximation is given by:

$$f(x) = P_n(x) + R_n(x),$$

where the remainder term $R_n(x)$ satisfies:

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x - x_0)^{n+1},$$

for some $c \in (x_0, x)$. The approximation is most accurate when x is close to x_0.

0.14 Example: Third-Order Taylor Polynomial for $\sin(x)$

Find $P_3(x)$ for $f(x) = \sin(x)$ centered at $x_0 = 0$.

$$\begin{aligned} P_3(x) &= f(0) + f'(0)(x-0) + \frac{f''(0)}{2!}(x-0)^2 + \frac{f'''(0)}{3!}(x-0)^3 \\ &= x - \frac{x^3}{6}. \end{aligned}$$

0.14.1 Error Analysis

The remainder term for $n = 3$ is:

$$R_3(x) = \frac{f^{(4)}(c)}{4!}x^4.$$

Since $f^{(4)}(x) = \sin(x)$, we have:

$$R_3(x) = \frac{\sin(c)}{24}x^4.$$

For $x = 0.1$:

$$|R_3(0.1)| \leq \frac{|\sin(0.1)|}{24}(0.1)^4 < 4.2 \times 10^{-7}.$$

This shows the high accuracy of the Taylor series approximation.

0.15 Linear Approximation of $\sqrt{16.1}$

We approximate $\sqrt{16.1}$ without using the square root algorithm.

0.15.1 Solution

Let $f(x) = \sqrt{x}$ and choose an expansion point $x_0 = 16$, since $\sqrt{16} = 4 \in \mathbb{Z}$ is easily computable. Using the first-order Taylor approximation:

$$f(x_0 + h) \approx f(x_0) + hf'(x_0),$$

where $h = 0.1$.

0.15.2 Computation

$$\begin{aligned} f(16) &= \sqrt{16} = 4, \\ f'(x) &= \frac{1}{2\sqrt{x}}, \quad f'(16) = \frac{1}{8}, \\ f(16.1) &\approx 4 + 0.1 \times \frac{1}{8} = 4.0125. \end{aligned}$$

The exact value is $4.01248052955\ldots$, with a small truncation error. Since the machine error is on the order of 10^{-14}, it is negligible compared to the Taylor approximation error.

0.16 Algorithm Quantification

Numerical methods construct a sequence of better approximations, converging to a solution α.

Given a sequence $\{\alpha_n\}$:

$$\lim_{n\to\infty} \alpha_n = \alpha.$$

We quantify convergence speed by analyzing $|\alpha - \alpha_n| \leq c$, where c is a target error.

0.16.1 Example: Convergence of $\sin(1/n)$

Consider $\alpha_n = \sin(1/n)$, which converges to $\alpha = 0$ as $n \to \infty$. We rewrite:

$$\lim_{n\to\infty} \sin(1/n) = \lim_{h\to 0} \sin(h),$$

which is easier to analyze. Expanding $\sin(h)$ in a Taylor series:

$$\sin(h) = h - \frac{h^3}{3!} + \frac{h^5}{5!} + \dots.$$

For small h, $\sin(h) \approx h$, implying:

$$|\alpha_n - \alpha| \leq \frac{1}{n}.$$

Thus, α_n converges to $\alpha = 0$ with rate of convergence $O(1/n)$.

0.17 Big-O Notation

For a sequence $\{A_n\}$, if:

$$|A_n - A| \leq k|B_n| \quad \text{for sufficiently large } n,$$

where k is a constant, then we say:

$$A_n = A + O(B_n).$$

0.17.1 Example: $\sin(1/n)$ Convergence

From before, $|\alpha_n - \alpha| \leq 1/n$, so:

$$A_n = \sin(1/n) \text{ converges to } A = 0 \text{ with rate of convergence } O(1/n).$$

0.17.2 Example: Convergence of $n\sin(1/n)$

We evaluate:

$$\lim_{n\to\infty} n\sin(1/n) = 1.$$

Changing variables, $h = 1/n$, we obtain:

$$\lim_{h\to 0} \frac{\sin(h)}{h} = 1.$$

Expanding $\sin(h)/h$ in Taylor form:

$$\frac{\sin(h)}{h} = 1 - \frac{h^2}{6} + O(h^4).$$

For small h:

$$\frac{\sin(h)}{h} - 1 \approx -\frac{h^2}{6},$$

so α_n converges to $\alpha = 1$ with rate $O(1/n^2)$.

0.18 Takeaways

Big-O notation quantifies algorithm efficiency by ignoring constants and focusing on convergence trends. Constants vary across systems, so we care about general convergence patterns rather than specific values.

0.19 Review

Why did we expand around $h_0 = 0$?

Because we want to know what happens with h_0, choosing 0 as our expansion point makes sense. The Taylor series approximation is more accurate the closer you are to your expansion point.

0.20 Another Example

Find the rate of convergence of the following $h \to 0$.

$$\lim_{h\to 0} \cos h + \frac{1}{2}h^2 = 1$$

$\alpha = 1$
$\alpha_h = cosh + \frac{1}{2}h^2$

$$\begin{aligned} \alpha_h - \alpha &= cosh + \frac{1}{2}h^2 - 1 \\ &= 1 - \frac{h^2}{2!} + \frac{h^4}{4!} + O(h^6) + \frac{1}{2}h^2 - 1 \\ &= \frac{h^4}{4!} + O(h^6) \end{aligned}$$
$$= O(h^4)$$

0.21 Chapter 6 - Direct methods for solving linear systems

Preface

In MATH 240, we learned methods for solving linear systems of equations where our $n \times m$ matrix has few rows and few columns. In Numerical Analysis, we will learn methods for solving linear systems where our matrix has thousands or millions of rows and columns.

We will first study methods that give an answer ina fixed number of steps, subject only to roundoff errors. This method only has error from the accumulation of numerical representation errors.

0.21.1 Linear Systems of Equations

Linear Systems of Equations

$$\begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{bmatrix}$$

0.21.2 Elementary Row Operations

1. Multiply row i by a constant $\lambda \neq 0$, denoted by $\lambda E_i \to E_i$
2. Add a multiple of row i to row j, denoted by $E_i + \lambda E_j \to E_i$
3. Interchange rows i and j, denoted by $E_i \leftrightarrow E_j$

0.21.3 Notes on Gaussian Elimination

The idea behind Gaussian Elimination is to transform the matrix into a 'triangular' equivalent matrix problem of the upper triangular or lower triangular form.

0.21.4 Example 1

$$\begin{bmatrix} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 12 \end{bmatrix}$$

0.22 Complexity of Gaussian Elimination

How does the number of operations change with the size of the matrices?

We can count up the number of multiplications and divisions to go to upper triangular form.

$$\begin{bmatrix} * & \dots & * & x_1 \\ \vdots & \ddots & \vdots & \vdots \\ * & \dots & * & x_n \end{bmatrix} \to \begin{bmatrix} * & * & \dots & * & x_1 \\ 0 & * & & \vdots & \vdots \\ \vdots & \vdots & & * & x_n \\ 0 & * & \dots & * & x_{n+1} \end{bmatrix}$$

This proceeds as follows:
Provided $a_{11} \neq 0$, the operation corresponding to

$$E_j - \frac{a_{ji}}{a_{11}} E_i \to E_j$$

0.23 Partial Pivoting

Parital pivoting is the simplest technique to avoid generating massive round-off errors in the Gaussian Elimination algorithm.

The idea is to find the largest element beneath the pivot and swap its row with the pivot row.

Parital pivoting is sufficient for most linear systems. However, it can be inadequate for certain problems.

0.23.1 Example

Consider the linear system:

$E_1 : 30.00x_1 + 591400x_2 = 491700$

$E_2 : 5.291x_1 - 6.130x_2 = 46.78$

No row exchanges are carried out during partial pivoting.

Now, the multiplier is $m_{21} = \frac{5.291}{30.000} = 0.1764$ and $(E_2 - m_{21}E_1 \rightarrow E_1)$ gives the system

$30.00x_1 + 591400x_2 \approx 591700$

$-104300x_1 \approx -104400$

0.24 Scaled Partial Pivoting

We can improve the accuracy of the partial pivoting algorithm if we scale coefficients before deciding on row exchanges.

The scaling factor is the largest absolute value of any coefficient in the current row. The idea is to select the largest scaled value a_{ik}/S_i corresponding to elements that are below the pivot.

The extra work to apply the partial pivoting algorithm is some constant times $O(n^2)$.

In rare instances, complete pivoting may be needed, which is $O(n^3)$ extra work.

0.24.1 An excerpt on Time Complexity

Total cost of partial pivoting is $O(n^3) + O(n^2) = O(n^3)$

Complete pivoting is $O(n^3) + O(n^3) = O(n^3)$

In the end, the dominant term of complete pivoting is $(c_1 + c_2)O(n^3)$, which is technically more than $c_1O(n^3)$ for partial pivoting, but it is the extra work is insignificant compared to the total work of the actual algorithm.

0.25 Some Review (Definitions)

1. Two matrices A and B are equal if they have the same size and if each element a_{ij} in A is equal to b_{ij} in B.

2. $A + B$ for two similarly sized matrices A and B is defined as the $n \times n$ matrix whose entries are $(a_{ij} + b_{ij})$ for all $i = 1..n$ and $j = 1..m$.

3. λA for a scalar λ and a matrix A is defined as the matrix whose entries are λa_{ij} for all $i = 1..n$ and $j = 1..m$.

0.26 Determinants

A very useful concept of linear algebra is the determinant of a matrix. The determinant of a matrix A is denoted by $\det(A)$ or $|A|$.

Determinants are important, in part, because of the following theorem:

0.26.1 Theorem

The following statements are equivalent for any $n \times n$ matrix A.

1. $\det(A) \neq 0$
2. The equatoin $Ax = 0$ has a unique solution $x = 0$.
3. The system $Ax = b$ has a unique solution for any n-dimensional column vector b.
4. The matrix A is nonsingular.
   i.e. A^{-1} exists.

0.26.2 Definition of the Determinant

The definition of the determinant is somewhat involved:

(a) If $A = [a]$, then $\det(A) = a$

(b) If A is some $n \times n$ matrix, the minor M_{ij} of A is the determinant of the $(n-1) \times (n-1)$ submatrix of A obtained by removing the i^{th} row and j^{th} column.

(c) The $\underline{\text{cofactor}}$ A_{ij} of A associated with M_{ij} is defined by $A_{ij} = (-1)^{i+j} \det(M_{ij})$.

(d) The determinant of the $n \times n$ matrix A, when $n > 1$ is given either by

$\det(A) = \sum_{j=0}^{n} a_{ij} A_{ij}$ (Row cofactor expansion) or

$\det(A) = \sum_{j=0}^{n} a_{ij} A_{ij}$ (Column cofactor expansion)

0.26.3 Complexity of the Determinant Algorithm

Assume there are no free zero entries in A, and you must compute the fully expanded determinant of A.

Work (in general) for a 4x4 matrix

$$\begin{aligned}
&= 4 \times (\text{work to compute } \det(3 \times 3)) \\
&= 4 \times 3 \times (\text{work to compute } \det(2 \times 2)) \\
&= 4 \times 3 \times 2 \times (\text{work to compute } \det(1 \times 1)) \\
&= 4 \times 3 \times 2 \times 1 \\
&= \boxed{4!}
\end{aligned}$$

Which implies that the determinant of a $n \times n$ matrix can be computed in $O(n!)$ time.

0.27 More ways of computing the determinant

If $A = [a_{ij}]$ is an $n \times n$ matrix that is either upper or lower triangular form, then $\det(A) = \prod_{i=1}^{n} a_{ii}$.

0.28 LU-Decomposition

Suppose that we wanted to solve the system

$$Ax = b_k \qquad A \in \mathbb{R}^{n\times n}$$

for several different $b_k \in \mathbb{R}^n$.

This type of matrix problem arises frequently in initial value problems. If we apply Gaussian Elimination to solve the system, then $O(n^3)$ operations are required for each b_k. On the other hand, suppose we could factor A into the form $A = LU$ where L is lower triangular and U is upper triangular.

Then, we can let $y = Ux \implies Ly = b_k$, and solving for y via forward substitution is $O(n^2)$ operations.

Next, we have to solve for $Ux = y$ via backward substitution, which is also $O(n^2)$ operations.

The total number of operations is $O(n^2)+O(n^2) = O(n^2)$, which is much better than the $O(n^3)$ operations required for Gaussian Elimination.

Finding L and U is $O(n^3)$ requires $O(n^3) + O(n^3)$ operations **HOWEVER**, by doing this expensive operation once, we can save a lot of operations when there are many b_k.

0.28.1 LU-Factorization

We will proceed to derive this LU-factorization using Gaussian Elimination.

Example:

$$\begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 39 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} 1 \\ 5 \\ 14 \end{bmatrix}$$

We want to zero out entries below the pivot element a_{11} in the first row. So we want to take $E_2 - 3E_1 \to E_2$. This same effect can be obtained by multiplying A by the elementary matrix E_{21}.

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 39 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 4 & -8 & 30 \end{bmatrix}$$

An elementary matrix is equal to the edentity matrix except for one nonzero entry off of the main diagonal.

Now we want to zero out the remaining entry below the pivot.

This can be done by taking $E_3 - 2E_1 \to E_3$ or by left-multiplying A by the elementary matrix E_{31}.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 4 & -8 & 30 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & -4 & 24 \end{bmatrix}$$

To obtain an upper triangular system, we would finally zero out the remaining entry below the pivot in E_2, taking $E_3 - 4E_2 \to E_3$. Or, we can left multiply by the elementary matrix E_{32}.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -4 & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & -4 & 24 \end{bmatrix} = \begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & 0 & 4 \end{bmatrix}$$

Notice that $E_{32}E_{31}E_{21}A$ is upper triangular.

Set $U = E_{32}E_{31}E_{21}$ and $L = A$.

Now $E_{32}^{-1}E_{31}^{-1}E_{21}^{-1}U = L$.

To find the inverses of elementary matrices, we have to recall their meaning.

For example, $E_{21} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ subtracts 3 times the first row from the second.

The inverse will need to add 3 times the first row to the second.

$$\implies (E_{21})^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Which costs $O(1)$ operations to flip the sign of the single entry.

Furthermore, notice that

$L \cong E_{21}^{-1}E_{31}^{-1}E_{32}^{-1}$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 4 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 4 & 1 \end{bmatrix}$$

So the matrix multiplication can be performed by putting the nonzero offdiagonal elements of the elementary matrices intot he appropriate positions in the matrix L.

This means that the matrix L is easily constructed during the Gaussian Elimination process just by storing the multipliers.

We conclude that

$$\underbrace{\begin{bmatrix} 2 & -2 & 3 \\ 6 & -7 & 14 \\ 4 & -8 & 30 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 2 & 4 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 2 & -2 & 3 \\ 0 & -1 & 5 \\ 0 & 0 & 4 \end{bmatrix}}_{U}$$

0.29 LU Decomposition (contd.)

THM: If Gaussian elimination can be performed without row exchanges, then the matrix A has a unique LU factorization where L is lower triangular and with all diagonal entries equal to 1 and U is an upper triangular matrix.

Once the matrix factorization is complete, the solution to

$$LUx = b$$

is found by first setting

$$y = Ux$$

then determining the vector y from $Ly = b$ using forward substitution. The variable x is found from $Ux = y$ using backward substitution.

Notice: in the theorem, **we need to be able to perform Gaussian Elimination without row exchanges**. This means that we can't use row exchanges to solve the system.

0.30 Matrix Factorization

If A is any nonsingular matrix, $Ax = b$ can be solved by Gaussian Elimination with the possibility of row exchanges. If we can find the row interchanges required to solve the system by Gaussian Elimination, then we can re-arrange the equations in an order that would ensure that no row interchanges are required.

$\implies$ There is a rearrangement of the equations that permits Gaussian Elimination without row exchanges.

But WHY does the rearrangement of the equations break LU decomposition?

0.31 Permutation Matrix

An $n \times n$ permutation matrix P is obtained by rearranging the rows of the identity matrix. This gives a matrix with precisely one nonzero entry in each row and column. The nonzero entries are all 1's.

On the other hand, multiplying A on the right by P will exchange the second and third **columns** of A.

We will be using the following two properties of permutation matrices:

1. If $K_1, ..., K_n$ is a permutation of the integers $1, ..., n$, and the permutation matrix $P = [p_{ij}]$ is defined by
$$p_{ij} = \begin{cases} 1 & i = K_j \\ 0 & \text{otherwise} \end{cases}$$
Then, PA permutes the rows of A according to (BIGASS MATRIX MISSING HERE) data from chapter6-part2.pdf page 3.

2. If P is a permutation matrix, then P^{-1} exists and $P^{-1} = P^T$

Now our approach will be to left multiply the system

$$Ax = b$$

by the appropriate permutiation matrix P_1 so that the system

$$(PA)x = Pb$$

can be solved without row exchanges. Then PA can be factorized into

$$PA = LU$$

This tells us that

$$\begin{aligned} P^{-1}LUx &= b \\ LUx &= Pb \end{aligned}$$

which can be solved rapidly for x.

0.32 Speical Types of Matrices

Where can Gauassian Elimination be performe without row exchanges?

0.32.1 Strictly Diagonally Dominant Matrices

Def. An $n \times n$ matrix A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{j=1; j\neq i}^{n} |a_{ij}| \text{ for all } i = 1, \dots, n$$

Example

$$\begin{bmatrix} 3h & h & -h \\ 4 & 10 & 4 \\ 1 & 1 & -3 \end{bmatrix}$$

This matrix is strictly diagonally dominant when $h \neq 0$.

Thm. 1

A strictly diagonally dominant matrix A is nonsingular.

Thm. 2

Let A be a strictly diagonally dominant matrix. Then Gaussian Elimination can be performed on any linear system of the form $Ax = b$ to obtain its unique solution without row or column interchanges, and the computations are stable to the growth of roundoff error.

0.32.2 Symmetric, Positive Definite Matrices

Def. A matrix A is positive definite if

$$\forall x \neq 0 (x^T A x > 0)$$

Example

Find all values of α for which the matrix

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & \alpha \end{bmatrix}$$

is positive definite.

Ans.

$$\begin{aligned} x^T A x &= \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 1 \\ -1 & 1 & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= x_1^2 + x_2^2 + \alpha x_3^2 - 2x_1x_3 + 2x_2x_3 \\ &= (x_1 - x_3)^2 + (x_2 + x_3)^2 + (\alpha - 2)x_3^2 \end{aligned}$$

A is positive definite $\iff$ $\alpha > 2$.

Some necessary conditions for an $n \times n$ matrix to be positive definite u include:

(a) A is nonsingular

(b) $a_{ii} > 0$ for each $i = i...n$

(c) ???

(d) $(a_{ij})^2 < a_{ii}a_{jj}$ for each $i \neq j$

HOWEVER: These are not sufficient conditions for positive definiteness. they are necessary but not sufficient.

We would like necessary and sufficient conditions for a matrix to be positive definite.

0.32.3 Leading Principle Submatrices

A leading principle submatrix of a matrix A is a matrix of the form

$$A_k = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & & & \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{bmatrix}$$

for some $1 \leq k \leq n$.

Based on this definition, we have the following theorem:

Thm. 3

A symmetrix matrix A is positive definite if and only if each of its leading principle submatrices has a positive determinant.

As it turns out, we don't need to carry out row exchanges when Gaussian Elimination is used on a symmetric, positive definite matrix.

Thm. 4

A symmetric matrix A is positive definite if and only if Gaussian Elimination withour row exchanges can be performed on the linear system $Ax = b$ with all the pivot elements positive. Moreover, in this case, the computations are stable with respect to the growth of roundoff error.

Corollary: The matrix A is symmetric positive definite if and only if A can be factored in the form LDL^T where L is lower triangular with 1's on its diagonal and D is a diagonal matrix with positive diagonal entries.

A modification of the LU factorization algorithm can be made to factor a symmetric positive definite matrix into the form

$$A = LDL^T$$

This LDL^T factorization only requires $\frac{n^3}{6}+n^2-\frac{7n}{6}$ multiplications/divisions, and $\frac{n^3}{6}-\frac{n}{6}$ additions/subtractions. This is only half the number of operations as LU factorization.

A version of this algorithm can also be constructed for matrices that are symmetric but not positive definite.

Corollary 2: The matrix A is positive definite if and only if A can be factored int he form LL^T where L is lower triangular with nonzero diagonal entries. Once again, a modification of the LU factorization algorithm can be made. This method, called Choleski's Algorithm, factors a symmetric positive definite matrix into the form

$$A = LL^T$$

Choleski's Algorithm only requires $\frac{n^3}{6} + \frac{n^2}{2} - \frac{2n}{3}$ multiplications/divisions, and $\frac{n^3}{6} - \frac{n}{6}$ additions/subtractions, which is even less than the LDL^T factorization. However, for small n, Choleski's Algorithm may be slower because it requires n square roots to be computed.

0.32.4 Band Matrices

Another important class of matrices that arise in a wide variety of applications are band matrices. A band matrix is a matrix of the form

0.33 More Special Matrices

0.33.1 Band Matrices

Another important class of matrices that arise in a wide variety of applications are called band matrices. These matrices concentrate all their nonzero entries about the diagonal.

Definition

An $n \times n$ matrix is called a band matrix if there exist integers p and q such that $1 < p, q < n$ having the property that $a_{ij} = 0$ whenever $i + p \leq j$ or $j + q \leq i$.

The bandwidth of a band matrix is defined as $w = p + q - 1$. We subtract 1 from our count because we do not want to double count the diagonal.

We will focus on the important case of tridiagonal matrices, which are band matrices with $p = q = 2$, i.e., the matrix is tridiagonal if the nonzero entries are on the main diagonal and the diagonals above and below the main diagonal.

Suppose A can be factored into the triangular matrices L and U. Suppose that the matrices can be found in the form:

$$L = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & & \\ & & \ddots & \\ 0 & \dots & l_{n,n-1} & l_{nn} \end{bmatrix}$$

$$U = \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ & & \ddots & \\ 0 & 0 & \dots & u_{nn} \end{bmatrix}$$

The zero entries of A are automatically generated by LU. Multiplying $A = LU$, we also find the following conditions:

1.

$$a_{11} = l_{11}, \quad a_{i,i-1} = l_{i,i-1}, \quad i = 2, 3, \dots, n$$

2.
$$a_{ii} = l_{i,i-1}u_{i-1,i} + l_{ii}, \quad i = 2, 3, \ldots, n$$

3.
$$a_{i,i+1} = l_{ii}u_{i,i+1}, \quad i = 1, 2, \ldots, n-1$$

This system is straightforward to solve: (1) gives us l_{11} and the off-diagonal entries of L. (2) and (3) are used alternately to obtain the remaining entries of L and U. This solution technique is often referred to as **Crout Factorization**.

If we count up the number of operations, we find:

- $(5n-4)$ multiplications/divisions - $(3n-3)$ additions/subtractions

Crout Factorization can be applied to a matrix that is positive definite or one that is strictly diagonally dominant. See the text for another general case where it can be applied.

0.34 Iterative Techniques in Matrix Algebra

We are interested in solving large linear systems $Ax = b$.

Suppose the matrix A has a high ($> 99.9\%$) sparsity, i.e., most of the entries are zeros. We would like to take advantage of this spare structure to reduce the amount of computational work required. Unfortunately, Gaussin Elimination is often unable to take advantage of the sparse structure. For this reason, we consider <u>iterative techniques</u>.

0.35 Vector Norms

To estimate how well a particular iterative technique approximates the true solution, we will need some measurement of distance. This motivates the notion of the vector norm.

Def. A vector norm on $\mathbb{R}^n$ is a function $||\cdot||$ from $\mathbb{R}^n \to \mathbb{R}^n$ satisfying the following properties:

- $||x|| \geq 0$ for all $x \in \mathbb{R}^n$
- $||x|| = 0 \iff x = \mathbf{0}$
- $||\alpha x|| = |\alpha| ||x||$ for all $\alpha \in \mathbb{R}$ and $x \in \mathbb{R}^n$
- $||x + y|| \leq ||x|| + ||y||$ for all $x, y \in \mathbb{R}^n$

Def. 2. The l_2 or Euclidean norm of the vector x is given by

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

This represents the usual notion of distance.

Def. 3. The infinity or max norm of a vector x is given by

$$||x||_\infty = \max_{i=1}^{n} |x_i|$$

Def. 4. If $x, y \in \mathbb{R}^2$, then the l_2 distance between x and y is given by

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

and the l_∞ distance between x and y is given by

$$||x-y||_\infty = \max_{i=1}^{n} |x_i - y_i|.$$

0.36 Sequences?

Iterative techniques generate a sequence of vectors.

Def. A sequence $\left\{x^k\right\}_{k=1}^{\infty}$ of vectors in $\mathbb{R}^n$ is said to converge to x with respect to the norm $||\cdot||$ if, given any $\epsilon > 0$, there exists an integer $N('\epsilon)$ such that

$$||x^{(k)} - x|| < \epsilon \text{ for all } k \geq N$$

The notation $N('\epsilon)$ is used to emphasize that N is dependent on ϵ, however, N is not a function of ϵ.

0.36.1 Thm.

The sequence of vectors $\left\{x^k\right\}$ converges to x in $\mathbb{R}^n$ with respect to $||\cdot||_\infty$ if and only if $\lim_{k\to\infty} x_i^{(k)} = x_i$. for each i.

0.36.2 Thm. 2

For each $x \in \mathbb{R}^n$

$$||x||_\infty \leq ||x||_2 \leq \sqrt{n}||x||_\infty$$

This theorem relates the infinity norm to the Euclidean norm, which is very useful in the context of iterative techniques.

Example 1

Prove that $x^{(k)} = \left(\frac{1}{k}, 1+e^{1-k}, -\frac{2}{k^2}\right)^t$ is convergent w.r.t. $||\cdot||_2$.

We know $0 \leq ||x^{(k)} - x||_2 \leq \sqrt{3}||x^{(k)} - x||_\infty$

$$\lim_{k\to\infty} ||x^{(k)} - x||_\infty = 0 \implies \lim_{k\to\infty} ||x^{(k)} - x||_2 = 0$$

So, $\left\{x^{(k)}\right\}$ is convergent to x w.r.t. $||\cdot||_2$.

It can be shown that <u>all</u> norms on $\mathbb{R}^n$ are equivalent with respect to convergence.

i.e. If $||\cdot||_a$ and $||\cdot||_b$ are norms on $\mathbb{R}^n$, and $\left\{x^{(k)}\right\}_{k=1}^{\infty}$ has the limit x with respect to $||\cdot||_a$, then $\left\{x^{(k)}\right\}_{k=1}^{\infty}$ also has the limit x with respect to $||\cdot||_b$.

0.37 Matrix Norms

Def. A <u>matrix norm</u> on the set of all $n \times n$ matrices ($R^{n\times n}$) is a real-valued function $||\cdot||$ defined on this set satisfying for all $n \times n$ matries A and B and all real numbers α :

1. $||A|| \geq 0$
2. $||A|| = 0 \iff A = 0$
3. $||\alpha A|| = |\alpha|||A||$
4. $||A+B|| \leq ||A|| + ||B||$
5. $||AB|| \leq ||A||||B||$

Def. A distance between two $n \times n$ matrices A and B is

$$||A - B||$$

0.37.1 Thm.

If $||\cdot||$ is a vector norm on $\mathbb{R}^n$, then

$$||A|| = \max_{||x||=1} ||Ax||.$$

is a matrix norm.

This is called the natural or induced matrix norm associated with the vector norm.

The following result gives a bound on the value of $||Ax||$:

0.37.2 Thm.

For any vector $x \neq 0$, matrix A, and any natural norm $||\cdot||$, we have

$$||Ax|| \leq ||A|| \cdot ||x||.$$

Notes on the Infinity Norm

The infinity norm is defined as

$$||x||_\infty = \max_i |x_i|$$

So, it's the maximum absolute value of every entry in x.

Example: $x = \begin{pmatrix} 1 \\ -2 \\ 1.5 \end{pmatrix}$

$||x||_\infty = 2$

Because the largest element (in magnitude) is -2.

0.38 Computing the Infinity Norm and the 1-Norm

Computing the ∞-norm of a matrix is straightforward:

0.38.1 Thm.

If $A = (a_{ij})$ is a $n \times n$ matrix, then

$$||A||_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$

Example:

Find

$$\left|\left|\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}\right|\right|_\infty$$

$$\begin{aligned} \sum_{j=1}^{n} |a_{1j}| &= |2| + |-1| + |0| = 3 \\ \sum_{j=1}^{n} |a_{2j}| &= |-1| + |2| + |-1| = 4 \\ \sum_{j=1}^{n} |a_{3j}| &= |0| + |-1| + |2| = 3 \\ \implies \ ||A||_\infty &= \max\{3, 4, 3\} = 4 \end{aligned}$$

0.38.2 Visualizing the Infinity Norm

Images courtesy of ChatGPT. I got really confused by the concept of the infinity norm, so I asked the bot for help.

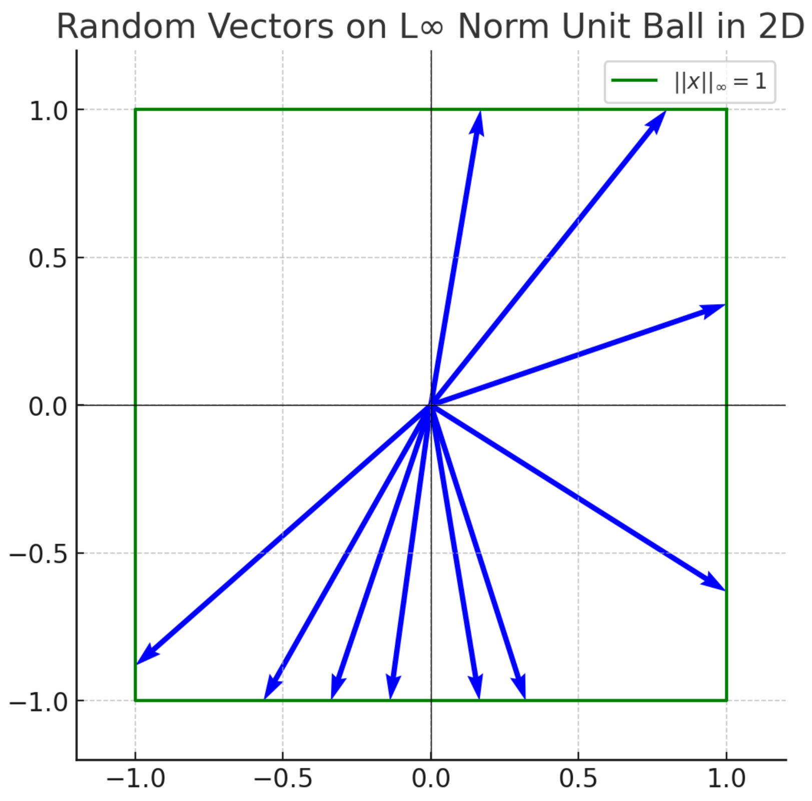

Figure 2: Visualizing the Infinity Norm

So the reason why the infinity norm, visualized this way, looks like a square, is because the equation $||x||_\infty = 1$ is equivalent to saying “the set of all vectors x such that the the largest component magnitude of the vector is 1”. Meaning this is the set of all vectors that have $x = 1$ or $y = 1$

0.39 Eigenvalues and Eigenvectors

To calculate the l_∞-norm of a matrix, we did not need to directly appy the definition. This is also true for the l_2-norm, however, we will need to introduce eigenvalues and eigenvectors to apply this technique.

First we will need the following definition:

Def. If A is a square matrix, the polynomial defined by

$$p(\lambda) = \det(A - \lambda I)$$

is called the characteristic polynomial of A. It is easily shown that p is an n^{th} degree polynomial.

Now we can introduce eigenvalues and eigenvectors.

Def. If p is the characteristic polynomial of the matrix A, the zeros of p are called eigenvalues, or characteristic values of A. If λ is an eigenvalue of A and $x \neq 0$ has the property that $(A - \lambda I)x = 0$ then x is called an eigenvector, or characteristic vector, of A corresponding to the eigenvalue λ.

The professor does not specifically discuss eigenvectors and eigenvalues in the context of Numerical Analysis, but they will be important for future problems. He also does not mention how to compute them.

However, he did suggest that we review how to compute them by hand and study the related theorems.

0.40 Finding the l_2-norm of a matrix

Def. The spectral radius $\rho(A)$ of a matrix A is defined as

$$\rho(A) = \max |\lambda| \text{ where } \lambda \text{ is an eigenvalue of } A$$

ex.

$$\rho(\begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}) = \max\{|3|, |3|, |1|\} = 3$$

And now, we can consider the following:

Thm. If A is a $n \times n$ matrix, then

1. $||A||_2 = \sqrt{\rho(A^T A)}$
2. $\rho(A) \leq ||A||$ for any natural norm $|| \cdot ||$

My editor broke in the middle so you should look at the Chapter 7 PDF notes for the proofs and examples. The end of this section is around page 19 of the PDF. (35.13)

When we use iterative matrix techniques, we want to know when powers of a matrix become small.

Def. We call an $n \times n$ matrix A convergent if

$$\lim_{k\to\infty} \left(A^k\right)_{ij} = 0, \text{ for all } i, j$$

ex. Consider $A = \begin{bmatrix} \frac{1}{2} & 0 \\ 16 & \frac{1}{2} \end{bmatrix}$

$$A^2 = \begin{bmatrix} \frac{1}{4} & 0 \\ 16 & \frac{1}{4} \end{bmatrix}$$
$$A^3 = \begin{bmatrix} \frac{1}{8} & 0 \\ 12 & \frac{1}{8} \end{bmatrix}$$
$$A^4 = \begin{bmatrix} \frac{1}{16} & 0 \\ 8 & \frac{1}{16} \end{bmatrix}$$
$$A^k = \begin{bmatrix} \frac{1}{2^k} & 0 \\ P_k & \frac{1}{2^k} \end{bmatrix}$$

where $P_k = \begin{cases} 16 & k = 1 \\ \frac{16}{2^{k-1}} + \frac{1}{2}P_{k-1} & k > 1. \end{cases}$

Since $\lim_{k\to\infty} P_k = 0$, we also know that $\lim_{k\to\infty} P_k = 0$. $\therefore$ A is a convergent matrix.

Notice that this convergent matrix has a spectral radius (see Lecture 12 notes , page 5) less than 1.

This generalizes:

Thm. The following statements are equivalent:

1. A is a convergent matrix.
2. $\rho(A) < 1$
3. $\lim_{n\to\infty} A^n x = 0$ for every x
4. $\lim_{n\to\infty} ||A^n|| = 0$ for all natural norms $||\cdot||$

Iterative techniques convert the system $Ax = b$ into an equivalent system of the form $x = Tx + c$ where T is a fixed matrix and c is a vector. An initialv vector $x^{(0)}$ is chosen, and then a sequence of approximate solution vectors is generated:

$$x^{(k)} = Tx^{(k-1)} + c$$

Iterative techniques are rarely used in very small systems (i.e. when n^3 is small). In these cases, iterative techniques may be slower since they require several iterations to obtain the desired accuracy.

IDEA: It is possible to "split" the matrix A :

$$\begin{aligned} Ax &= b \\ [M + (A - M]\, x &= b \\ Mx &= b + (M - A)x \\ x &= (I - M^{-1}A)x + M^{-1}b \end{aligned}$$

Iteration becomes

$$x^{(k+1)} = (I - M^{-1}A)x^{(k)} + M^{-1}b$$

We set $T \cong I - M^{-1}A$ (the amplification matrix) and $c \cong M^{-1}b$.

$$x^{(k+1)} = Tx^{(k)} + c$$

How do we choose M?
We want:

1. M easy to "invert"

2. M "close to A" in the sense that $\rho(T)$ is small.

ex. Let $M = D = \begin{bmatrix} a_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & a_{nn} \end{bmatrix}$
This gives the <u>Jacobi Iterative Method</u>.
In the text's notation,

$$A = D - L - U$$

Where D is diagonal, L is lower triangular, and U is upper triangular.

$$\begin{aligned} Ax &= b \\ (D-L-U)x &= b \\ Dx &= (L+U)x + b \\ x &= D^{-1}(L+U)x + D^{-1}b \end{aligned}$$

Which results in the iteration

$$x^{(k+1}) = D^{-1}(L+U)x^{(k)} + D^{-1}b$$

Let $T = D^{-1}(L+U)$ and $c = D^{-1}b$.

$$x^{(k+1)} = Tx^{(k)} + c$$

See example in the notes, there are too many matrices to type out in LaTeX. See "Chapter 7.pdf" page 27 (7-36.7)

Comments on Jacobi's Method

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b$$

1. The algorithm requirs $a_{ii} \neq 0$ for $i = 1, \ldots, n$ If one of the $a_{ii} = 0$, and the system is nonsingular, then a reordering of the equations can be performed so that no $a_{ii} = 0$.

2. To speed convergence, the equations should be arranged such that $|a_{ii}$ is as large as possible.

3. A possible stopping criterion is to iterate until $\frac{||x^{(k)}-x^{(k-1)}||}{||x^{(k-1)}||} \leq \epsilon$

If we write out Jacobi's Method

$$x^{(k+1)} = D^{-1}(L+U)x^{(k)} + D^{-1}b$$

we find that

$$x_i^{(k+1)} = \frac{\sum_{j=1;j\neq i}^{n}(-a_{ij}x_j^{(k)}) + b_i}{a_{ii}}$$

Notice that to compute $x_i^{(k+1)}$, the components $x_i^{(k)}$ are used. But, for $i > 1$, $x_1^{(k+1}, x_2^{(k+1)}, \ldots, x_n^{(k+1)}$ have already been computed and are likely better approximations to the actual solutions than

$$x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}$$

So it seems reasonable to compute with these most recently computed values.

i.e.:

$$x_i^{(k+1)} = \frac{-\sum_{j=1}^{i=1}(a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k)} + b_i}{a_{ii}}$$

This is called the Gauss-Seidel iterative technique, and it also has a matrix formulation with $M \cong (D-L)$:

$$\begin{aligned} Ax &= b \\ (D-L-U)x &= b \\ (D-L)x &= Ux + b \\ x &= (D-L)^{-1}Ux + (D-L)^{-1}b \end{aligned}$$

$\Longrightarrow$ iteration becomes

$$x^{(k+1)} = (D-L)^{-1}Ux^{(k)} + (D-L)^{-1}b$$

*Notice that $D-L$ is lower triangular. It is invertible $\iff$ each $a_{ii} \neq 0$

0.41 Solutions of Nonlinear Systems

The Basic Problem:

Find a root of $x \in \mathbb{R}$ of an equation of the form $f(x) = 0$ for a given continuous function f.

0.42 The Bisection Method

In most cases it is not really possible to solve analytically. We will consider iterative methods to approximate the solution. Our first method will be the Bisection Method. We must start with an interval $[a, b]$ with $f(a)$ and $f(b)$ of opposite signs.

By the intermediate value theorem, there exists a number c in (a, b) such that $f(c) = 0$.

Thm. If $f \in C[a, b]$ and k is any number between $f(a)$ and $f(b)$, then there exists a number c in (a, b) such that $f(c) = k$.

Now set $a_1 = a$ and $b_1 = b$ and let p_1 be the midpoint of $[a_1, b_1]$.

1. Compute the midpoint:

$$p_1 = \frac{1}{2}(a_1 + b_1)$$

2. If $f(p_1) = 0$, then we are done. Set $p = p_1$.

3. If $f(p_1)$ and $f(a_1)$ are of opposite signs, then there must exist a root $p \in (a_1, p_1)$ such that $f(p) = 0$. Set $a_2 = a_1$ and $b_2 = p_1$.

4. Otherwise, if $f(p_1)$ and $f(b_1)$ are of opposite signs, then there must exist a root $p \in (p_1, b_1)$ such that $f(p) = 0$. Set $a_2 = p_1$ and $b_2 = b_1$.

5. Reapply the process to the new interval $[a_2, b_2]$.

6. Once the stopping criteria are satisfied, set the midpoint of the interval as the estimate for the root.

0.42.1 Possible Stopping Criteria

1. $\dfrac{b_n - a_n}{2} <$ TOL $\quad$ <u>or</u> $\quad |p_n - p_{n-1}| <$ TOL

   GOOD: Ensures that the returned root value p_n is within tolerance of the exact value p

   GOOD: Easy error analysis

   BAD: Does not ensure that $f(p_n)$ is small.

   BAD: An absolute rather than relative measure of error.

2. $\dfrac{|p_n - p_{n-1}}{p_n} <$ TOL$, p_n \neq 0$

   Usually preferred over (1) if nothing is known about $f(\cdot)$ or p

3. $|f(p_n)| <$ TOL

   Ensures that $f(p_n)$ is small, but p_n may differ significantly from the true root p.

4. We can also carry out a fixed number of iterations $N-$ This is closely related to (1)

The best stopping criteria will depend on what is known about f and p and on the type of problem. It's often useful to use criteria 2 (relative error test) and criteria 4 (fixed number of steps) together.

Lemma If the spectral radius $\rho(T)$ satisfies $\rho(T) < 1$ then $(I - T)^{-1} = I + T + T^2 + \dots$

And we will prove the following theorem:

Thm. For any $x^{(0)} \in \mathbb{R}^n$, $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ the sequence defined by

$$x^{(k)} = Tx^{(k-1)} + c$$

converges to the unique solution of

$$x = Tx + c \text{ if and only if } \rho(T) < 1$$

Proof ($\Longleftarrow$).: assume $\rho(T) < 1$

$$\begin{aligned} x^{(k)} &= Tx^{(k-1)} + c \\ &= T(Tx^{(k-2)} + c) + c \\ &= T^2x^{(k-2)} + (T + I)c \\ &\vdots \\ &= T^kx^{(0)} + (T^{k-1} + \dots + T + I)c \end{aligned}$$

Since $\rho(T) < 1$, the matrix T is convergent and

$$\lim_{k\to\infty} T^kx^{(0)} = 0$$

Proof ($\Longrightarrow$).

HAS NOT BEEN WRITTEN DOWN YET

The *Lemma* implies that

$$\lim_{k\to\infty} x^{(k)} = \lim_{k\to\infty} T^kx^{(0)} + \lim_{k\to\infty}\left(\sum_{j=0}^{k-1} T^j\right) c = 0 + (1 - T)^{-1}c$$

$\Longrightarrow$ $\left\{x^{(k)}\right\}$ converges to the unique solution of $x = Tx + c$

i.e. $(I - T)x = c \implies x = (I - T)^{-1}c$

This allows us to derive some related results on the rates of convergence.

Corollary: If $||T|| < 1$ for any natural matrix norm and c is a given vector, then the sequence $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ defined by

$$x^{(k)} = Tx^{(k-1)} + c$$

converges for any $x^{(0)} \in \mathbb{R}^n$ to a vector $x \in \mathbb{R}^n$ and the following error bounds hold:

(i) $||x - x^{(k)}|| \leq ||T|^k||x^{(0)} - x||$

(ii) $||x - x^{(k)}|| \leq \frac{||T||^k}{-||T||}||x^{(1)} - x^{(0)}||$

Note, however, that $\rho(A) \leq ||A||$ for <u>any</u> natural norm. In practice,

$$||x - x^{(k)}|| \approx \rho(T)^k ||x^{(0)} - x||$$

so it is desirable to have $\rho(T)$ as small as possible.

Some results for Jacobi's and Gauss-Seidel methods:

Thm. If A is strictly diagonally dominant, then for any choice of $x^{(0)}$, both the Jacobi and Gauss-Seidel methods give sequences $\left\{x^{(k)}\right\}_{k=0}^{\infty}$ that converge to the unique solution of $Ax = b$.

No general results exist tot tell which of the two methods will converge more quickly, but the following result applies in a variety of examples:

Thm. Stein Rosenberge

If $a_{ij} \leq 0$ for each $i \neq j$ and $a_{ii} > 0$ for each $i = 1, 2, \ldots, n$, then exactly one of the following holds.

(a) $0 \leq \rho(T_g) < \rho(T_j) < 1$

0.43 Successive Over-Relaxation (SOR)

To define, suppose $\tilde{x}^{(k+1)}$ is the iterate from Gauss-Seidel using $x^{(k)}$ as the initial guess. The $(k+1)^{st}$ iterate of SOR is defined by

$$x^{(k+1)} = \omega\tilde{x}^{(k)} + (1 - \omega)x^{(k)}$$

where $1 < \omega < 2$. It can be difficult to select ω optimally. Indeed, the answer to this question is not known for general $n \times n$ linear systems.

However, we do have the following results:

Thm. (kahan): If $a_{ii} \neq 0$ for each i, then

$$\rho(T_{SOR}) \geq |\omega - 1|$$

$\implies$ SOR can converge only if $0 < \omega < 2$

Thm. (ostrowski-reich): If A is a positive definite matrix and $0 < \omega < 2$, then the SOR method converges for any choice of initial approximate vector $x^{(0)}$

Thm.. If A is positive definite and tridiagonal, then

$$\rho(T_g) = [\rho(T_j)]^2 < 1$$

and the optimal choice of ω for the SOR method is

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(T_j)^2}}$$

with this choice of ω, we have $\rho(T_{SOR}) = \omega - 1$

Bisection Method Example

He starts with an example on the bisection method. I will provide one later on.

Lowkey I missed notes from page 5.7 to 5.10 but I'll add them later. (page 5 of chapter 2 part 1 notes)

0.44 Fixed Point Iteration

We wish to find the roots of an equation $f(p) = 0$. We will be focusing on methods that iterate to find the root:

$$p_{n+1} = g(p_n)$$

We start by considering the fixed point problem.
Def. A fixed point p is the value of p such that $g(p) = p$.
We can form a fixed pt. problem from a root finding problem.

$$f(p) = 0$$

Find a fixed point problem.
e.g. set $g(x) = f(x) + x$
$g(p) = f(p) + p$
$p = g(p)$
there are many choices of g
Try $g(x) = f^3(x) + x$
Notice that fixed point problems and root finding problems are equivalent. (???)

0.44.1 Example

We are given $g(p) = p$. Formulate a root finding problem.

$$f(x) = g(x) - x$$

Now we have $f(p) = 0$. We now have a root finding problem.

$$f(p) = 0 \implies g(p) = p$$

i.e. f has a root p implies g has a fixed point p.

$$g(p) = p \implies f(p) = 0.$$

i.e. g has a fixed point p implies f has a root p.
There are many possible choices for g: example $g(x) - x = (f(x))^3$
Our ultimate goal is to find functions with fixed points.

0.45 Theorems

Existence and Uniqueness

$\star$If $g \in C[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$ then $g(x)$ has a fixed point in $[a,b]$.

$\star\star$ Suppose, in addition, that $g'(x)$ exists on (a,b) and that a positive constant $k < 1$ exists with

$$|g'(x)| \leq k < 1 \text{ for all } x \in (a,b).$$

then the fixed point in $[a,b]$ is unique

Proof. ($\star$): Existence
If $g(a) = a$ or $g(b) = b$ then g has a fixed point at an endpoint.
Suppose not, then it must be true that $g(a) > a$ and $g(b) < b$.
Define $h(x) = g(x) - x$. Then h is continuous on $[a,b]$ (adding two continuous functions yields a continuous function) and

$$h(a) = g(a) - a > 0 \text{ and } h(b) = g(b) - b < 0.$$

The **IVT** implies that there exists a $p \in (a,b)$ for which $h(p) = 0$
This $g(p) - p = 0 \implies p$ is a fixed point of g

Proof. ($\star\star$): Uniqueness
Suppose, in addition,

$$\forall x \in (a,b), |g'(x)| \leq k < 1.$$

and that p and q are both fixed points in $[a,b]$ with $p \neq q$.
By the **MVT**, a number c eixsts between p and q such that

$$\frac{g(p) - g(q)}{p - q} = g'(c).$$

then

$$\begin{aligned}|p-q| &= |g(p)-g(q)| \\ &= |g'(c)||p-q| \\ &\leq k|p-q| \\ &< |p-q|\end{aligned}$$

Which is a contradiction
This contradiction must come from the assumption that $p \neq q$
$\therefore p = q$ and the fixed point is unique.
We want to approximate the fixed point of a function g.
IDEA

- choose an initial approximation p_0
- generate a squence $\{p_n\}_{n=0}^{\infty}$ such that $p_n = g(p_{n-1}); n \geq 1$

If the sequence converges to p and g is continuous;

$$\begin{aligned}p &\equiv \lim_{n\to\infty} p_n \\ &= \lim_{n\to\infty} g(p_{n-1}) \\ &= g(\lim_{n\to\infty} p_{n-1}) \\ &= g(p)\end{aligned}$$

This gives the Fixed Point Algorithm:
See next notes package

We want to convert from $0 = f(x)$ to $x = g(x)$, which is to convert from a root finding problem to a fixed point problem.

One way to do this, very simply, is to just add x to both sides of the equation.

$$x^3 + 4x^2 - 10 = 0$$
$$x^3 + 4x^2 - 10 + x = x$$

In general, if your iterative method converges very quickly, you will not have a guarantee of convergence. Therefore, you should use a mix of methods to get a good initial guess and then quickly converge to the fixed point.

0.46 Convergence

Why do some methods converge and some diverge? Why do they converge with different rates?

Consider a simple example:

ex.

$$g(x) = ax + b.$$

We have

$$\begin{aligned} x_1 &= ax_0 + b \\ x_2 &= ax_1 + b = a(ax_0 + b) + b = a^2x_0 + (1+a)b \\ x_3 &= ax_2 + b = a^3x_0 + (a + a + a^2)b \end{aligned}$$

and by induction,

$$x_n = \begin{cases} a^n x_0 + \left(\frac{1-a^n}{1-a}\right)b & a \neq 1 \\ x_0 + nb & a = 1. \end{cases}.$$

$$\therefore \lim_{n\to\infty} x_n = \begin{cases} \frac{1}{1-a}b & |a| < 1 \\ x_0 & a = 1, b = 0. \end{cases}$$

No proper limit exists for all other values of a, b.

0.47 Fixed Point Theorem

When does a fixed point iteration converge? How quickly does it converge?

For this, we turn to the **fixed point theorem**.

Thm. Let $g \in C[a,b]$ and suppose $g(x) \in [a,b]$ for all $x \in [a,b]$.

Suppose, in addition, that g' exists on (a,b) and a positive constant $k < 1$ exists with

$$|g'(x)| \leq k \text{ for all } x \in (a,b).$$

Then for any number p_0 in (a,b), the sequence defined by

$$p_n = g(p_{n-1}) \qquad n \geq 1.$$

converges to the unique fixed point p in $[a,b]$

Proof. Our earlier **Thm.** (existence + uniqueness) tells us that a unique fixed point exists in $[a,b]$. Notice that g maps $[a,b]$ into itself, so the sequence $\{p_n\}_{n=0}^{\infty}$ is defined for all $n \geq 0$ and $p_n \in [a,b]$ for all n.

We may apply the **mean value theorem** to g to show that for any n

$$\begin{aligned} |p_n - p| &= |g(p_{n-1}) - g(p)| \\ &= |g'(c)||p_{n-1} - p| \\ &\leq k|p_{n-1} - p| \end{aligned}$$

Where $c \in (a,b)$. Applying the inequality inductively gives

$$somegarbageihaven'twrittendownyet.$$

Since $k < 1$, $\lim_{n\to\infty} |p_n - p| \leq \lim_{n\to\infty} k^n |p_0 - p| = 0$

$$\therefore \{p_n\}_{n=0}^{\infty} \text{ converges to } p.$$

This proof also gives us a natural bound for the error.

0.48 Newton's Method

(or Newton-Raphson Method)

One of the most powerful and well-known methods for solving a root-finding problem

$$f(x) = 0.$$

Pros: Much faster than bisection
Cons: Needs $f'(x)$, not guaranteed to converge
Want: $x = p$ s.t. $f(x) = 0$
Idea: Use slope as well as function values

0.48.1 Derivation (by Taylor's Thm.)

Want: $x = p$ s.t. $f(x) = 0$

Suppsoe $f \in C^2[a, b]$. Let $\overline{x} \in [a, b]$ be an approximation to p s.t. $f'(\overline{x}) \neq 0$ and $|\overline{x} - p|$ is sufficiently small. Then

$$f(x) = f(\overline{x}) + f'(\overline{x})(x - \overline{x}) + \frac{1}{2} f''(\xi(x))(x - \overline{x})^2 \qquad \xi \text{ lies between } x, \overline{x}.$$

Set $x = p$

$$0 = f(\overline{x}) + f'(\overline{x})(p - \overline{x}) + \frac{1}{2} f''(\xi(p))(p - \overline{x})^2.$$

$p - \overline{x}$ is very small $\implies$ $|p - \overline{x}|$ is even smaller. So we just drop the error term $\frac{1}{2} f''(\xi(p))(p - \overline{x})^2$

$$0 \approx f(\overline{x}) + f'(\overline{x})(p - \overline{x}).$$

Solve for p:

$$\tilde{p} = \overline{x} - \frac{f(\overline{x})}{f'(\overline{x})}.$$

Take $p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}$

As a stopping criterion, we might use

$$|p_n - p_{n-1}| \leq \mathit{TOLERANCE} \quad \epsilon.$$

Called "absolute error approximation".

We might also use "relative error approximation"

$$\frac{|p_n - p_{n-1}|}{|p_{n-1}|} \leq \epsilon.$$

(1) Newton's method fails:

$$f'(p_n) = 0.$$

$\implies$ method is not effective if f' is equal to zero at p. It will also not perform well if f' is close to 0.

Also we see in the derivation that $|p - \overline{x}|$ needs to be small, which implies we need a good initial guess.

ex.:

Use Newton's Method to compute the square root of a number R. We want to find the roots of $p^2 - R = 0$.

Let

$$\begin{aligned} f(x) &= x^2 - R \\ f'(x) &= 2x \end{aligned}$$

Newton's Method takes the form

$$\begin{aligned} p_n &= p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \\ &= p_{n-1} - \frac{p_{n-1}^2 - R}{2p_{n-1}} \\ &= \frac{1}{2}(p_{n-1} + \frac{R}{p_{n-1}}) \end{aligned}$$

Method is credited to Heron, a Greek Engineer circa 100BC -> 100AD.

Try $R - 2$:

$$\begin{aligned}
p_0 &= 2 \\
p_1 &= 1.5 \\
p_2 &= 1.416666 \\
p_3 &= 1.41425162 \\
p_4 &= 1.414211356 \qquad \text{12 digits correct}
\end{aligned}$$

Newton's method can be shown to converge under reasonable assumptions (smoothness of $f(\cdot)$, $f'(p) \neq 0$ and a good initial guess)

Thm. (Convergence):