Numerical Analysis Course Notes

Isaac Holt

February 7, 2023

Contents

1 Chebyshev Polynomials

Theorem 1.0.1. let $w_n(x) = (x - x_0)\dots(x - x_n)$ with distinct nodes $x_0, \dots, x_n$, $x_j \subset [-1, 1]$. Then the maximum of $|w_n(x)|$ on $[-1, 1]$ attains its smallest value (2^{-n}) iff x_j are the zeros of $T_{n+1}(x)$.

Proof. ($\Longleftarrow$): By construction $2^{-n}T_{n+1}(x)$ is a monic polynomial (highest power of x is 1) with $n + 1$ roots in $[-1, 1]$. Suppose $S_{n+1}(x) = (x - z_0)\dots(x - z_n)$ is another monic polynomial such that $\max |S_{n+1}(x)| < 2^{-n} = \max |2^{-n}T_{n+1}(x)|$. Let $q_n(x) := 2^{-n}T_{n+1}(x) - S_{n+1}(x)$. Then $q_n(x) \in P_n$ since the coefficient of x^{n+1} in $T_{n+1}(x)$ and $S_{n+1}(x)$ are both 1 and so cancel out.

Then $q_n(y_j) = 2^{-n}T_{n+1}(y_j) - S_{n+1}(y_j)$ (y_j are the extrema of $T_n(x)$). $|S_{n+1}(y_j)| < 1$ by hypothesis. Therefore $q_n(y_j) > 0$ if j is odd and < 0 otherwise.

Since we have $n + 2$ of y_j, q_n has at least $n + 1$ zeros. But since $q_n \in P_n$, we must have $q_n(x) = 0$. Therefore $S_{n+1}(x) = 2^{-n}T_{n+1}(x)$. □

Remark. To use this in $[a, b]$ instead of $[-1, 1]$, one simply maps $x_j \to a + (x_j + 1)\frac{b-a}{2}$.

Remark. Putting the above into Cauchy's error formula, we have

$$\sup |f(x) - p(x)| \leq 2^{-n}(\frac{b-a}{2})^{n+1} \frac{1}{(n+1)!}$$

Remark. We have by the above theorem, $\max |w_n(x)| = \max |(x - x_0)\dots(x - x_n) \geq 2^{-n}$ for any choice of $x_1, \dots, x_n$, $x_j \subset [-1, 1]$. So 2^{-n} is a lower bound for $|w_n(x)|$.

The upper bound is given by $\max |w_n(x)| \leq \epsilon |b - a|^n$.

2 Root Finding

2.1 Bracketing: Bisection

Given $f \in C^0([a,b])$ with $f(a)f(b) < 0$, repeat:

- let $(a_0, b_0) = (a, b)$
- let $m_n = \frac{1}{2}(a_n + b_n)$
- if $f(m_n)f(a_n) \geq 0$, set $(a_{n+1}, b_{n+1}) = (m_n, b_n)$
- otherwise, set $(a_{n+1}, b_{n+1}) = (a_n, m_n)$

$b_{n+1} - a_{n+1} = \frac{1}{2}(b_n - a_n)$. By the Intermediate Value Theorem, if $f(m_n) \neq 0$, for some $p \in (a_n, b_n)$, $f(p) = 0$.

$|p - m_n| \leq 2^{-(n+1)}(b-a)$

Remark. Each time, the width of the interval halves. In principle, we could get an approximation to any desired accuracy, but there are some caveats (e.g. with floating points).

2.2 Bracketing: False Position

Suppose we have $|f(b)| \ll |f(a)|$, then we would expect p to be closer to b than to a. Instead of $m_n = \frac{1}{2}(a_n + b_n)$, set

$$m_n = b_n - f(b_n)\frac{b_n - a_n}{f(b_n) - f(a_n)}$$

i.e. m_n is the x-intercept of the line from $(a_n, f(a_n))$ to $(b_n, f(b_n))$. This should sometimes give much faster approximation than bisection, but not always.

2.3 Aside: Continuity and Convergence

Definition 2.3.1. $f : I \to \mathbb{R}$ is continuous at $x \in I$ if for every $\epsilon > 0$, for some $\delta(x, \epsilon)$, $|y - x| < \delta \Rightarrow |f(x) - f(y)| < \epsilon$ for every $y \in B(x)$ ($B(x)$ is an open interval containing x).

Remark. In general, δ depends on ϵ and x. When δ is independent of x, f is uniformly continuous.

Definition 2.3.2. $f : I \to \mathbb{R}$ is Lipschitz continuous in I if for some $L > 0$, $|f(y) - f(x)| \leq L|y - x|$ for every $x \in I, y \in I$. In this case, $\delta = \epsilon/L$.

Remark. L (like δ above) is not unique. The smallest such L is called the Lipschitz constant of f in I.

Lemma 2.3.3.

1. If f is differentiable and I is compact, f is Lipschitz in I.
2. If f is Lipschitz, f is continuous.

Proof.

$$f(y) - f(x) = \int_x^y f'(x)ds$$

$$|f(y) - f(x)| = |\int_x^y f'(x)ds| \le \int_x^y |f'(x)|ds$$

$$\le \max_{s\in I} |f'(s)| \int_x^y ds = \max_{s\in I} |f'(s)||y-x|$$

We can take $L = \max_{s\in I} |f'(s)|$ □

Remark. The converses of 1. and 2. are false.

Remark.

- When f is continuous in I, we write $f \in C^0(I)$.
- When f is differentiable in I, we write $f \in C^1(I)$.
- When f is Lipschitz in I, we write $f \in C^{0,1}(I)$.
- We can then write $C^1(I) \subsetneq C^{0,1}(I) \subsetneq C^0(I)$.

Definition 2.3.4. A sequence (x_n) in $\mathbb{R}^d$ converges to x if for every $\epsilon > 0$, for some $N(\epsilon)$, for every $n \ge N(\epsilon)$, $|x_n - x| < \epsilon$.

This relies on us knowing x in the first place.

Definition 2.3.5. A sequence (x_n) is a Cauchy sequence if for every $\epsilon > 0$, for some $N(\epsilon)$, for every $m \ge N, n \ge N$, $|x_n - x_m| < \epsilon$.

Theorem 2.3.6. Let (x_n) be a Cauchy sequence in $\mathbb{R}^d$. Then (x_n) converges.

This is useful as it allows us to prove convergence without knowing x.

2.4 Fixed Point Iterations

We seek x such that $f(x) = 0$ for a function f. We rewrite this as

$$x = g(x)$$

We then seek to solve this equation by iterations:

1. pick some x_0
2. set $x_{n+1} = g(x_n)$

Theorem 2.4.1. (1d local convergence theorem): Let $g \in C'([a,b])$ have a fixed point $x_\star \in [a,b]$ $(g(x_\star) = x_\star)$ with $|g'(x_\star) < 1$. Then for x_0 sufficiently close to $x_\star$, the iteeration $x_{n+1} = g(x_n)$ converges to $x_\star$.

Proof. Let $g'(x_\star) = L \in (0,1)$ ($g'(x_\star) < 0$ is analogous). Since g' is continuous at $x_\star$, for every $L' \in (L,1)$, for some $\delta(L') > 0$, $g'(x) \le L' < 1$ for every $x \in (x_\star - \delta, x_\star + \delta) = B_\delta$, therefore for every $x \in B_\delta$, $y \in B_\delta$, $|g(x) - g(y)| \le \sup_{s\in B_\delta} |g'(s)||x-y| = L'|x-y|$ with $L' < 1$.

Let $x_\star \in B_\delta$, then $|g(x) - x_\star| = |g(x) - g(x_\star)| \le L'|x - x_\star|$ since $x_\star = g(x_\star)$. So $x - x_\star\delta$ as $x \in B_\delta$, so $|g(x) - x_\star| \le L'\delta \le \delta$, therefore $g(B_\delta) \subseteq B_\delta$. □

Remark. We do not need to know $x_\star$ to apply the 1d local convergence theorem, we just need to know that $|g'(x)| < 1$ for every $x \in I$ for some interval I.

2.5 Order of convergence

Order of convergence is a rough measure of how quickly $x_n \to x$. We mainly look at sequences arising from iterations with a nice RHS (so not bisection).

Definition 2.5.1. Let $x_n \to x_*$ and assume that $x_n \neq x_*$ for every $n \geq 0$. $x_n \to x_*$ with order at least $\alpha > 1$ if

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_*|}{|x_n - x_*|^\alpha} = \lambda < \textit{todo}$$

and with order $\alpha = 1$ if also $\lambda < 1$.

Example 2.5.2. $x_n = n^{-\beta}$, $\beta > 0$

$$\frac{|x_{n+1} - x_*|}{|x_n - x_*|} = (\frac{n}{n+1})^\beta \to 1$$

Example 2.5.3. $x_n = e^{-n}$

$$\frac{|x_{n+1} - x_*|}{|x_n - x_*|} = 1/e < 1$$

Example 2.5.4. $x_n = \frac{1}{n!}$

$$\frac{|x_{n+1} - x_*|}{|x_n - x_*|} = 1/(n+1) \to 0$$

The order of convergence of $x_n \to x_*$ is

$$\alpha = \sup\{\beta : \lim_{n\to\infty} \frac{|x_{n+1} - x_*|}{|x_n - x_*|^\beta} < \textit{todo}\}$$

for $\alpha > 1$.
For $\alpha = 1$, we also require that the limit < 1.
The convergence is linear if $\alpha = 1$, superlinear if $\alpha > 1$ and sublinear otherwise.

Remark. Order of convergence need not be an integer.

Remark. We need to know x_* in order to determine order of convergence.

Remark. Our definition is not comprehensive for general sequences.

Applying this to iterations:
$x_{n+1} - x_* = g(x_n) - g(x_*) = (x_n - x_*)g'(c_n)$ for some $c \in \operatorname{conv}\{x_n, x_*\}$ by the Mean Value Theorem.
Therefore

$$|x_{n+1} - x_*||x_n - x_*| = |g'(c_n)| \to g'(x_*)$$

We conclude that for $g \in C^2(I)$, the iteration $x_{n+1} = g(x_n)$ converges linearly if $g'(x_*) \neq 0$ and $|g'(x_*)| < 1$, and superlinearly otherwise.

Proposition 2.5.5. Let $g \in C^{N+1}(D)$ for some $D \subseteq \mathbb{R}$ and let $g(x_*) = x_*$, with x_* in the interior of D.

Then the iteration $x_{n+1} = g(x_n)$ converges to x_* for x_0 sufficiently close to x_* with order $N + 1$ iff $g'(x_*) = g''(x_*) = \cdots = g^{(N)}(x_*) = 0$ and $g^{(N+1)}(x_*) \neq 0$.

Proof. $x_{n+1}-x_* = g(x_n)-g(x_*) = g(x_*)+(x_n-x_*)g'(x_*)+\cdots+\frac{(x_n-x_*)^N}{N!}g^{(N)}(x_*)+\frac{(x_n-x_*)^{N+1}}{(N+1)!}g^{(N+1)}(c_n)-g(x_*) = \frac{(x_n-x_*)^{N+1}}{(N+1)!}g^{(N+1)}(c_n)$. Thus

$$|x_{n+1}-x_*||x_n-x_*|^{N+1} = \frac{|g^{(N+1)}(c_n)|}{(N+1)!} \to \frac{|g^{(N+1)}(x_*)|}{(N+1)!} < \mathit{todo}$$

□

2.6 Higher order iterative methods

We want to rearrange $f(x)=0$ to get faster convergence.

$x_{n+1} = g(x_n) = x_n + \phi(x_n)f(x_n)$ for some ϕ.

Using the above proposition, we need $g'(x_*)=0$.

$g'(x_*) = 1+\phi'(x_*)f(x_*)+\phi(x_*)f'(x_*) = 0$

So if $f'(x_*)\neq 0$, we take $\phi(x) = -\frac{1}{f'(x)}$.

This is the Newton-Raphson method:

$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.

Multiplication of a computer is parallelisable while division is not. So we can use Newton-Raphson to divide numbers with multiplication.

To compute $x_* = 1/b$ so $f(x_*) = \frac{1}{x_*}-b = 0$, so using Newton-Raphson, $x_{n+1} = x_n - \frac{{x_n}^{-1}-b}{-{x_n}^{-2}} = x_n(2-bx_n)$ which involes only multiplication and subtraction. When taking out the exponent, $x_0 \in \left[\frac{1}{2},1\right)$, and this converges quickly.

Remark. This only works with floating points, not integers. Floating point division is 5 times faster than integer division.

Remark. It can be difficult in practice to determine the interval/domain of convergence for Newton-Raphson. We should always perform a "sanity check" when using it.

Example 2.6.1. One advantage of iterative methods is that they also work (in principle) in higher dimensions.

Suppose $f:\mathbb{R}^2\to\mathbb{R}^2$ has a root at $\underline{p}=(p_1,p_2)$. Using Taylor expansion at the current point $\underline{x_n}$, derive Newton-Raphson (2D):

$$\underline{x_{n+1}} = \underline{x_n} - (Df)^{-1}\underline{f}(\underline{x_n})$$

We write $\underline{x_*}=\underline{p}$ such that $f_1(p_1,p_2)=f_2(p_1,p_2)=0$. Taylor-expanding at $\underline{x}$:

$$0 = f_1(p_1,p_2) = f_1(x_1,x_2)+(p_1-x_1)\frac{\partial f_1}{\partial x_1}(x_1,x_2)+(p_2-x_2)\frac{\partial f_1}{\partial x_2}(x_1,x_2)+O(|\underline{p}-\underline{x}|^2)$$

$$0 = f_2(p_1,p_2) = f_2(x_1,x_2)+(p_1-x_1)\frac{\partial f_2}{\partial x_1}(x_1,x_2)+(p_2-x_2)\frac{\partial f_2}{\partial x_2}(x_1,x_2)+O(|\underline{p}-\underline{x}|^2)$$

In matrix form:

$$(0,0) = (f_1(\underline{x}),f_2(\underline{x})) = (Df)(x_1,x_2)\cdot(x_1-p_1,x_2-p_2)+O(|\underline{p}-\underline{x}|^2)$$

Assuming that Df is invertible (equivalently, $f'(\underline{x})\neq 0$), we can multiply the equation by $(Df)^{-1}$ to get

$$(p_1, p_2) = (x_1, x_2) - (((Df)^{-1})(x_1, x_2)) \cdot (f_1(\underline{x}), f_2(\underline{x})) + O(|\underline{p} - \underline{x}|^2)$$

So

$$\underline{p} = \underline{x} - (((Df)^{-1})(x_1, x_2))\underline{f}(\underline{x}) + O(|\underline{p} - \underline{x}|^2)$$

We can use this to construct our iteration by replacing $\underline{x}$ with $\underline{x_n}$ and $\underline{p}$ with $\underline{x_{n+1}}$, and removing the $O(|\underline{p} - \underline{x}|^2)$.

2.7 Secant method

One disadvantage of Newton-Raphson is that we need the derivative, f'. If f is complicated or is itself computed numerically, we need to approximate f'. An alternative method is the secant method.

The secant method approximates f' with:

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

so the iteration becomes

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n)$$

Remark. This is a (scalar) two-step method: $x_{n+1} = g(x_n, x_{n-1})$, where x_n and x_{n-1} are needed to calculate x_{n+1}.

Theorem 2.7.1. Let $f \in C^2$ with $f(x_*) = 0$ and $f'(x_*) \neq 0$. Then the secant method is convergent with order $\alpha = \frac{1+\sqrt{5}}{2}$ for every $x_0 \neq x_1$ sufficiently close to x_*.

Proof. TODO: See video on Panopto □

Remark.

1. When implementing the secant method, one must be careful with floating point effects:

$$\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$$

   becomes very inaccurate as $x_n - x_{n-1} \to 0$.

2. $e_n := x_n - x_*$ alternates in sign if $g'(x_k) < 0$ and has the same sign if $g'(x_*) > 0$. From the proof of the method,

$$e_{n+1} = e_n e_{n-1} \frac{f''(\theta_n)}{f'(\phi_n)}$$

   where $\theta_n, \phi_n \in \operatorname{conv}\{x_{n-1}, x_n, x_{n+1}\}$.

   For $e_0 e_1 < 0$ and n sufficiently large, the error e_n follows the pattern $+, +, -$ or $-, -, +$.

3 Numerical differentiation

3.1 Forward/backward difference

$f'(x) = \lim_{h\to 0} \frac{f(x+h)-f(x)}{h} \approx \frac{f(x+h)-f(x)}{h}$ for some small $h \neq 0$.

More rigorously, we can use Taylor series:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(\xi)$$

where $\xi \in \mathrm{conv}\{x, x+h\}$. So

$$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{h}{2} f''(\xi)$$

The error $\frac{h}{2} f''(\xi)$ is of order $O(h)$.

For $h > 0$ this is called forward difference.

For $h < 0$, this is called backward difference.

3.2 Centred difference

$$f(x \pm h) = f(x) \pm hf'(x) + \frac{h^2}{2} f''(x) \pm \frac{h^3}{6} f^{(3)}(\xi_\pm)$$

Then

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \frac{h^2}{12}\left(f^{(3)}(\xi_+) + f^{(3)}(\xi_-)\right)$$

The error $\frac{h^2}{12}\left(f^{(3)}(\xi_+) + f^{(3)}(\xi_-)\right)$ is of order $O(h^2)$.

Remark. It is important to **not take h too small** when computing numerically. There is always a tradeoff between formal analytical accuracy and floating point errors. Formal analytical accuracy is better for smaller $|h|$, floating point errors are worse for smaller $|h|$.

Remark. Sometimes, we can only take $h < 0$ or $h > 0$ (not both), e.g. when solving differential equations numerically. If we have an ODE

$$\frac{du}{dt} = F(u)$$

Given $u(0)$ we want to solve for $u(t), t \geq 0$. We approximate $u(t)$ by a $u(t_n)$ with $t_n = n\delta t, n \in \mathbb{N}$, with $\delta t > 0$ small. Then

$$\frac{du}{dt} \approx \frac{u(t_n + \delta t) - u(t_n)}{\delta t} \approx F(u(t_n))$$

3.3 Richardson extrapolation

Let $f'(x) - \frac{f(x+h)-f(x)}{h} = f'(x) - R_h^{(1)}(x) = c_1(x)h + c_2(x)h^2 + c_3(x)h^3 + \cdots$ but suppose we cannot compute the c_k.

$R_{h/2}^{(1)}(x) = f'(x) - c_1\frac{h}{2} - c_2\frac{h^2}{4} - c_3\frac{h^3}{6} - \cdots$. We can use this to eleminate c_1:

$$2R_{h/2}^{(1)}(x) - R_h^{(1)}(x) = f'(x) - c_2' h^2 - c_3' h^3 - \cdots$$

So the error is of order $O(h^2)$. Then

$$f'(x) = R_h^{(2)}(x) + O(h^2)$$

where $R_h^{(2)} = 2R_{h/2}^{(1)}(x) - R_h^{(1)}(x)$.

Now, $R_{h/2}^{(2)}(x) = f'(x) - c_2' \frac{h^2}{4} - c_3' \frac{h^3}{8}$ and we use this to eliminate c_2':

$$4R_{h/2}^{(2)}(x) - R_h^{(2)}(x) = 3f'(x) + O(h^3)$$

So we set

$$R_h^{(3)} := \frac{2^2 R_{h/2}^{(2)}(x) - R_h^{(2)}(x)}{2^2 - 1} = f'(x) + O(h^3)$$

Remark.

1. We only need to specify the powers of h, not the coefficients c_k as long as they are **non-zero**. So when using centred difference to calculate $R_h^{(1)}$ then $R_h^{(2)}$, $R_h^{(3)}$ will be different.

2. Richardson extrapolation works with many other approximation methods involving a small parameter (not just differentiation).

3. There is no standard notation for this method.

4. Some series expansions have irregular/non-integer powers, e.g. the Airy function $\mathrm{Ai}(x)$ which is a solution of $\frac{d^2f}{dx^2} = xf(x)$.

4 Linear systems

Given $A \in M_n(\mathbb{R})$ and $\underline{x} \in \mathbb{R}^n$, we want to solve for $\underline{x}$:

$$A\underline{x} = \underline{b}$$

We want to minimise the error when computing this with floating points, and minimise the amount of computation for: large n, for different $\underline{b}$ (with the same A) and when A has particular forms.

Definition 4.0.1. The **transpose** of a square matrix A, A^T is defined as

$$(A^T)_{i,j} = A_{j,i}$$

Definition 4.0.2. A is **symmetric** if $A = A^T$.

Definition 4.0.3. A is **skew-symmetric** if $A = -A^T$.

Definition 4.0.4. A is **non-singular** if for every $\underline{b}$, for some $\underline{x}$, $A\underline{x} = \underline{b}$.

Definition 4.0.5. A is **positive definite** if $(A\underline{x}) \cdot \underline{x} > 0 \quad \underline{x} \neq \underline{0}$.

Definition 4.0.6. A is **positive semi-definite** if $(A\underline{x}) \cdot \underline{x} \geq 0 \quad \forall x$.

Lemma 4.0.7. If A is positive definite, then A is non-singular.

Proof. Omitted. □

Lemma 4.0.8. The converse of the above lemma is false.

Proof. A is non-singular $\Rightarrow -A$ is non-singular, but A is positive-definite $\Rightarrow -A$ is negative definite. □

Definition 4.0.9. A matrix A is **lower triangular** if $A_{i,j} = 0 \quad \forall j > i$.

Definition 4.0.10. A is **upper triangular** if $A_{i,j} = 0 \quad \forall j < i$.

Remark. We often write U for an upper triangular matrix and L for a lower triangular matrix.

Definition 4.0.11. To solve $Ux = b$, we can use **backward substitution**. Starting from the last equation,

$$\begin{aligned}
x_n &= \frac{1}{U_{n,n}} b_n \\
x_{n-1} &= \frac{1}{U_{n-1,n-1}}(b_{n-1} - U_{n-1,n}x_n) \\
&\vdots \\
x_j &= \frac{1}{U_j}\left(b_j - \sum_{i=j+1}^{n} U_{j,i}x_i\right)
\end{aligned}$$

Definition 4.0.12. Similarly, we can solve $L\underline{x} = \underline{b}$ for a lower triangular matrix L with forward substitution. Starting from the first equation,

$$x_1 = \frac{b_1}{L_{1,1}}$$

$$\vdots$$

$$x_j = \frac{1}{L_{j,j}} \left(b_j - \sum_{i=1}^{j-1} L_{j,i} x_k \right)$$

Remark. If $U_{j,j} = 0$ or $L_{j,j} = 0$ for some j, then this method doesn't work. This is expected, because in this case $\det U = 0$ (or $\det L = 0$).

Definition 4.0.13. If A is neither upper nor lower triangular, then we can transform A into a (usually) upper triangular matrix, by **Gaussian elimination**.

The following operations leave the solution $\underline{x}$ unchanged:

1. Swapping two rows
2. Adding a scalar multiple of a row to another row.

4.1 Computational complexity

Definition 4.1.1. For simplicity, we assume that each elementary floating point operation takes 1 unit of time, called 1 **flop**.

Remark. In practice, binary64 multiplication takes roughly 3 times as long as addition or multiplication, and binary64 division takes roughly 10 times as long as addition or subtraction.

Definition 4.1.2. Let $f(n) > 0$ and $g(n) > 0$ for large n. We write

$$f(n) \sim o(g(n)) \quad \text{if } \lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Definition 4.1.3. Let $f(n) > 0$ and $g(n) > 0$ for large n. We write

$$f(n) \sim O(g(n)) \quad \text{if } \limsup_{n\to\infty} \frac{f(n)}{g(n)} < \infty$$

Equivalently,

$$f(n) \sim O(g(n)) \quad \text{if } \exists C, \exists N, \forall n \leq N, f(n) \leq Cg(n)$$

Remark. From these definitions, we have

$$f(n) \sim o(g(n)) \implies f(n) \sim O(g(n))$$

Example 4.1.4.

- $100n^3 + 10^6 n^2 \sim O(n^3)$
- $n! + 10^{100} n^{100} \sim O(n!)$
- $n! \sim O(n^{n+\frac{1}{2}} e^{-n})$

Remark. In this module, we will be calculate the computational complexity to **leading order**, e.g. we distinguish $3n^2 + 5n$ form $30n^2$ but not from $3n^2 - 2n$.

Proposition 4.1.5. Backwards substitution on U where U is an $n \times n$ matrix is an $O(n^2)$ operation.

Proof.

- Computing $x_n = b_n / U_{n,n}$ takes 1 flop.
- Computing $x_{n-1} = \frac{(b_{n-1} - U_{n-1,n} x_n)}{U_{n-1,n-1}}$ takes 3 flops.
- $\vdots$
- Computing $x_1 = \frac{1}{U_1}\left(b_1 - \sum_{i=j+2}^{n} U_{1,i} x_i\right)$ takes $2n - 1$ flops.

So in total, backward substitution takes $1 + 3 + \cdots + 2n - 1 = n^2$ flops. □

Proposition 4.1.6. The number of flops needed for solving $Ax = b$ where A is an $n \times n$ matrix is

$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$$

Proof. To zero the first column for rows $i \in \{2, \ldots, n\}$:

- $\alpha_{i,1} = -A_{i,1}/A_{1,1}$ (1 flop).
- $A_{i,j} \to A_{i,j} + \alpha_{i,1} A_{1,j}$ for $j \in \{2, \ldots, n\}$ and $A_{i,1} = 0$ ($2(n-1)$ flops).
- $b_i \to b_i + \alpha i, 1 b_1$ (2 flops).

This is $2n + 1$ flops in total for each row, and this is done for $n - 1$ rows, so in total there are $(2n + 1)(n - 1)$ flops. For row 2, there are $(2n - 1)(n - 2)$ flops.

In general, to zero column k for rows $i \in \{k + 1, \ldots, n\}$:

- $\alpha_{i,k} = -A_{i,k}/A_{k,k}$ (1 flop).
- $A_{i,j} \to A_{i,j} + \alpha_{i,k} A_{k,j}$ for $j \in \{k + 1, \ldots, n\}$ ($2(n - k)$ flops).
- $b_i \to b_i + \alpha_{i,k} b_k$ (2 flops).

This is $2n - 2k + 3$ flops in total for each row, so for the $n - k$ rows, in total there are $(2n - 2k + 3)(n - k)$ flops.

So the total number of flops for Gaussian elimination is

$$\begin{aligned}
\sum_{k=1}^{n-1}(2n - 2k + 3)(n - k) &= \sum_{k=1}^{n-1} 2n^2 - 2kn + 3n - 2nk + 2k^2 - 3k \\
&= (n-1)(2n^2 + 3n) - (4n + 3)\sum_{k=1}^{n-1} k + 2\sum_{k=1}^{n-1} k^2 \\
&= (n-1)(2n^2 + 3n) - (4n + 3)\frac{n(n-1)}{2} \\
&\quad + \frac{1}{3}(n-1)n(2n-1) \\
&= \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n
\end{aligned}$$

Adding the n^2 flops from back substitution, in total to solve $Ax = b$, the number of flops needed is

$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n$$

$\square$

Remark. Gaussian elimination with (row) pivotting can transfomr every non-singular matrix into an upper-triangular matrix.

4.2 Pivoting and roundoffs

Definition 4.2.1. To reduce round-off errors, we define **row/partial pivoting**:

- When zeroing column k, look for $A_{j,k}$ with $j \in \{k, \dots, n\}$ with the largest absolute value, and swap row j with row k.

Remark. Multiplying a pivot row by a large constant does not improve reduction of round-off errors. The **ratio** of the pivot element to the other elements in the row is more important.

Remark. For better results, we can also perform column/full pivoting (so we swap rows and columns), but this is much more complicated.

4.3 LU decomposition

Lemma 4.3.1. Steps of the Gaussian elimination process can be written as matrix multiplication. During Gaussian elimination, let $A^{(k)}$ be the matrix during Gaussian elimination which has zeros below the diagonal in the first k columns.

$$A^{(s)} = F^{(s)} A^{(s-1)}$$

where

$$\begin{aligned} F^(_{i,j}s) &= \delta_{i,j} - f_i^{(s)} e_j^{(s)} \\ f_i^{(s)} &= (0, \dots, 0, A^{s-1}_{s+1,s}/A^{(s-1)}_{s,s}, \dots, A^{(s-1)}_{n,s}/A^{(s-1)}_{s,s}) \\ e_j^{(s)} &= \delta_{s,j} \end{aligned}$$

Proof. To zero column s below the leading diagonal, we do

$$A^{(s)}_{i,j} = A^{(s-1)}_{i,j} - \frac{A^{(s-1)}_{i,s}}{A^{(s-1)}_{s,s}} A^{(s-1)}_{s,j}$$

for $i \in \{s+1, \dots, n\}$ and $j \in \{s, \dots, n\}$.

$$\begin{aligned} A^{(s)}_{i,j} &= \sum_k \left(\delta_{i,k} A^{(s-1)}_{k,j} - \frac{A^{(s-1)}_{i,s}}{A^{(s-1)}_{s,s}} \delta_{s,k} A^{(s-1)}_{k,j} \right) \\ &= \sum_k \left(\delta_{i,k} - \frac{A^{(s-1)}_{i,s}}{A^{(s-1)}_{s,s}} \cdot \delta_{s,k} \right) A^{(s-1)}_{k,j} \end{aligned}$$

$\square$

Definition 4.3.2. For vectors u and v, the **tensor product** of u and v is a matrix defined by

$$(u \otimes v)_{i,j} = u_i v_j$$

Definition 4.3.3. A **Frobenius matrix** of index s is a unit lower triangular matrix whose only (possibly) non-zero elements are subdiagonal elements in column s (other than the leading diagonal, which is all 1s).

Lemma 4.3.4. Let $F^{(s)} = I - f^{(s)} \otimes e^{(s)}$. Then its inverse is

$$G^{(s)} = I + f^{(s)} \otimes e^{(s)}$$

Proof.

$$\begin{aligned}
\sum_k F^{(s)}_{i,k} G^{(s)}_{k,j} &= \sum_k \left(\delta_{i,k} - f^{(s)}_i e^{(s)}_k\right) \left(\delta_{k,j} + f^{(s)}_k e^{(s)}_j\right) \\
&= \sum_k \left(\delta_{i,k} - f^{(s)}_i \delta_{s,k}\right) \left(\delta_{k,j} + f^{(s)}_k \delta_{s,j}\right) \\
&= \sum_k \left(\delta_{i,j} + f^{(s)}_i \delta_{s,j} - f^{(s)}_i \delta_{s,j} - f^{(s)}_i f^{(s)}_k \delta_{s,k} \delta_{s,j}\right) \\
&= \delta_{i,j} - f^{(s)}_i f^{(s)}_s \delta_{s,j}
\end{aligned}$$

But $f^{(s)}_s = 0$ which completes the proof. □

Lemma 4.3.5. Let $G^{(s)} = I + f^{(s)} \otimes e^{(s)}$ as above. Then for every $s \in \{1, \ldots, n-1\}$,

$$G^{(1)} G^{(2)} \cdots G^{(s)} = I + \sum_{r=1}^{s} f^{(r)} \otimes e^{(r)}$$

Proof. Use induction on s.

- For $s = 1$, $G^{(1)} = I + f^{(1)} \otimes e^{(1)}$ trivially.
- Assume the statement is true for some s.
- For $s + 1$,

$$\begin{aligned}
(G^{(1)} \cdots G^{(s)} G^{(s+1)})_{i,j} &= \sum_k (G^{(1)} \cdots G^{(s)})_{i,k} G^{(s+1)}_{k,j} \\
&= \left(\sum_k \left(\delta_{i,k} \sum_{r=1}^{s} f^{(r)}_i \delta_{r,k}\right)\right) \left(\delta_{k,j} + f^{(s+1)}_k \delta_{s+1,j}\right) \\
&= \delta_{i,j} + \sum_{r=1}^{s} f^{(r)}_i \delta_{r,j} + f^{(s+1)}_i \delta_{s+1,j} \\
&\quad + \sum_k \left(f^{(s+1)}_k \delta_{s+1,j} \sum_{r=1}^{s} f^{(r)}_i \delta_{r,k}\right) \\
&= \delta_{i,j} + \sum_{r=1}^{s+1} f^{(r)}_i \delta_{r,j}
\end{aligned}$$

  which completes the induction.

□

Theorem 4.3.6. Suppose that A can be reduced to an upper-triangular matrix U by Gaussian elimination without pivoting. During Gaussian elimination, let $A^{(k)}$ be the matrix during Gaussian elimination which has zeros below the diagonal in the first k columns. Then there is a **unit** lower triangular matrix L such that

$$A = LU$$

Also, the subdiagonal elements of L are the coefficients used in the reduction of A:

$$\forall i > j, L_{i,j} = {A^{(j-1)}}_{i,j} / {A^{(j-1)}}_{j,j}$$

Proof. We write the Gaussian elimination process as

$$\begin{aligned} U = A^{(n-1)} &= F^{(n-1)} A^{(n-2)} \\ &= F^{(n-1)} F^{(n-2)} A^{(n-3)} \\ &= F^{(n-1)} F^{(n-2)} \cdots F^{(1)} A^{(0)} \end{aligned}$$

Now left-multiplying by $L = G^{(1)} \cdots G^{(n-1)}$:

$$LU = G^{(1)} \cdots G^{(n-1)} F^{(n-1)} F^{(n-2)} \cdots F^{(1)} A^{(0)}$$

But $G^{(k)} F^{(k)} = I$ by Lemma 4.3.4 so this simplifies to $LU = A$. □

Theorem 4.3.7. Every non-singular square matrix A can be written as

$$PA = LU$$

where L and U are lower and upper triangular matrices and P is a permutation matrix (exactly one non-zero element in each row and column).

Proof. Too difficult.batyter □

Remark. Once we have L and U, we can solve $Ax = LUx = b$ with

1. Solve $Ly = b$ for y using forward substitution.

2. Solve $Ux = y$ for x using backward substitution.

Example 4.3.8. $A = [[1, 2, 3, 6], [2, 8, 6, 5], [-4, -8, 0, 0], [0, 12, 9, -6]]$. Then $A^{(1)} =$
$[[1, 2, 3, 6], [0, 4, 0, -7], [0, 0, 12, 24], [0, 12, 9, -6]]$, $A^{(2)} = [[1, 2, 3, 6], [0, 4, 0, -7], [0, 0, 12, 24], [0, 0, 9, 15]]$
$[[1, 2, 3, 6], [0, 40, 0, -7], [0, 0, 12, 24], [0, 0, 0, -3]]$.

Then $L = [[1, 0, 0, 0], [L_{2,1}, 1, 0, 0], [L_{3,1}, L_{3,2}, 1, 0], [L_{4,1}, L_{4,2}, L_{4,3}, 1]] = [[1, 0, 0, 0], [2, 1, 0, 0], [-4, 0,$

Then if $b = [1, 2, 4, 4]$, solving $Ly = b$ with forward substitution gives $y = [1, 0, 8, -2]$.

Then solving $Ux = y$ for x gives $x = [-\frac{10}{3}, \frac{7}{6}, -\frac{2}{3}, \frac{2}{3}]$.

Remark. Comparing LU decomposition with Gaussian elimination and back substitution, LU decomposition involves less work when solving for different values of b (same values of A). Gaussian elimination and back substitution is $O(n^3)$ for each value of b. LU decomposition is $O(n^3)$ for the first value of b, but then solving for different values of b afterwards is $O(n^2)$.

Remark. LU decomposition is less prone to round off errors than computing A^{-1} and is less computationally expensive for some values of A:

- When A is tridiagonal, L and U are as well, but A^{-1} generally isn't.
- When A is sparse (most elements are 0), L and U are also sparse, but A^{-1} generally isn't.

Proposition 4.3.9. If L is an $n \times n$ lower triangular, invertible matrix, then L^{-1} is also lower triangular.

Proof. Let $M = L^{-1}$, then $LM = I$. Since L^{-1} exists, $L_{i,i} \neq 0$ and $M_{i,i} \neq 0$ $\forall 1 \leq i \leq n$.

We have $I_{i,j} = \delta_{i,j}$ for every i, j. Then $0 = I_{1,j} = \delta_{1,j} = L_{1,1}M_{1,j} \Rightarrow M_{1,j} = 0$ $\forall j > 1$ (since $L_{1,1} \neq 0$).

Now for row 2, $1 = \delta_{2,2} = L_{2,1}M_{1,2} + L_{2,2}M_{2,2} + L_{2,3}M_{3,2} + \cdots$ but $L_{2,j} = 0 \forall j > 2$ since L is lower triangular so $1 = L_{2,1}M_{1,2} + L_{2,2}M_{2,2}$. Similarly, $0 = \delta_{2,3} = L_{2,1}M_{1,3} + L_{2,2}M_{2,3} + L_{2,3}M_{3,3} + \cdots = L_{2,1}M_{1,3} + L_{2,2}M_{2,3} = L_{2,2}M_{2,3} \Rightarrow M_{2,3} = 0$. Generally,

$$\delta_{2,j} = \sum_{k=1}^{n} L_{2,k}M_{k,j} = L_{2,1}M_{1,j} + L_{2,2}M_{2,j} + L_{2,3}M_{3,j} + \cdots = L_{2,1}M_{1,j} + L_{2,2}M_{2,j}$$

But for $j > 2$, $M_{1,2}, \ldots, M_{1,n} = 0$. Since $L_{2,2} \neq 0$, $M_{2,j} = 0$ $\forall j > 2$.

This process continues for the rest of the rows by induction: assume that $M_{i,j} = 0$ $\forall i \in \{1, \ldots, s-1\}, j > i$. Now we show that this holds for $i \in \{1, \ldots, s\}$.

$$\begin{aligned}\delta_{s,j} &= \sum_{k=1}^{n} L_{s,k}M_{k,j} \\ &= \sum_{k=1}^{s} L_{s,k}M_{k,j}\end{aligned}$$

For $j > s$,

$$\begin{aligned}\delta_{s,j} &= \sum_{k=1}^{s} L_{s,k}M_{k,j} \\ &= L_{s,1}M_{1,j} + \cdots + L_{s,s-1}M_{s-1,j} + L_{s,s}M_{s,j}\end{aligned}$$

But by assumption, $L_{s,1}M_{1,j} + \cdots + L_{s,s-1}M_{s-1,j} = 0$, so $\delta_{s,j} = L_{s,s}M_{s,j} = 0 \Rightarrow M_{s,j} = 0$ as $L_{s,s} \neq 0$. □

4.4 Normed vector spaces

Definition 4.4.1. Given a vector space V, a **norm** $||.|| : V \to \mathbb{R}_+$ satisfies the following properties:

1. $||x + y|| \leq ||x|| + ||y|| \quad \forall x, y \in V$.
2. $||c \cdot x|| = |c|||x|| \quad \forall x \in V, c \in \mathbb{C}$.
3. $||x|| = 0 \iff x = 0$ (if this is not satisfied, the norm is called a **seminorm**).

Example 4.4.2. Some examples of norms:

1. The L_2 norm,

$$||x||_2 := \left(\sum_{k=1}^{n} |x_k|^2 \right)^{1/2}$$

2. The L_1 norm,
$$||x||_1 := \sum_{k=1}^{n} |x_k|$$
3. $||x||_\infty := \max_{k\in\{1,\ldots,n\}} |x_k|$
4. For $p \in [1,\infty)$,
$$||x||_p := \left(\sum_{k=1}^{n} |x_k|^p\right)^{1/p}$$

Definition 4.4.3. Let $\tilde{V}$ be the set of $n \times n$ matrices over $\mathbb{R}$, then
$$||A||_{\text{row}} := \max_i \sum_{j=1}^{n} |A_{i,j}|$$
$$||A||_{\text{col}} := \max_j \sum_{i=1}^{n} |A_{i,j}|$$
$$||A||_{\text{Frob}} := \left(\sum_{i,j} |A_{i,j}|^2\right)^{1/2}$$
are the **row-sum norm**, the **column-sum norm**, and the **Frobenius matrix norm**, respectively.

Definition 4.4.4. A matrix norm $||.||$ on $\tilde{V}$ is **submultiplicative** if
$$\forall A, B \in \tilde{V}, \quad ||AB|| \leq ||A|| \cdot ||B||$$

Definition 4.4.5. Given a vector norm $||.||_{\text{vec}} : V \to \mathbb{R}_+$, the **matrix norm** induced by $||.||_{\text{vec}}$ is defined as
$$||A||_{\text{mat}} = \sup_{x\neq 0} \frac{||Ax||_{\text{vec}}}{||x||_{\text{vec}}} = \max_{||x||_{\text{vec}}=1} ||Ax||_{\text{vec}}$$

Remark. Often, we omit vec or mat when the meaning of the norm is clear from context.

Example 4.4.6. By linearity of vector norms,
$$||Ax||_\star \leq ||A||_\star ||x||_\star$$
where $|||A||_\star$ is the matrix norm induced by the vector norm $||.||_\star$.

Example 4.4.7. Let $V = \mathbb{R}^2$ and $||.|| = ||.||_2$. Compute $||A||_2$ where
$$A = \begin{pmatrix} 3 & 1 \\ -5 & -1 \end{pmatrix}$$

By definition,
$$\begin{aligned}
||A||_2 &= \max_{||x||_2=1} ||Ax||_2 \\
&= \max_{\theta\in[0,2\pi]} ||A \cdot (\cos(\theta), \sin(\theta))||_2 \\
&= \max_{\theta\in[0,2\pi]} ||(3\cos(\theta) + \sin(\theta), -5\cos(\theta) - \sin(\theta))||_2
\end{aligned}$$

Let

$$\begin{aligned} f(\theta) &= (3\cos(\theta) + \sin(\theta))^2 + (5\cos(\theta)) + \sin(\theta))^2 \\ &= 34\cos(\theta)^2 + 2\sin(\theta)^2 + 16\cos(\theta)\sin(\theta) \\ &= 16\cos(2\theta) + 8\sin(2\theta) + 18 \end{aligned}$$

so $f'(\theta) = -32\sin(2\theta) + 16\cos(2\theta)$, then $f'(\theta) = 0 \Rightarrow \tan(2\theta) = 1/2$ which gives

$$\max_\theta f(\theta) = 18 + \frac{40}{\sqrt{5}}$$

which gives

$$||A||_2 = \sqrt{18 + \frac{40}{\sqrt{5}}}$$

Definition 4.4.8. A matrix A is **normal** if $AA^T = A^TA$.

Theorem 4.4.9. If a matrix A is normal, then A is diagonalisable.

Proof. Too long. □

Theorem 4.4.10. If A is symmetric, then its eigenvalues are real, and

$$||A||_2 = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } A\}$$

For non-symmetric A,

$$||A||_2 = \max\{|\lambda| : \lambda \text{ is an eigenvalue of } AA^T\}$$

Proof. Too long. □

Proposition 4.4.11. The vector 1-norm induces the matrix column-sum norm.

Proof. $||x||_1 = \sum_i |x_i|$ and $||A||_{\text{col}} = \max_j \sum_i |A_{i,j}|$.

$$\begin{aligned} ||Ax||_1 &= \sum_i \left| \sum_j A_{i,j} \right| \\ &\leq \sum_i \sum_j |A_{i,j} x_j| \\ &= \sum_i \sum_j |A_{i,j}||x_j| = \sum_j \left(\sum_i |A_{i,j}| \right) |x_j| \\ &\leq \left(\max_j \sum_i |A_{i,j}| \right) \sum_j |x_j| = ||A||_{\text{col}} ||x||_1 \end{aligned}$$

Hence,

$$||A||_1 \leq ||A||_{\text{col}}$$

and also

$$\begin{aligned} ||A||_1 &= \max_{||x||_1 = 1} ||Ax||_1 \\ &\geq \max_k ||Ae^{(k)}|| \end{aligned}$$

where $e_j^{(k)} = \delta_{j,k}$. So

$$\begin{aligned} ||A||_1 &\geq \max_k \sum_i \left| \sum_j A_{i,j}\delta_{j,k} \right| \\ &= \max_k \sum_i |A_{i,k}| = ||A||_{\text{col}} \end{aligned}$$

Hence $||A||_1 = ||A||_{\text{col}}$. $\square$

Example 4.4.12. Let

$$A = \begin{bmatrix} 3 & 1 & -4 \\ 1 & -5 & 9 \\ -2 & 6 & 5 \end{bmatrix}$$

so $||A||_{\text{col}} = \max\{3+1+2, 1+5+6, 4+9+5\} = 18$. The max is attained at column 3. Let $\underline{x} = (0,0,1)$ then $A\underline{x} = (-4,9,5)$ and $||A\underline{x}||_1 = 4+9+5 = 18$.

Proposition 4.4.13. The vector ∞-norm induces the matrix row-sum norm.

Proof. We want to show that

$$\max_{||x||_\infty = 1} ||Ax||_\infty = \max_i \sum_j |A_{i,j}|$$

First, we show $||A||_\infty \leq ||A||_{\text{row}}$ for every A and x:

$$\begin{aligned} ||Ax||_\infty &= \max_i \left| \sum_j A_{i,j}x_j \right| \\ &\leq \max_i \sum_j |A_{i,j}||x_j| \\ &\leq \max_i \left\{ \max_k |x_k| \sum_j |A_{i,j}| \right\} \\ &= \max_k |x_k| \max_i \sum_j |A_{i,j}| \\ &= ||x||_\infty ||A||_{\text{row}} \end{aligned}$$

Assuming that $x \neq 0$, we divide by $||x||_\infty$ and take the maximum:

$$||A||_\infty = \max_{||x||_\infty} \frac{||Ax||_\infty}{||x||_\infty} \leq \max_{||x||_\infty \neq 0} ||A||_{\text{row}} = ||A||_{\text{row}}$$

To show equality, for every A, we need to find an x that gives equality in the inequalities above. By linearity, we can take $||x||_\infty = 1$. We want to find an x such that

$$\max_k \left| \sum_j A_{k,j}x_j \right| = \max_i \sum_j |A_{i,j}|$$

Let i^* be an i that realises the maximum. Then let

$$x_j = \text{sgn}(A_{i^*,j})$$

Let $k = i^*$, then

$$\sum_j A_{i^*,j}x_j = \sum_j A_{i^*,j} \cdot \text{sgn}(A_{i^*,j}) = \sum_j |A_{i^*,j}| = \max_i \sum_j |A_{i,j}|$$

$\square$

Remark.

$$||A|| = \sup_{y\neq 0} \frac{||Ay||}{||x||} \quad \forall x \neq 0$$

Therefore

$$||Ax|| \leq ||A|| \cdot ||x|| \quad \forall x$$

Example 4.4.14. (Problems class) Show the equivalence of the induced norm definitions.

$$\begin{aligned}
\sup_{x\neq 0} \frac{||Ax||}{||x||} &= \sup_{x\neq 0} \frac{||Ax||/||x||}{||x||/||x||} \\
&= \sup_{x\neq 0} \frac{||A(x/||x||)||}{||(x/||x||)||} \\
&= \sup_{x\neq 0} ||A(x/||x||)|| \\
&= \max_{||y||=1} ||Ay||
\end{aligned}$$

Example 4.4.15. (Problems class) Show that $||AB|| \leq ||A|| \cdot ||B||$ for all matrices A, B and every induced norm $||\cdot||$.

$$\begin{aligned}
\sup_{x\neq 0, Bx\neq 0} \frac{||ABx||}{||x||} &= \sup_{x\neq 0, Bx\neq 0} \frac{||ABx||}{||Bx||}\frac{||Bx||}{||x||} \\
&\leq \sup_{Bx\neq 0} \frac{||ABx||}{||Bx||} \cdot \sup_{x\neq 0} \frac{||Bx||}{||x||} \\
&= ||A|| \cdot ||B||
\end{aligned}$$

When $Bx = 0$,

$$0 = ||ABx|| = ||A|| \cdot ||Bx|| = 0$$

so we can remove the condition $Bx \neq 0$ on the sups in the equation above.

4.5 Errors and condition numbers

In the case where we want to solve $Ax = b$ for x, but with some floating po0int error in b, we have $b + \delta b$ instead of b, where δb is unknown but bounded (small). So we have

$$A(x + \delta x) = b + \delta b$$

It is not always the case that δx is also small. We want to find bounds for δx in terms of δb. We have that $A\delta x = \delta b$. Assuming A^{-1} exists, $\delta x = A^{-1}\delta b$. Using any norm $||\cdot||_*$,

$$||\delta x||_* = ||A^{-1}\delta b||_* \leq ||A^{-1}||_* ||\delta b||_*$$

Assuming $x \neq 0$, divide by $||x||_*$:

$$\frac{||\delta x||_*}{||x||_*} \leq ||A^{-1}|| \frac{||\delta b||_*}{||x||_*}$$

Now since $Ax = b$,

$$||b||_* = ||Ax||_* \leq ||A||_* ||x||_* \implies \frac{1}{||x||_*} \leq ||A||_* \frac{1}{||b||_*}$$

Therefore,

$$\frac{||\delta x||_*}{||x||_*} \leq ||A||_* ||A^{-1}||_* \frac{||\delta b||_*}{||b||_*}$$

This means the relative error in x is less than a constant multiplied by the relative error in b.

Definition 4.5.1. For any norm $||\cdot||_*$, we define

$$\kappa_*(A) := ||A||_* ||A^{-1}||_*$$

which is called the ***-condition** number of the matrix A.

Example 4.5.2. (Problems class) Prove that $\kappa(A) \geq 1$ for every non-singular matrix A and for every induced matrix norm.

$$1 = ||I|| = ||AA^{-1}|| \leq ||A|| \cdot ||A^{-1}||$$

by Example 4.4.15.

Example 4.5.3. (Problems class) Prove that

$$||x||_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

is a norm for every $p \in [1, \infty)$.

$$\begin{aligned}
(||x+y||_p)^p &= \sum_i |x_i + y_i|^p \\
&= \sum_i |x_i + y_i| \cdot |x_i + y_i|^{p-1} \\
&\leq \sum_i (|x_i| + |y_i|)|x_i + y_i|^{p-1} \\
&= \sum_i |x_i| \cdot |x_i + y_i|^{p-1} + \cdots \\
&\leq \left(\sum_i |x_i|^r \right)^{1/r} \left(\sum_j |x_j + y_j|^{q(p-1)} \right)^{1/q} + \cdots
\end{aligned}$$

by the Holder inequality (TODO: define this inequality). Let $r = p, q = 1/(1-1/p) = p/(p-1)$. Then

$$\begin{aligned}
(||x+y||_p)^p &\leq \left(\sum_i |x_i|^p \right)^{1/p} \left(\sum_j |x_j + y_j|^{(p-1)p/(p-1)} \right)^{(p-1)/p} + \cdots \\
&= ||x||_p (||x+y||)_p^{p-1} + ||y||_p (||x+y||)_p^{p-1}
\end{aligned}$$

4.6 L^2 norm, eigenvalues and diagonalisability

Theorem 4.6.1. Let $\{\lambda_j\}$ be the eigenvalues of a matrix A. Then

$$\max\{|\lambda_j|\} < 1 \iff A^k \to 0 \quad \text{as } k \to \infty$$

Example 4.6.2. Let

$$A = \begin{bmatrix} 0.99 & 10^6 \\ 0 & 0.99 \end{bmatrix} \Longrightarrow A^{128} = \begin{bmatrix} 0.2765\ldots & 3.751 \cdot 10^6 \\ 0 & 0.2765\ldots \end{bmatrix}$$

This does not seem to agree with Theorem 4.6.1, however the theorem is correct, as products and powers of matrices are more complicated than those of complex numbers. For example, for every $z \in \mathbb{C}$, $|z|^k = |z^k|$ but generally $||A^k|| \neq ||A||^k$ for a matrix A and a norm $||\cdot||$. $||A||^k$ could grow before it then tends to zero. This occurs especially when A is not diagonalisable (A is defective, meaning that its eigenvectors do not span $\mathbb{R}^n$).

In fact, Theorem 4.6.1 is easy to prove for diagonalisable A, but difficult for defective A.

Definition 4.6.3. The **spectral radius** of a square matrix A is defined as

$$\rho(A) := \max\{|\lambda_j|\}$$

where the λ_j are the eigenvalues of A.

Proposition 4.6.4. If A is symmetric (or hermitian), then

- All its eigenvalues are real, so $Bv_i = \lambda_i v_i$ where $\lambda_i \in \mathbb{R}$.
- Its eigenvectors v_i form a basis of $\mathbb{R}^n$: every vector $x \in \mathbb{R}^n$ can be written as a linear combination of the eigenvectors of A, i.e. $x = \sum_i c_i v_i$ where v_i are the eigenvectors of A and c_i are real numbers.
- Its eigenvectors can be chosen to be orthonormal, i.e. $v_i \cdot v_j = \delta_{ij}$.
- Parseval's identity holds:

$$(||x||_2)^2 = \left(\sum_i c_i v_i, \sum_j c_j v_j\right) = \sum_{i,j} c_i c_j (v_i, v_j) = \sum_i c_i^2$$

Definition 4.6.5. A square matrix A is **normal** if it commutes with its transpose:

$$AA^T = A^T A$$

($AA^* = A^*A$ for complex matrices).

Definition 4.6.6. A matrix U is unitary if

$$U^* = U^{-1}$$

Definition 4.6.7. A matrix A is unitarily diagonalisable if $A = UDU^*$ for some unitary matrix U and diagonal matrix D.

Theorem 4.6.8. (**The spectral theorem**) A matrix A is unitarily diagonalisable if and only if it is normal.

Example 4.6.9. Let $a \neq 1$ and

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

is not normal, as

$$AA^T = \begin{bmatrix} 1+a^2 & a \\ a & 1 \end{bmatrix} \neq A^T A = \begin{bmatrix} 1 & a \\ a & 1+a^2 \end{bmatrix}$$

$\lambda = 1$ is the only eigenvalue, with eigenvector $(1, 0)$. A is not unitarily diagonalisable.

Theorem 4.6.10. The induced 2-norm of a matrix A is

$$\sqrt{\rho(A^T A)} = \max\{\sqrt{|\lambda|} : \lambda \text{ is an eigenvalue of } A^T A\}$$

Proof. For every matrix A, $A^T A$ is symmetric, so by Proposition 4.6.4 its eigenvalues are real and its eigenvectors, $\{v_i\}$ for $i \in \{1, \dots, n\}$, are orthonormal and span $\mathbb{R}^n$. So for every $x \in \mathbb{R}^n$,

$$x = \sum_i c_i v_i$$

for some $c_i \in \mathbb{R}$, $A^T A v_i = c_i v_i$ and $(v_i, v_j) = \delta_{i,j}$.

$$\begin{aligned}
(||Ax||_2)^2 = (Ax, Ax) = (A^T Ax, x) &= \left(A^T A \sum_i c_i v_i, \sum_j c_j v_j \right) \\
&= \left(\sum_i \lambda_i c_i v_i, \sum_k c_j v_j \right) \\
&= \sum_{i,j} \lambda_i c_i c_j (v_i, v_j) \\
&= \sum_i \lambda_i c_i^2
\end{aligned}$$

We claim that $\lambda_j \geq 0 \ \forall j$. Indeed, if $\lambda_s < 0$, let $x = v_s$, then

$$(||Av_s||_2)^2 = \lambda_s < 0$$

Write $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ (counting multiplicities). Then

$$(||Ax||_2)^2 = \sum_i \lambda_i c_i^2 \leq \lambda_n \sum_i c_i^2 = \lambda_n (||x||_2)^2$$

which gives

$$||A||_2 = \sup_{x \neq 0} \frac{||Ax||_2}{||x||_2} \leq \sqrt{\lambda_n}$$

Taking $x = v_n$ gives $(||Ax_n||_2)^2 = \lambda_n (||x_n||_2)^2$, which gives equality in the above inequality. □

Remark. $\rho(A^T A) = \rho(AA^T)$.

Example 4.6.11. Let

$$A = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix} \implies A^T A = \begin{bmatrix} 1 & 0 \\ 0 & 1+a^2 \end{bmatrix}$$

The eigenvalues of $A^T A$ are given by $0 = (1-\lambda)(1+a^2-\lambda) - a^2$, which gives $\lambda_1 \approx 2$ and $\lambda_2 \approx 2a^2$ for $|a| >> 1$.