DSA2102 NUMERICAL COMPUTATIONS LECTURE NOTES

Q

Abstract. This set of notes was written during the author's second year of undergraduate study. It is not guaranteed that claims made in this set of notes are correct, and, if there are any mistakes, they will almost surely be made by the author.

Contents

1. Fundamentals and Preliminaries

1.1. **Background.**

Generally when dealing with a real-world problem, we want to first model the situation. This is done by first properly phrasing a problem.

Definition 1.1. (Well-posed problems, informal) A problem is called a well-posed problem if there exists a unique solution. Furthermore, we expect the solution to be stable and continuous.

Remark 1.2. By the term 'stability', we mean that a small change in input will not cause a huge change in the output. For instance, suppose our solution is $f(x) = x^{100}$. Since $f(1)$ and $f(1.01)$ has a huge difference in value, we tend to think that the solution is unstable.

After having a problem, we can analyse it numerically. Generally, the strategy for dealing with numerical computations are to change some complicated questions to some simpler ones:

- replace infinite with finite;
- replace calculus with algebra;
- replace nonlinear with linear;
- replace arbitrary with structured.

Remark 1.3. Generally, the above approaches are employed due to computers' limited ability to do computation.

1.2. **Errors and error analysis.**

Normally, with the above approach, it becomes rather impossible for us to get the exact solution. We then consider the approximation we can have:

- modelling (by introducing assumptions which may not hold in the reality),
- measurement (with uncertainty introduced),
- prior computation (with uncertainty introduced),
- truncation (by reducing the number of digits involved in computation),
- discretisation (to change something continuous into something one by one),
- rounding.

Definition 1.4. (Absolute Error/Relative Error) Let x_c be a computed version of the exact quantity x. Then,

$$\textbf{absolute error} = |x_c - x|,$$

and

$$\textbf{relative error} = \frac{|x_c - x|}{|x|},$$

if the latter quantity exists.

Definition 1.5. (Precision) In this lecture, precision is informally defined as the number of digits a number keeps.

Remark 1.6. The idea of precision here is assess the amount of information carries. For instance, 3.4415 carries more information as compared to 3.44. However, if we view both of them as approximations of π, then they are both inaccurate (or at least not accurate enough). From this example, we can see that there is a fine distinction between accuracy and precision.

1.3. **Working examples.**

Suppose now our aim is to compute the value of f for some $f : \mathbb{R} \mapsto \mathbb{R}$ at $x = a$. We use the notations

- desired value: x;
- true input (approximated desired value): $\hat{x}$;
- desired function: f;
- approximated desired function: $\hat{f}$.

During our computation, we will have

$$\begin{aligned}\text{total error} &= \hat{f}(\hat{x}) - f(x) \\ &= \big[\hat{f}(\hat{x}) - f(\hat{x})\big] + \big[f(\hat{x}) - f(x)\big] \\ &= \text{computational error} + \text{data error}.\end{aligned}$$

Definition 1.7. (Computational error, informal) The computation error can be thought as the error occurs when we use different approaches to do the computation.

Definition 1.8. (Data error, informal) The data error can be seen as error inherited from the input data.

Remark 1.9. For instance, when we use $\hat{f}$ to approximate f, there will be some inaccuracies introduced. Since these inaccuracies come from our method of doing computations, say using a function to approximate another function, we consider this as the computational error. Yet, suppose our data was poorly collected, which will introduce a lot of inaccuracies into our computations. Since these errors are not introduced in our computation process per se but from the data, we consider them data error.

Remark 1.10. Since data error is inevitable and cannot be reduced by improving our computation algorithms, our discussion will be focusing on the computational error. Alternatively, the computational error can be numerically defined as

$$\text{computational error} = \text{truncation error} + \text{rounding error}.$$

We now consider a more concrete example:

Example 1.11. Suppose $f : \mathbb{R} \mapsto \mathbb{R}$ is a differentiable function. We want to compute the derivative of f at certain point.

By the definition of derivative, we have

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{1}$$

Since $h \to 0$ can be seen as a continuous process of h tending to 0 (in other words, h is a variable that can be infinitely small), to make the computations, we choose to use some small (but well-chosen) h to do the computations, which reduce (1) to

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \tag{2}$$

We now assess how much error has been introduced and try to determine what value h should take. By Taylor's theorem, for some $x \leq a \leq x + h$, we should have

$$f(x+h) \approx f(x) + f'(x)h + f''(a)\frac{h^2}{2}. \tag{3}$$

For (2), it is equivalent to truncating the quadratic term in (3). If we know that $|f''(t)| \leq M$ for all t near x (the point we want to determine $f'(x)$), the truncation error is bounded by

$$\frac{|f''(a)|h^2}{2} \leq \frac{h^2 M}{2} < \frac{hM}{2}.$$

Now, we have given an upper bound for truncation error, so we go back to (2) for the computation of $f'(x)$. Suppose our algorithm is able to give an approximate value of $f(x)$ and $f(x+h)$ with errors bounded by ε. Referring to (2), the rounding error is then bounded by

$$\frac{\varepsilon + \varepsilon}{h} = \frac{2\varepsilon}{h}.$$

The total error is thus bounded by

$$E(h) = \frac{hM}{2} + \frac{2\varepsilon}{h},$$

which can been seen as a function of h. To minimise $E(h)$, we can use the knowledge from calculus.

Using the same notations, an alternative way of thinking is that from (3), we have

$$f'(x) \approx \frac{f(x+h) - f(x) - f''(a)\frac{h^2}{2}}{h},$$

so the error (or uncertainty) is taken as

$$E(h) = \frac{\varepsilon + \varepsilon + M\frac{h^2}{2}}{h} = \frac{hM}{2} + \frac{2\varepsilon}{h}.$$

While this gives a more intuitive way of thinking, it reduces the significance of (1) to (2) a mere change from infinitely small variable h to a finitely small fixed h. It is difficult to say that which interpretation captures the real intention of the lecturer.

Now, we have given a theoretical upper bound for the computational error, but this does not take into account the data error. Therefore, probably we need another way to assess the quality of our approximation. We first define a few more concepts.

Definition 1.12. (Forward error) The forward error is defined as the difference between the actual value and our approximated value, which is $|\hat{f}(x) - f(x)|$.

Definition 1.13. (Backward error) The backward error is defined as the difference between the actual input and the pre-image of our approximated value, which is $|\hat{x} - x|$.

Definition 1.14. (Condition number) The condition number for an algorithm is given by

$$\text{condition number} = \frac{|\text{relative forward error}|}{|\text{relative backward error}|}.$$

Example 1.15. Let $y = \sqrt{x}$. We now want to compute the value of y at $x = 2$, i.e., the value of $\sqrt{2}$. Suppose we give $\sqrt{2} \approx 1.4$. The forward error is given by $|\sqrt{2} - 1.4|$, while the backward error is give by $|2 - 1.96|$ (noticing that taking $x = 2$ gives us $\sqrt{2}$ and taking $x = 1.96$ giving us 1.4.

Remark 1.16. We can use the condition number to assess the quality of our approximation algorithm. A high condition number indicates that a small backward error (a small change in the input) will lead to a high forward value (a huge difference in the output value), which then implies that the approximation may not be done properly.

Remark 1.17. It is noted that to compute the forward error, we need to the actual value of the function, which is impossible in the real practice. Therefore, more useful forms of the defining equation of condition number can be

$$\begin{aligned}\text{condition number} &= \frac{|\frac{f(\hat{x})-f(x)}{f(x)}|}{|\frac{\hat{x}-x}{x}|} \\ &= \frac{|\frac{\Delta y}{y}|}{|\frac{\Delta x}{x}|},\end{aligned}$$

$$|\text{relative forward error}| = \text{condition number} \times |\text{relative backward error}|.$$

Here, Δy can be interpreted in different ways, such as a small change in y or the error in y.

Example 1.18. Let $y = \sqrt{x}$. We now want to compute the derivative of y at some points $x = a$.

Proof. Since $f'(x) \approx \frac{f(x+h)-f(x)}{h}$, we have

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

or equivalently

$$\Delta x f'(x) \approx f(x + \Delta x) - f(x) = \Delta y.$$

This gives us

$$\begin{aligned}\text{condition number} &= \frac{|\frac{f(x+\Delta x)-f(x)}{f(x)}|}{|\frac{(x+\Delta x)-x}{x}|} \\ &= \frac{|\frac{\Delta x f'(x)}{f(x)}|}{|\frac{\Delta x}{x}|} \\ &= \left|\frac{x f'(x)}{f(x)}\right|.\end{aligned}$$

Here, think $\hat{x} = x + h$ in the way that $\hat{x}$ is a random value close to x but not equal to x, which leads to a small change in the value of $f(x)$. Furthermore, $f(x)$ and $f'(x)$ here are exact function which can be computed by using calculus knowledge. In this case $f(x) = \sqrt{x}$ and $f'(x) = 1/(2\sqrt{x})$. This gives us

$$\text{condition number} = \left|\frac{x f'(x)}{f(x)}\right| = \left|\frac{x\left(\frac{1}{2\sqrt{x}}\right)}{\sqrt{x}}\right| = \frac{1}{2}.$$

Using the condition number and the relative backward error (which can be computed and predicted), we can predict the relative forward error by using the equation

$$|\text{relative forward error}| \lessapprox \text{condition number} \times |\text{relative backward error}|,$$

which gives us some sense how accurate the approximation is. □

Remark 1.19. Throughout the module, the concept of condition number is frequently mentioned to demonstrate how accurate an algorithm can be, but it is never tested or explained thoroughly by the lecturer. In this set of notes, the author has tried his best to give proofs on how the condition number for various algorithms are derived. If one has difficulty in understanding the proof (possibly due to the author's poor writing ability), one may just take condition number as a measure of how 'good' an algorithm is. An algorithm with a small condition number is what we want.

Example 1.20. $y = \tan x$.

Proof. Let $f(x) = \tan x$. We have $f'(x) = \sec^2 x = 1 + \tan^2 x$, which gives us

$$\text{condition number} = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x + x\tan^2 x}{\tan x}\right| = \left|x\left(\tan x + \frac{1}{\tan x}\right)\right|,$$

which has no upper bound. In this case, let's see what will happen if we compute tan(1.57078) and tan(1.57079):

$$\tan(1.57078) \approx 61249, \quad \tan(1.57079) \approx 158057.$$

There is a small difference between 1.57078 and 1.57079, i.e., a small backward error. Yet, it gives a significant forward error. This can be explained by the unbounded condition number (at least to some extent). □

Definition 1.21. (Sensitivity, informal) If a small change in the input value can lead to a huge difference in the output value, then the problem is sensitive. This is in line with the definition of reference text (pg.50).

2. Linear system

2.1. **Linear systems.**

We suppose the readers have basic knowledge on how to manipulate matrix addition, matrix multiplication, scalar multiplication, inner product. A linear system in this set of notes is the same as a linear system of equations.

Definition 2.1. (Linearity) Suppose $f : \mathbb{R}^n \mapsto \mathbb{R}^m$. We say that f is linear iff

(1) $f(x + y) = f(x) + f(y)$;
(2) $f(\alpha x) = \alpha f(x)$.

Definition 2.2. (Affinity) We say that f is affine iff there exists a linear function g and a constant $c \in \text{dom}(f)$ such that

$$f(x) = g(x) + c,$$

where $\text{dom}(f)$ is the domain of f.

Example 2.3. $f(x) = 2x$ is a linear equation, whilst $f(x) = 2x + 3$ is an affine equation.

From this point onwards, let $\mathbf{Ax} = \mathbf{b}$ be a system of linear equations (or equivalently, a linear system), where $\mathbf{A}$ is a $m \times n$ matrix.

Definition 2.4. (Null space) The null space of a matrix $\mathbf{A}$ is defined by

$$\text{Null}(A) = \{\mathbf{x} \in \mathbb{R}^n : \mathbf{Ax} = \mathbf{0}\}.$$

Definition 2.5. (Spanning set) Let S be a set. We define $\text{Span}(S)$ by

$$\text{Span}(S) := \left\{ \mathbf{v} = \sum_{i=1}^{n} a_i \mathbf{s}_i : n \in \mathbb{Z}^+, a_i \in \mathbb{R}, \mathbf{s}_i \in S \right\}.$$

Theorem 2.6.

(1) *The linear system* $\boldsymbol{Ax} = \boldsymbol{b}$ *has a solution if* $b \in \text{Span}(\boldsymbol{A}_{column})$*, where* $\boldsymbol{A}_{column}$ *is the set of all column vectors of* $\boldsymbol{A}$*; and*

(2) *the linear system* $\boldsymbol{Ax} = \boldsymbol{b}$ *has a unique solution if* $\text{Null}(\boldsymbol{A}) = \{\boldsymbol{0}\}$.

2.2. **Pseudo-inverse.**

Definition 2.7. (Pseudo-inverse) Let $\mathbf{A}$ be a matrix. We define the pseudo-inverse of $\mathbf{A}$, denoted as $\mathbf{A}^\dagger$, by the following set of rules:

(1) if $\mathbf{A}$ is invertible, then $\mathbf{A}^\dagger = \mathbf{A}^{-1}$;

(2) if $\mathbf{A}$ is injective (having linearly independent columns), then $\mathbf{A}^\dagger := (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$; and

(3) if $\mathbf{A}$ is surjective (having linearly independent rows), then $\mathbf{A}^\dagger := \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$.

Lemma 2.8. *There are some results on pseudo-inverse:*

(1) *if $\boldsymbol{A}$ is injective, then $\boldsymbol{A}^\dagger\boldsymbol{A} = \boldsymbol{I}$, i.e., $\boldsymbol{A}$ has a left inverse;*

(2) *if $\boldsymbol{A}$ is surjective, then $\boldsymbol{A}\boldsymbol{A}^\dagger = \boldsymbol{I}$, i.e., $\boldsymbol{A}$ has a right inverse.*

Remark 2.9. For more information, one may search for Moore-Penrose inverse.

Remark 2.10. As the prefix 'pseudo' suggests, while pseudo-inverse is not the matrix inverse, it behaves like an inverse under certain conditions.

Lemma 2.11. *Suppose we are dealing with real numbers only. The linear system $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$ is always consistent, i.e., it always has a solution.*

Lemma 2.12. *If the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has multiple solutions, then $\hat{x} = \boldsymbol{A}^\dagger\boldsymbol{b}$ is a solution to the linear system.*

Proposition 2.13. *Let $\boldsymbol{A}$ be a $m \times n$ matrix. If the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is consistent, then the solutions to this system are exactly*

$$(*) \qquad \boldsymbol{x} = \boldsymbol{A}^\dagger\boldsymbol{b} + (\boldsymbol{I} - \boldsymbol{A}^\dagger\boldsymbol{A})\boldsymbol{y}$$

for any $\boldsymbol{y} \in \mathbb{R}^n$.

Proof. We first prove that every such $\mathbf{x}$ is a solution to the linear system.

If $\mathbf{A}$ is invertible or injective, then we have $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}$ by definition. Hence, $(*)$ is reduced to $\mathbf{x} = \mathbf{A}^\dagger\mathbf{b}$, which is a solution to the equation.

If $\mathbf{A}$ is surjective, then

$$\begin{aligned}\mathbf{Ax} &= \mathbf{AA}^{\dagger}\mathbf{b} + \mathbf{A}(\mathbf{I} - \mathbf{A}^{\dagger}\mathbf{A})\mathbf{y} \\ &= \mathbf{A}[\mathbf{A}^T(\mathbf{AA}^T)^{-1}]\mathbf{b} + \mathbf{Ay} - \mathbf{A}[\mathbf{A}^T(\mathbf{AA}^T)^{-1}]\mathbf{Ay} \\ &= \mathbf{b} + (\mathbf{Ay} - \mathbf{IAy}) \\ &= \mathbf{b}.\end{aligned}$$

The proof of all solutions $\mathbf{x}$ being in the form of $(*)$ is beyond the author's ability. □

2.3. **Matrix norms.**

Definition 2.14. (Norm) A norm on a $\mathbb{R}$-vector space X is a real-valued function $\|\cdot\| : S \mapsto \mathbb{R}^+$ satisfying

(1) (Triangular inequality) $\|x\| + \|y\| \geq \|x + y\|$;

(2) (Positive definite) $\|x\| = 0 \iff x = 0$;

(3) (Scalar multiplicative) $(\forall a \in \mathbb{R})(\forall x \in X)(\|ax\| = |a| \, \|x\|)$.

Remark 2.15. Since in my definition, $\|\cdot\|$ is a non-negative function, it naturally satisfies the condition of non-negativity stated in the lecture slides.

Example 2.16. (p-norm) For any integer $p \in \mathbb{Z}^+$, we define p-norm as

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |\mathbf{x}_i|^p \right)^{\frac{1}{p}}.$$

The reader may check that when $p = n = 2$, it is the distance formula in $x - y$ plane.

Remark 2.17. In some sense, norm is a generalised version of length. For instance, for the set of continuous functions, it is difficult to say how long $f(x) = x$ is in our intuitive sense of length, so we introduce norms to describe lengths/magnitudes in this space. It is to be noted that there is no unique norm on a set. In other words, we can define a number of norms as long as they satisfy the definition of norms.

Proposition 2.18. (Least-square methods) *If the linear system $\boldsymbol{Ax} = \boldsymbol{b}$ has solutions,*

$$\|\hat{\boldsymbol{x}}\|_2 \leq \|\boldsymbol{x}\|_2$$

for any other solution $\boldsymbol{x}$. If the system has no solution, then

$$\|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{b}\|_2 \leq \|\boldsymbol{Ax} - \boldsymbol{b}\|_2$$

for all other $\boldsymbol{x}$.

Definition 2.19. (Induced matrix norm) Let $\mathbf{A}$ be a matrix. The induced matrix norm $\|\mathbf{A}\|$ is defined as

$$\|\mathbf{A}\| := \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \sup_{\|\mathbf{x}\|=1} \|\mathbf{Ax}\| .$$

The equality holds because

$$\frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{A}(\|\mathbf{x}\|\,\bar{\mathbf{x}})\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{x}\|\,\|\mathbf{A}\bar{\mathbf{x}}\|}{\|\mathbf{x}\|} = \|\mathbf{A}\bar{\mathbf{x}}\| ,$$

where $\bar{\mathbf{x}}$ is a unit vector.

Definition 2.20. (p-norm for matrices) Let $\mathbf{A}$ be a $m \times n$ matrix. We define the p-norm of a matrix by setting

$$p = 1 : \|\mathbf{A}\|_1 := \max_j \sum_{i=1}^{m} \|\mathbf{A}_{ij}\| , \text{i.e., the largest column sum;}$$

$$p = 2 : \|\mathbf{A}\|_2 := \sqrt{\lambda_{max}(\mathbf{A}^T\mathbf{A})} = \sigma_{\max}(A), \text{i.e., the largest eigenvalue of } \mathbf{A}^T\mathbf{A};$$

$$p = \infty : \|\mathbf{A}\|_\infty := \max_i \sum_{j=1}^{n} \|\mathbf{A}_{ij}\| , \text{i.e., the largest row sum.}$$

Proposition 2.21. (Consistency of norms) *Let $\boldsymbol{A}$ be a $m \times n$ matrix and $\boldsymbol{x} \in \mathbb{R}^m$. We have*

$$\|\boldsymbol{Ax}\| \leq \|\boldsymbol{A}\| \cdot \|\boldsymbol{x}\| .$$

Remark 2.22. The writer doubts whether this holds for all possible norms defined on the set of all $m \times n$ matrices with real entries. At the very least, this holds for the induced norm.

2.4. **Analysis of errors.**

We now proceed to the analysis of forward error and backward error in the computation of $\mathbf{Ax}$ for some invertible matrix $\mathbf{A}$. Unlike the way presented in the lecture slides, the author chooses to analyse the magnitude of relative forward error and relative backward error first.

(1) Analysis of the relative forward error:

Let $\hat{\mathbf{y}}$ be the computed value and $\mathbf{y}$ be the theoretical value. We have $\Delta\mathbf{y} = \hat{\mathbf{y}} - \mathbf{y}$. One way to represent this error is to consider

$$\mathbf{y} = \mathbf{Ax}$$
$$\hat{\mathbf{y}} = (\mathbf{A} + \mathbf{E})\mathbf{x}$$

for some matrix $\mathbf{E}$, which gives us

$$\hat{\mathbf{y}} - \mathbf{y} = \mathbf{Ex} \implies \|\hat{\mathbf{y}} - \mathbf{y}\| = \|\mathbf{Ex}\|.$$

To eliminate $\mathbf{x}$ from the expression, we use the fact that $\mathbf{A}$ is invertible, i.e., $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y}$:

$$\|\hat{\mathbf{y}} - \mathbf{y}\| = \|\mathbf{Ex}\| \implies \|\Delta\mathbf{y}\| = \left\|\mathbf{EA}^{-1}\mathbf{y}\right\|.$$

By the consistency of norms, we have

$$\|\Delta\mathbf{y}\| = \left\|\mathbf{EA}^{-1}\mathbf{y}\right\| \leq \|\mathbf{E}\| \left\|\mathbf{A}^{-1}\right\| \|\mathbf{y}\|.$$

Since $\mathbf{E}$ can be seen as the actual matrix used in computation if all other errors are negligible, we can write $\mathbf{E} = \Delta\mathbf{A}$, which gives us the final form of inequality:

$$\|\Delta\mathbf{y}\| \leq \|\mathbf{E}\| \left\|\mathbf{A}^{-1}\right\| \|\mathbf{y}\| \implies \frac{\|\Delta\mathbf{y}\|}{\|\mathbf{y}\|} \leq \|\Delta\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| = \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|},$$

where $\dfrac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|}$ represents the relative error in the magnitude of the matrix $\mathbf{A}$.

Remark 2.23. In the author's understanding, in the analysis of forward error, we shall assume that the input value (used in computation) is accurate. Similarly, when analysing the backward error, we shall assume that the algorithm will not introduce additional uncertainties.

(2) Analysis of the relative backward error:

Let $\hat{\mathbf{x}}$ be the actual value used for computation and $\mathbf{x}$ be the true value. We have $\Delta\mathbf{x} = \hat{\mathbf{x}} - \mathbf{x}$. Consider

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

$$\mathbf{A}\mathbf{x}' = \mathbf{b} + \Delta\mathbf{b}. \tag{2}$$

Since (2) − (1) gives us $\mathbf{A}(\Delta\mathbf{x}) = \Delta\mathbf{b}$, given that $\mathbf{A}$ is invertible, we have

$$\|\hat{\mathbf{x}} - \mathbf{x}\| = \|\Delta\mathbf{x}\| = \left\|\mathbf{A}^{-1}\Delta\mathbf{b}\right\|.$$

By the consistency of norms and the non-negativity of norms, we have

$$\|\Delta\mathbf{x}\| = \left\|\mathbf{A}^{-1}\Delta\mathbf{b}\right\| \leq \left\|\mathbf{A}^{-1}\right\| \|\Delta\mathbf{b}\|. \tag{3}$$

We now need an estimation of how large $\|\mathbf{x}\|$ can be in terms of $\|\mathbf{b}\|$ (since there has been a $\|\Delta\mathbf{x}\|$ in (3). Considering $\|\mathbf{A}\mathbf{x}\| = \|\mathbf{b}\|$, we have

$$\|\mathbf{b}\| = \|\mathbf{A}\mathbf{x}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \implies \|\mathbf{x}\| \geq \frac{\|\mathbf{b}\|}{\|\mathbf{A}\|}. \tag{4}$$

Combining (3) and (4) gives us

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\| \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}.$$

2.5. **Condition number in matrix computations.**

Some readers may have realised that the condition number defined in the lecture slides makes less sense with the above analysis. By definition, we shall have

$$\text{condition number} = \frac{|\text{relative forward error}|}{|\text{relative backward error}|},$$

or equivalently,

$$\text{condition number} = \frac{\left|\frac{\Delta y}{y}\right|}{\left|\frac{\Delta x}{x}\right|}.$$

Yet, the way that the lecture slides define condition number is

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \left\|\mathbf{A}^{-1}\right\|,$$

which appears in our analysis of both relative forward error and relative backward error. The best explanation the author can give is, in the analysis of relative forward error, we consider the matrix as part of the input, and in the analysis of relative output error, we consider $\mathbf{b}$ to be the output.

One last part of lecture 3 is that since it is difficult to compute $\left\|\mathbf{A}^{-1}\right\|$, we would like to give it an approximation:

$$\begin{aligned} \mathbf{Ax} = \mathbf{y} &\implies \|\mathbf{Ax}\| = \|\mathbf{y}\| \\ &\implies \|\mathbf{x}\| = \left\|\mathbf{A}^{-1}\mathbf{y}\right\| \leq \left\|\mathbf{A}^{-1}\right\| \|\mathbf{y}\| \\ &\implies \frac{\|\mathbf{x}\|}{\|\mathbf{y}\|} \leq \left\|\mathbf{A}^{-1}\right\|. \end{aligned}$$

2.6. **Applications of different norms.**

$\|\mathbf{A}\|_1$ measures the distance between two points in terms of the total difference of their coordinates.

$\|\mathbf{A}\|_2$ measures the Euclidean distance, i.e., the length of the straight line connecting two points.

$\|\mathbf{A}\|_\infty$ measures the maximum difference in the coordinates of two points.

3. Algorithms for solving linear systems

3.1. Elementary row operations.

Definition 3.1. (Equivalent systems) Two linear systems are said to be equivalent if they have the same solutions.

Definition 3.2. (Elementary row operations, ERO) There are three EROs in total:

(1) swap a row with another;

(2) add or subtract a multiple of one row from another; and

(3) multiply a row by a nonzero constant.

Lemma 3.3. *Applying EROs to a linear system yields its equivalent system, i.e., applying EROs to a linear system does not change the solutions to the linear system.*

3.2. Matrix representation of EROs.

Let $\mathbf{A}$ be a $m \times n$ matrix. Each ERO operated on $\mathbf{A}$ can be carried out by pre-multiplying an elementary matrix $\mathbf{E}$ to $\mathbf{A}$.

(1) Swap the a-th row and the b-th row:

$$\mathbf{E}_{a\leftrightarrow b} = (e_{ij})_{m\times m} = \begin{cases} 1 & \text{if } (i = j \text{ and } i \neq a \text{ and } j \neq b); \\ 1 & \text{if } (i = a \text{ and } j = b) \text{ or } (i = b, j = a); \\ 0 & \text{otherwise.} \end{cases}$$

For instance, if $\mathbf{A}$ is a 3×4 matrix, the matrix $\mathbf{E}_{1\leftrightarrow 2}$ for swapping the first row and the second row of $\mathbf{A}$ is

$$\mathbf{E}_{1\leftrightarrow 2} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

(2) Add a multiple k of a-th row to b-th row:

$$\mathbf{E}_{b+ka} = (e_{ij})_{m\times m} = \begin{cases} 1 & \text{if } i = j; \\ k & \text{if } i = b \text{ and } j = a; \\ 0 & \text{otherwise.} \end{cases}$$

For instance, if $\mathbf{A}$ is a 3×4 matrix, the matrix $\mathbf{E}_{2+3(1)}$ for adding 3 times of the first row to the second row of $\mathbf{A}$ is

$$\mathbf{E}_{2+3(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

(3) Multiply the a-th row by a non-zero constant k:

$$\mathbf{E}_{ka} = (e_{ij})_{m\times m} = \begin{cases} 1 & \text{if } i = j \neq a; \\ k & \text{if } i = j = a; \\ 0 & \text{otherwise.} \end{cases}$$

For instance, if $\mathbf{A}$ is a 3×4 matrix, the matrix $\mathbf{E}_{3(3)}$ for multiplying the third row of $\mathbf{A}$ by 3 is

$$\mathbf{E}_{3(3)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}.$$

Lemma 3.4. *All elementary matrices $\boldsymbol{E}$ are invertible.*

Lemma 3.5. $(\boldsymbol{E}_{i+cj})^{-1} = (\boldsymbol{E}_{i-cj})$

Lemma 3.6. *Let c_1, c_2 be real numbers, and $i_1 \geq i_2$. We have $\boldsymbol{E}_{i_1+c_1j_1}\boldsymbol{E}_{i_2+c_2j_2} = \boldsymbol{E}_{i_1+c_1j_1,i_2+c_2j_2}$*

Example 3.7. By definition, we have

$$\mathbf{E}_{3+2(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}, \mathbf{E}_{2+3(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Manually computing out $(\mathbf{E}_{3+2(2)})^{-1}$ and $\mathbf{E}_{2+3(1)} \times \mathbf{E}_{3+2(2)}$ gives us

$$(\mathbf{E}_{3+2(2)})^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{pmatrix}, \mathbf{E}_{2+3(1)} \times \mathbf{E}_{3+2(2)} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 2 & 1 \end{pmatrix}.$$

Yet, since $3 > 2$, if we compute $\mathbf{E}_{3+2(2)} \times \mathbf{E}_{2+3(1)}$, we will have

$$\mathbf{E}_{3+2(2)} \times \mathbf{E}_{2+3(1)} = \begin{pmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 6 & 2 & 1 \end{pmatrix},$$

which is equivalent to adding 3 times the first row to the second row first, and then add 2 times the result second row to the third row.

Remark 3.8. In reference text, $\mathbf{E}_{i+kj}$ is denoted as $\mathbf{L}_{ji}(c)$ to emphasise that the matrix is in fact lower triangular.

3.3. **Special matrices.**

In this section, we consider the linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a $m \times n$ matrix, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$.

Definition 3.9. (Permutation) A matrix $\mathbf{P}$ is called a permutation matrix of order m iff

1) $\mathbf{A}$ is a square matrix of order m;

2) every column of $\mathbf{A}$ has exactly one entry to be 1;

3) every row of $\mathbf{A}$ has exactly one entry to be 1; and

4) all other entries of $\mathbf{A}$ are 0.

Proposition 3.10. *Permutation matrices are orthogonal.*

Corollary 3.11. *If $\boldsymbol{A}$ is a permutation matrix, then $\boldsymbol{A}$ is invertible and $\boldsymbol{A}^{-1} = \boldsymbol{A}^T$.*

Definition 3.12. (Lower triangular matrices) An $m \times n$ matrix L is lower triangular if its entries satisfy $l_{ij} = 0$ for $i < j$.

Definition 3.13. (Upper triangular matrices) An $m \times n$ matrix U is upper triangular if its entries satisfy $u_{ij} = 0$ for $i > j$.

Definition 3.14. (Diagonal matrix) An $m \times n$ matrix D is diagonal if its entries satisfy $d_{ij} = 0$ for $i \neq j$.

3.4. **Complexity analysis of Gaussian elimination.**

Let $\mathbf{Ax} = \mathbf{b}$ be a linear system, where $\mathbf{A}$ is a $n \times n$ matrix. We can write its augmented matrix as

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{cccc|c} a_{11} & a_{12} & \dots & a_{1n} & b_1 \\ a_{21} & a_{22} & \dots & a_{2n} & b_2 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} & b_n \end{array}\right).$$

Carrying out the Gaussian elimination is essentially to eliminate all columns of the matrix. By the phrase "eliminating a column", we mean to convert all entries below the main diagonal to zeros by using EROs. To achieve so, we subtract a multiple of one row from all rows below it. For instance, if we want to eliminate the first column, we carry out the following operations for each of $2 \leq i \leq n$:

(1) calculate $c_i = \dfrac{a_{i1}}{a_{11}}$;

(2) subtract c_i times the first row from the i-th row.

In (1), we conduct 1 operation. In (2), we conduct $2n$ operations: multiplying all entries in the first row by c_1 (n operations) and subtract the newly obtained first row from the i-th row (n operations, since there are n elements in the i-th row). Therefore, in total, we will conduct $2n+1$ elements to convert a_{i1} to 0 for $2 \leq i \leq n$.

Similarly, if we want to eliminate the second column, we will carry out a similar process. Yet this time, since we do not need to consider the first column at all, the total number of operations required to convert a_{i2} will be $2(n-1)+1$ for $3 \leq i \leq n$. With a similar argument, the number of operations required to convert a_{ij} to zero will be $2(n+1-j)+1$ for $1 \leq j < i \leq n$.

Therefore, the total operations for Gaussian elimination is

$$\begin{aligned}
\sum_{i=2}^{n}\sum_{j=1}^{i-1} 2(n+1-j) &= \sum_{i=2}^{n}\left(2(n+1)(i-1) - 2\frac{(i-1)i}{2}\right) \\
&= \sum_{i=2}^{n}\left(2(n+1)i - 2(n+1) - i^2 + i\right) \\
&= \sum_{i=1}^{n}\left((2n+3)i - 2(n+1) - i^2\right) \\
&= \frac{(2n+3)(n+1)(n)}{2} - 2n(n+1) - \frac{n(n+1)(2n+1)}{6} \\
&= \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{7}{6}n \\
&\equiv O(n^3)
\end{aligned}$$

To better illustrate this idea, we can put the number of operations required to convert a_{ij} to 0 at the (i,j)-th entry (if a_{ij} does not need to be converted, then the number of operations is

0), which gives us

$$\begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ 2n+1 & 0 & \dots & 0 & 0 \\ 2n+1 & 2(n-1)+1 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & \vdots & \vdots \\ 2n+1 & 2(n-1)+1 & \dots & 0 & 0 \\ 2n+1 & 2(n-1)+1 & \dots & 2(2)+1 & 0 \end{pmatrix}.$$

After conducting the Gaussian elimination, we need to do the backward substitution. For x_n, we naturally have $x_n = \dfrac{b_n}{a_{nn}}$, which involves 1 operation. For x_{n-1}, we need to use the value of x_n to solve $a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1}$, which gives us $x_{n-1} = \dfrac{b_{n-1} - a_{n-1,n}x_n}{a_{n-1,n-1}}$ that requires 3 operations, namely $a_{n-1,n} \times x_n$, $b_{n-1} - a_{n-1,n}x_n$, and the division. With a similar argument, to get x_i, we need $2(n-i)+1$ operations for $1 \leq i \leq n$. Hence in total, we need

$$\sum_{i=1}^{n} 2(n-i)+1 = \sum_{i=1}^{n}(2i-1) = n^2 \equiv O(n^2).$$

3.5. **The LU factorisation.**

In this section, we consider the linear system $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a $m \times n$ matrix, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$.

By Gaussian elimination discussed above, by pre-multiplying $\mathbf{A}$ with some lower triangular matrices (as EROs), we have $\mathbf{L}'\mathbf{A} = \mathbf{U}$ for some upper triangular matrix $\mathbf{U}$. From the lemmas in the previous sections, we have $\mathbf{L}'$ is invertible and denote $(\mathbf{L}')^{-1} = \mathbf{L}$. This gives us

$$\mathbf{A} = \mathbf{LU},$$

which is known as the $\mathbf{LU}$ factorisation.

With the $\mathbf{LU}$ factorisation, solving $\mathbf{Ax} = \mathbf{b}$ is equivalent to

(1) solve $\mathbf{Lc} = \mathbf{b}$ for $\mathbf{c}$;

(2) solve $\mathbf{Ux} = \mathbf{c}$ for $\mathbf{x}$.

As compared to the traditional Gaussian elimination, the **LU** factorisation is faster since it does not involve the computation of $\mathbf{b}$ until the backward-substitution step. If we are dealing with a series of questions

$$\mathbf{Ax} = \mathbf{b}_1$$
$$\mathbf{Ax} = \mathbf{b}_2$$
$$\vdots$$
$$\mathbf{Ax} = \mathbf{b}_k$$

we need to do Gaussian eliminations for each of the individual questions, so we need about $\frac{2kn^3}{3}$ operations.

Yet, for **LU** factorisation, we just need to do Gaussian elimination for $\mathbf{A}$ once and do two forward/backward substitutions for each individual question. The total number of operations required is thus $\frac{2n^3}{3} + 2kn^2$. When n is large, i.e. n^2 is negligible, the difference is significant.

3.6. **Partial pivoting.**

In the previous examples of Gaussian elimination, we always use a_{ii} as a pivot to eliminate the i-th column. Yet, if the pivot is of a very small value, this will cause problems. A detailed example of such problem is examined in the textbook section 2.3.

To avoid these problems, before we eliminate the j-th column, we swap the row involved in the elimination process with the largest entry in the j-th column to the pivoting position. That is, we select the row p satisfying

$$|a_{pj}| \geq |a_{ij}|$$

for $j \leq i \leq n$ and swap the i-th row with the p-th row. As a result, we ensure that the pivot entry is the greater than all entries below it.

Theorem 3.15. (Fundamental theorem of permutation matrices) *Let $\boldsymbol{P}$ be the $n\times n$ permutation matrix formed by a particular set of row exchanges applied to the identity matrix. For any $n \times n$ matrix $\boldsymbol{A}$, $\boldsymbol{PA}$ is the matrix obtained by applying exactly the same set of row changes to $\boldsymbol{A}$.*

Remark 3.16. This theorem suggests that a permutation matrix can be seen as a set of row changes. In other words, the row exchanges caused by pre-multiplication by $\mathbf{P}$ are exactly the ones involved in the construction of $\mathbf{P}$.

With the above knowledge, we are empowered to conduct $\mathbf{PA} = \mathbf{LU}$ factorisation, which is just the $\mathbf{LU}$ factorisation of the row-exchanged $\mathbf{A}$. By $\mathbf{PA} = \mathbf{LU}$ factorisation, we have

$$\mathbf{Ax} = \mathbf{b} \implies \mathbf{PAx} = \mathbf{Pb} \implies \mathbf{LUx} = \mathbf{b}'.$$

Remark 3.17. For banded systems and sparse matrices, instead of storing the whole matrix, we can store certain elements (but as a pure math student, the author does not care).

3.7. **Symmetric matrices.**

In this section, $\mathbf{A}$ is a square matrix of dimension n by default, unless otherwise specified.

Definition 3.18. (Symmetric matrix) Let $\mathbf{A} = (a_{ij})$. The matrix $\mathbf{A}$ is said to be symmetric iff $a_{ij} = a_{ji}$ for all $1 \leq i, j \leq n$.

Lemma 3.19. *A matrix $\boldsymbol{A}$ is orthogonally diagonalisable iff $\boldsymbol{A}$ is symmetric.*

Lemma 3.20. *Let $\boldsymbol{A} = (a_{ij})_{m\times n}$. We have $\boldsymbol{AA}^T$ and $\boldsymbol{A}^T\boldsymbol{A}$ are both square and symmetric.*

Proof. It is clear that both $\mathbf{AA}^T$ and $\mathbf{A}^T\mathbf{A}$ are square. We now prove that they are symmetric.

Write $\mathbf{A} = (a_{ij})_{m\times n}$, $\mathbf{A}^T = (a'_{ij})_{n\times m}, \mathbf{AA}^T = (c_{ij})_{m\times m}$. By the definition of symmetric matrix, we have $a_{ij} = a'_{ij}$. Thus, by the definition of matrix multiplication, we have

$$c_{ij} = \sum_{k=1}^{n} a_{ik}a'_{kj} = \sum_{k=1}^{n} a'_{ki}a_{jk} = \sum_{k=1}^{n} a_{jk}a'_{ki} = c_{ji}.$$

With a similar argument, we conclude that $\mathbf{A}^T\mathbf{A}$ is also symmetric. □

Definition 3.21. (Positive-definite matrix) A matrix $\mathbf{A}$ is positive-definite if $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$ for all vectors $\mathbf{x} \neq \mathbf{0}$; it is negative-definite iff $-\mathbf{A}$ is positive-definite; it is positive semi-definite iff $\mathbf{x}^T\mathbf{A}\mathbf{x} \geq 0$ for all $\mathbf{x}$.

Remark 3.22. There is a slight abuse of notation since technically, $\mathbf{x}^T\mathbf{A}\mathbf{x}$ is a 1×1 matrix. Yet, the definition treats it as a scalar.

Remark 3.23. Matrices can be non-zero and neither positive-definite nor negative-definite.

Proposition 3.24. *Positive-definite and negative-definite matrices are invertible.*

Proof. WLOG, let $\mathbf{A}$ be a positive-definite matrix. Suppose the proposition does not hold. Since $\mathbf{A}$ is not invertible, $\text{rank}(\mathbf{A}) \neq \dim(\mathbf{A})$. By the rank-nullity theorem, we have $\text{Null}(\mathbf{A}) > 0$, which implies that the basis of the null space of $\mathbf{A}$ contains at least one vector. Thus, the null space of $\mathbf{A}$ contains non-zero elements. Choose $\mathbf{u} \neq \mathbf{0}$ from the null space of $\mathbf{A}$. We have $\mathbf{u}^T\mathbf{A}\mathbf{u} = \mathbf{u}^T\mathbf{0} = 0$, a contradiction to the positive-definite nature of $\mathbf{A}$. □

Lemma 3.25. *If $\boldsymbol{A}$ is symmetric, then $\boldsymbol{A}$ is positive-definite if and only if all of its eigenvalues are positive.*

Proof. Let $\mathbf{A}$ be a symmetric matrix.

($\implies$) Suppose $\mathbf{A}$ is a positive-definite. Let λ be an eigenvalue of $\mathbf{A}$. By the definition of positive-definiteness, we have

$$\mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T(\lambda\mathbf{x}) = \lambda(\mathbf{x}^T\mathbf{x}) = \lambda \left\|\mathbf{x}\right\|_2 > 0 \implies \lambda > 0.$$

($\impliedby$) Suppose all eigenvalues of $\mathbf{A}$ are positive. Since $\mathbf{A}$ is symmetric, $\mathbf{A}$ is orthogonally diagonalisable. Thus, we can find an orthonormal basis $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ of $\mathbb{R}^{\dim(\mathbf{A})}$ such that every vector in the basis is an eigenvector of $\mathbf{A}$. Let $\mathbf{x}$ be arbitrarily given vector. We can write

$$\mathbf{x} = \sum_{i=1}^{n} c_i\mathbf{v}_i,$$

which gives us

$$\begin{aligned}\mathbf{x}^T\mathbf{A}\mathbf{x} = \left(\sum_{i=1}^{n} c_i\mathbf{v}_i\right)^T \mathbf{A}\left(\sum_{i=1}^{n} c_i\mathbf{v}_i\right) &= \left(\sum_{i=1}^{n} c_i\mathbf{v}_i^T\right)\left(\sum_{i=1}^{n} c_i\lambda_i\mathbf{v}_i\right)\\ &= \sum_{i=1}^{n} c_i^2\lambda_i\\ &> 0.\end{aligned}$$

Thus, $\mathbf{A}$ is positive-definite. □

Lemma 3.26. *If $\boldsymbol{A}$ is $n \times n$ symmetric positive-definite and $\boldsymbol{X}$ is an $n \times m$ matrix of full rank with $n \geq m$, then $\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X}$ is an $m \times m$ symmetric positive-definite matrix.*

Proof. The matrix $\mathbf{X}^T\mathbf{A}\mathbf{X}$ is obviously $m \times m$, and its symmetry follows

$$(\mathbf{X}^T\mathbf{A}\mathbf{X})^T = \mathbf{X}^T\mathbf{A}^T\mathbf{X} = \mathbf{X}^T\mathbf{A}\mathbf{X}.$$

We now prove that $\mathbf{X}^T\mathbf{A}\mathbf{X}$ is positive-definite. Since $\mathbf{X}$ is full rank with $n \geq m$, we have $\mathbf{X}\mathbf{v} = \mathbf{0} \iff \mathbf{v} = \mathbf{0}$, where $\mathbf{v}$ is a $m \times 1$ vector. Since $\mathbf{A}$ is positive definite, we have

$$\mathbf{v}^T(\mathbf{X}^T\mathbf{A}\mathbf{X})\mathbf{v} = (\mathbf{X}\mathbf{v})^T\mathbf{A}(\mathbf{X}\mathbf{v}) > 0$$

for all $\mathbf{v} \in \mathbb{R}^m$. □

Definition 3.27. (Principal submatrix) A principal submatrix of a square matrix $\mathbf{A}$ is a square submatrix whose diagonal entries are diagonal entries of $\mathbf{A}$.

Lemma 3.28. *Any principal submatrix of a symmetric positive-definite matrix is symmetric positive-definite.*

3.8. **Cholesky factorisation.**

Example 3.29. Consider the symmetric positive-definite matrix $\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix}$.

By the lemma 3.28, we conclude that $a > 0$. Our question is whether we can write $\mathbf{A} = \mathbf{L}\mathbf{L}^T$ for some upper triangular matrix $\mathbf{L}$.

To do so, we write

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} \sqrt{a} & 0 \\ u & v \end{pmatrix} \begin{pmatrix} \sqrt{a} & u \\ 0 & v \end{pmatrix} = \begin{pmatrix} a & u\sqrt{a} \\ u\sqrt{a} & u^2+v^2 \end{pmatrix}.$$

This gives us

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{c-\frac{b^2}{a}} \end{pmatrix} \begin{pmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c-\frac{b^2}{a}} \end{pmatrix} = \mathbf{L}\mathbf{L}^T.$$

A natural question is whether we can do this to all symmetric positive-definite matrices? The answer is yes.

Theorem 3.30. (Cholesky factorisation theorem) *If $\boldsymbol{A}$ is a symmetric positive definite $n \times n$ matrix, then there exists a lower triangular $n \times n$ matrix $\boldsymbol{L}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{L}^T$.*

Proof. The idea of the proof is to construct $\mathbf{L}$ inductively.

For the base case $n = 2$, we have

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} = \begin{pmatrix} \sqrt{a} & 0 \\ \frac{b}{\sqrt{a}} & \sqrt{c-\frac{b^2}{a}} \end{pmatrix} \begin{pmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c-\frac{b^2}{a}} \end{pmatrix} = \mathbf{L}\mathbf{L}^T.$$

Suppose we have constructed $\mathbf{L}$ for $n-1 \times n-1$ symmetric positive-definite matrices. We now proceed to constructing $\mathbf{L}$ for $\mathbf{A}$, a symmetric positive-definite matrix of dimension n.

First, we write $\mathbf{A}$ as

$$\mathbf{A} = \left(\begin{array}{c|c} a & \mathbf{b}^T \\ \hline \mathbf{b} & \mathbf{C} \end{array}\right),$$

where $\mathbf{b}$ is a column vector of length $(n-1)$ and $\mathbf{C}$ is an $(n-1)\times(n-1)$ symmetric positive-definite submatrix.

Set $\mathbf{u} = \mathbf{b}/\sqrt{a}$ as in the 2×2 case. We define the invertible matrix $\mathbf{S}$ as

$$\mathbf{S} = \left(\begin{array}{c|c} \sqrt{a} & \mathbf{u}^T \\ \hline 0 & \\ \vdots & \mathbf{I} \\ 0 & \end{array}\right).$$

Thus, by setting $\mathbf{A}_1 = \mathbf{C} - \mathbf{u}\mathbf{u}^T$, we have

$$\mathbf{S}^T \left(\begin{array}{c|c} 1 & 0\ \ldots\ 0 \\ \hline 0 & \\ \vdots & \mathbf{A}_1 \\ 0 & \end{array}\right) \mathbf{S} = \left(\begin{array}{c|c} \sqrt{a} & 0\ \ldots\ 0 \\ \hline \mathbf{u} & \mathbf{I} \end{array}\right) \left(\begin{array}{c|c} 1 & 0\ \ldots\ 0 \\ \hline 0 & \\ \vdots & \mathbf{A}_1 \\ 0 & \end{array}\right) \left(\begin{array}{c|c} \sqrt{a} & \mathbf{u}^T \\ \hline 0 & \\ \vdots & \mathbf{I} \\ 0 & \end{array}\right)$$

$$= \left(\begin{array}{c|c} a & \mathbf{b}^T \\ \hline \mathbf{b} & \mathbf{u}\mathbf{u}^T + \mathbf{A}_1 \\ & \end{array}\right) = \mathbf{A}.$$

Since $\mathbf{A}$ is symmetric positive-definite, $(\mathbf{S}^T)^{-1}\mathbf{A}\mathbf{S}^{-1}$ is positive-definite. Since $\mathbf{A}_1$ is a principal submatrix of $(\mathbf{S}^T)^{-1}\mathbf{A}\mathbf{S}^{-1}$, we conclude that $\mathbf{A}_1$ is symmetric positive-definite. By our induction

hypothesis, there exists a lower triangular matrix $\mathbf{L}'$ such that $\mathbf{A}_1 = \mathbf{L}'(\mathbf{L}')^T$. Define

$$\mathbf{L} = \left(\begin{array}{c|c} \sqrt{a} & 0 \ \ldots \ 0 \\ \hline \mathbf{u} & \mathbf{L}' \end{array}\right),$$

which gives us $\mathbf{L}\mathbf{L}^T = \mathbf{A}$ as desired. □

3.9. **Methods for checking positive-definite matrices.**

There are a few ways for us to check whether a matrix $\mathbf{A}$ is positive-definite:

(1) compute $\mathbf{x}^T\mathbf{A}\mathbf{x}$ directly and prove that the expression obtained is positive-definite;
(2) for symmetric matrix $\mathbf{A}$, we can check whether all of its eigenvalues are positive; and
(3) check whether all leading principal minors of $\mathbf{A}$ are positive.

Definition 3.31. (Minor) A minor of a matrix $\mathbf{A}$ is the determinant of a submatrix of $\mathbf{A}$ formed by deleting some of the columns and some of the rows of $\mathbf{A}$. The number of rows deleted must be equal to the number of columns deleted, which can be 0, i.e., deleting nothing.

Definition 3.32. (Principal minor) A principal minor of $\mathbf{A}$ is a minor of $\mathbf{A}$ with the additional condition that the indices of the deleted rows must be the same as those of deleted columns.

Definition 3.33. (Leading principal minor) A leading principal minor of $\mathbf{A}$ is a principal minor of $\mathbf{A}$ with the additional condition that the indices of the deleted rows must be a consecutive integer sequence starting from 1.

Remark 3.34. These definitions were not given by the lecturer but some online sources. The terms or the exact definitions (which the lecturer actually uses) may differ slightly from these ones.

4. Iterative methods

From this section onwards, let $\mathbf{Ax} = \mathbf{b}$ be a linear system which we want to solve, where $\mathbf{A}$ is a square matrix of dimension n.

4.1. **Motivation.**

Sometimes, it is computationally expensive to solve a linear system for exact solutions. Furthermore, it is usually impossible for us to get exact solutions (or the reader may try to give an exact value of π and $\sqrt{2}$).

Given that in the real-life, most of the time, we just need a fairly accurate solution to linear systems, we wonder whether we can find a method that is computationally desirable. By the term 'fairly accurate', we mean the difference between the solution calculated and the actual solution is small enough (like 10^{-11}). This gives the idea of iterative methods.

4.2. **A general analysis.**

To solve the linear system iteratively, we need to first have an initial guess $\mathbf{x}_0$. Then, each iteration is obtained by

$$\mathbf{x}_{k+1} = \mathbf{B}\mathbf{x}_k + \mathbf{c}$$

for all $k \in \mathbb{Z}_0^+$ until we get an approximation of desired accuracy.

We can obtain $\mathbf{B}$ and $\mathbf{c}$ by splitting $\mathbf{A}$ as two matrices $\mathbf{M}$ and $\mathbf{N}$ such that $\mathbf{A} = \mathbf{M} - \mathbf{N}$, where $\mathbf{M}$ is invertible. This gives us

$$\mathbf{Ax} = \mathbf{b} \implies \mathbf{Mx} = \mathbf{Nx} + \mathbf{b} \implies \mathbf{x} = \mathbf{M}^{-1}\mathbf{Nx} + \mathbf{M}^{-1}\mathbf{b}.$$

We can then conveniently choose $\mathbf{B} = \mathbf{M}^{-1}\mathbf{N}$ and $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$. The reason why we choose to split $\mathbf{A}$ in this way can be seen later.

Definition 4.1. (Stationary point) A stationary point to the recurrence relation $\mathbf{x}_{k+1} = f(\mathbf{x}_k)$ is a value $\mathbf{s}$ such that $\mathbf{s} = \mathbf{f}(\mathbf{s})$.

Lemma 4.2. *If $\boldsymbol{x}_{k+1} = \boldsymbol{B}\boldsymbol{x}_k + \boldsymbol{c}$ has a stationary point $\boldsymbol{s}$, then $\boldsymbol{x} = \boldsymbol{s}$ is a solution to the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$.*

Let $\mathbf{x}$ be the desired solution to the linear system. In the above iteration process, the error of $\mathbf{x}_{k+1}$ can be calculated by

$$\begin{aligned}\mathbf{e}_{k+1} = \mathbf{x}_{k+1} - \mathbf{x} &= \mathbf{M}^{-1}\mathbf{N}\mathbf{x}_k + \mathbf{M}^{-1}\mathbf{b} - \mathbf{x} \\ &= \mathbf{M}^{-1}\mathbf{N}\mathbf{x}_k + \mathbf{M}^{-1}\mathbf{A}\mathbf{x} - \mathbf{x} \\ &= \mathbf{M}^{-1}\mathbf{N}\mathbf{x}_k + (\mathbf{I} - \mathbf{M}^{-1}\mathbf{N})\mathbf{x} - \mathbf{x} \\ &= \mathbf{M}^{-1}\mathbf{N}(\mathbf{x}_k - \mathbf{x}) \\ &= \mathbf{M}^{-1}\mathbf{N}\mathbf{e}_k.\end{aligned}$$

To refine this iteration process, after each iteration, we

(1) compute the residue $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$;
(2) solve the linear system $\mathbf{A}\mathbf{s}_k = \mathbf{r}_k$ by using $\mathbf{A} = \mathbf{L}\mathbf{U}$ decomposition;
(3) take $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{s}_k$.

Theoretically, we can then have

$$\mathbf{A}(\mathbf{x}_k + \mathbf{s}_k) = \mathbf{A}\mathbf{x}_k - \mathbf{A}\mathbf{s}_k = (\mathbf{b} - \mathbf{r}_k) + \mathbf{r}_k = \mathbf{b}.$$

For this method, we take $\mathbf{B} = \mathbf{M}^{-1}\mathbf{N} = \mathbf{I} - (\mathbf{L}\mathbf{U})^{-1}\mathbf{A}$ and $c = \mathbf{M}^{-1}\mathbf{b} = (\mathbf{L}\mathbf{U})^{-1}\mathbf{b}$, where $\mathbf{A} = \mathbf{M} - \mathbf{N}$ and $\mathbf{M} = \mathbf{L}\mathbf{U}$.

The error convergence rate is given by

$$\|\mathbf{e}_{k+1}\| = \left\|(\mathbf{I} - (\mathbf{L}\mathbf{U})^{-1}\mathbf{A})\mathbf{e}_{k+1}\right\| \leq \left\|\mathbf{I} - (\mathbf{L}\mathbf{U})^{-1}\mathbf{A}\right\| \|\mathbf{e}_{k+1}\|.$$

Definition 4.3. (Stationary method) An iteration method is said to be stationary if $\mathbf{B}$ and $\mathbf{c}$ remain constant.

In this week's lecture, our main focus is on various stationary methods. Since the lecturer's notation is not very natural at this stage, the author chooses to use $(\mathbf{D}, \mathbf{U}, \mathbf{L})$ to indicate diagonal, upper triangular, and lower-triangular matrices instead of lecturer's $(\Delta, \Omega, \Lambda)$.

4.3. **Richardson method.**

In Richardson method, we take $\mathbf{M} = \frac{1}{\omega}\mathbf{I}$ and $\mathbf{N} = \frac{1}{\omega}\mathbf{I} - \mathbf{A}$, where ω is any real number. The advantage of taking this $\mathbf{M}$ is that it can be inverted very easily, i.e., $\mathbf{M}^{-1} = \omega\mathbf{I}$.

Under this choice of $\mathbf{M}$ and $\mathbf{N}$, the iteration becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \omega(\mathbf{b} - \mathbf{A}\mathbf{x}_k).$$

The error convergence rate is given by

$$\|\mathbf{e}_{k+1}\| = \|(\mathbf{I} - \omega\mathbf{A})\mathbf{e}_k\| \leq \|\mathbf{I} - \omega\mathbf{A}\| \, \|\mathbf{e}_{k+1}\| .$$

This iteration is useful in finding the least square solution to inconsistent linear systems. For instance, the process of minimising $F(x) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ for some matrix $\mathbf{A}$ and vector $\mathbf{b}$ can be reduced to the Richardson method.

4.4. **Jacobi method.**

Definition 4.4. (Strictly diagonally dominant matrix) Let $\mathbf{A}$ be a $n \times n$ matrix. We say that $\mathbf{A}$ is strictly diagonally dominant if for each $1 \leq i \leq n$, $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$.

Theorem 4.5. *If $\boldsymbol{A} = (a_{ij})_{n\times n}$ is strictly diagonally dominant, then*

(1) $\boldsymbol{A}$ *is a nonsingular matrix;*

(2) *for every vector $\boldsymbol{b}$ and every starting guess, the Jacobi method applied to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ converges to the unique solution.*

In Jacobi method, we decompose $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U} = \mathbf{D} - (-\mathbf{L} - \mathbf{U}),$$

where $\mathbf{D}$ is diagonal, $\mathbf{U}$ is upper triangular with diagonal entries being 0, and $\mathbf{L}$ is lower triangular with all diagonal entries being 0.

Then, we have

$$\mathbf{D}\mathbf{x} = \mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x} \implies \mathbf{x} = \mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{x}).$$

Basically, this decomposition says that for the i-th equation, we move everything except the i-th variables to the RHS of the equation.

Example 4.6. Consider the linear system

$$\begin{cases} 2x+y=1 \\ x+2y=2 \end{cases} \iff \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \mathbf{x} = \mathbf{b}.$$

Applying Jacobi method to this linear system gives us

$$\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \mathbf{x} = \mathbf{b} - \left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \right] \mathbf{x} \iff \begin{cases} 2x = 1-y \\ 2y = 2-x \end{cases};$$

$$\mathbf{x} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \left[\mathbf{b} - \left[\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \right] \mathbf{x} \right] \iff \begin{cases} x = \frac{1-y}{2} \\ y = \frac{2-x}{2} \end{cases}.$$

The error analysis for this method is given by

$$\|\mathbf{e}_{k+1}\| = \left\|(\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U}))\mathbf{e}_{k+1}\right\| \leq \left\|\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\right\| \|\mathbf{e}_{k+1}\|.$$

4.5. **Gauss-Seidel method.**

Theorem 4.7. *If $\boldsymbol{A} = (a_{ij})_{n\times n}$ is strictly diagonally dominant, then*

(1) $\boldsymbol{A}$ *is a nonsingular matrix;*

(2) *for every vector $\boldsymbol{b}$ and every starting guess, the Gauss-Seidel method applied to $\boldsymbol{Ax} = \boldsymbol{b}$ converges to the unique solution.*

Similar to Jacobi method, we also decompose $\mathbf{A}$ as $\mathbf{A} = \mathbf{D}+\mathbf{L}+\mathbf{U}$. Yet, this times, we write

$$\mathbf{A} = (\mathbf{D}+\mathbf{L}) - \mathbf{U}.$$

The iteration thus becomes

$$\mathbf{x}_{k+1} = (\mathbf{D}+\mathbf{L})^{-1}(\mathbf{b} - \mathbf{U}\mathbf{x}_k).$$

Example 4.8. Consider the linear system

$$\begin{cases} 2x+y=1 \\ \\ x+2y=2 \end{cases} \iff \begin{pmatrix} 2 & 1 \\ \\ 1 & 2 \end{pmatrix} \mathbf{x} = \mathbf{b}.$$

Applying Gauss-Seidel method to this linear system gives us

$$\begin{pmatrix} 2 & 0 \\ \\ 1 & 2 \end{pmatrix} \mathbf{x} = \mathbf{b} - \begin{pmatrix} 0 & 1 \\ \\ 0 & 0 \end{pmatrix} \mathbf{x} \iff \begin{cases} 2x = 1-y \\ \\ x+2y=2 \end{cases};$$

$$\mathbf{x} = \begin{pmatrix} \frac{1}{2} & 0 \\ \\ -\frac{1}{4} & \frac{1}{2} \end{pmatrix} \left[\mathbf{b} - \begin{pmatrix} 0 & 1 \\ \\ 0 & 0 \end{pmatrix} \mathbf{x} \right] \iff \begin{cases} x = \frac{1-y}{2} \\ \\ y = -\frac{1-y}{4} + 1 \end{cases}.$$

Here, it is to be noted that instead of calculating $y = -\frac{x}{2} + 1$ as per in Jacobi method, we calculate

$$y = -\frac{1-y}{4} + 1 = -\frac{1}{2}\left(\frac{1-y}{2}\right) + 1 = -\frac{1}{2}x' + 1,$$

where x' refers to the updated value of x.

From this example, we know that the idea of Gauss-Seidel method is to use the updated values of $x_1, x_2, \ldots, x_i$ to calculate the next iteration of x_{i+1}, where x_i refers to the i-th variable. The error analysis of Gauss-Seidel method is given by

$$\|\mathbf{e}_{k+1}\| = \left\|[(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}]\mathbf{e}_k\right\| \leq \left\|(\mathbf{D}+\mathbf{L})^{-1}\mathbf{U}\right\| \|\mathbf{e}_k\|.$$

4.6. **Successive over-relaxation.**

Sometimes, we want to get a balance between the previous guesses and the new guess by introducing a new relaxation parameter ω.

Let $\mathbf{x}_k$ be the k-th iteration, and $\mathbf{x}'_{k+1}$ be the $(k+1)$-th iteration provided by Gauss-Seidel method. We take the actual $(k+1)$-th iteration by

$$\mathbf{x}_{k+1} = (1-\omega)\mathbf{x}_k + \omega\mathbf{x}'_{k+1}.$$

Sometimes, we try to overshoot the solution a bit by taking $\omega > 1$. This method is called successive over-relaxation (SOR).

Under SOR, we have

$$\begin{aligned}(\mathbf{D}+\mathbf{L}+\mathbf{U})\mathbf{x}=\mathbf{b} &\implies \omega(\mathbf{D}+\mathbf{L}+\mathbf{U})\mathbf{x}=\omega\mathbf{b}\\ &\implies (\mathbf{D}+\omega\mathbf{L})\mathbf{x}=\omega\mathbf{b}-[\omega\mathbf{U}+(\omega-1)\mathbf{D}]\mathbf{x}.\end{aligned}$$

This leads to the iteration formula:

$$\mathbf{x}_{k+1}=(\mathbf{D}+\omega\mathbf{L})^{-1}\Big[\omega\mathbf{b}-[\omega\mathbf{U}+(\omega-1)\mathbf{D}]\mathbf{x}_k\Big].$$

With a similar idea, we can also obtain a damped Jacobi method by taking

$$\mathbf{A}=\left(\frac{1}{\omega}\mathbf{D}\right)+\left[\left(1-\frac{1}{\omega}\right)\mathbf{D}+\mathbf{L}+\mathbf{U}\right].$$

4.7. **Conjugate gradient.**

If $\mathbf{A}$ is symmetric positive-definite, then define

$$F(\mathbf{x})=\frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x}-\mathbf{x}^T\mathbf{b},$$

which attains its minimum at $\mathbf{x}$ satisfying $\mathbf{A}\mathbf{x}=\mathbf{b}$.

Definition 4.9. ($\mathbf{A}$-inner product) Let $\mathbf{A}$ be a symmetric positive-definite $n\times n$ matrix. For two n-vectors $\mathbf{v}$ and $\mathbf{w}$, we define the $\mathbf{A}$-inner product by

$$(\mathbf{v},\mathbf{w})_{\mathbf{A}}=\mathbf{v}^T\mathbf{A}\mathbf{w}.$$

If $(\mathbf{v},\mathbf{w})_{\mathbf{A}}=0$, we say that $\mathbf{v}$ and $\mathbf{w}$ are $\mathbf{A}$-conjugate.

Let $\mathbf{x}_0$ be the initial guess, $\mathbf{r}_0=\mathbf{b}-\mathbf{A}\mathbf{x}_0$ be the initial residual, and $\mathbf{d}_0=\mathbf{b}-\mathbf{A}\mathbf{x}_0$ be the initial direction. Whenever $\mathbf{r}_k\neq 0$, we conduct

(1) $\alpha_k=\frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{d}_k^T\mathbf{A}\mathbf{d}_k}$;
(2) $\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{d}_k$;
(3) $\mathbf{r}_{k+1}=\mathbf{r}_k-\alpha_k\mathbf{A}\mathbf{d}_k$;
(4) $\beta_k=\frac{\mathbf{r}_{k+1}^T\mathbf{r}_{k+1}}{\mathbf{r}_k^T\mathbf{r}_k}$; and

(5) $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$.

A detailed explanation of how the whole process works can be found in the reference text. The idea is similar to Gram-Schmidt process.

Theorem 4.10. *Let $\boldsymbol{A}$ be a symmetric positive-definite $n \times n$ matrix and let $\boldsymbol{b} \neq \boldsymbol{0}$ be a vector. In the Conjugate Gradient Method, assume that $\boldsymbol{r}_k \neq \boldsymbol{0}$ for $k < n$. Then for each $1 \leq k \leq n$,*

(a) the following three subspaces of $\mathbb{R}^n$ are equal:

$$\langle \boldsymbol{x}_1, \ldots, \boldsymbol{x}_k \rangle = \langle \boldsymbol{r}_0, \ldots, \boldsymbol{r}_{k-1} \rangle = \langle \boldsymbol{x}_0, \ldots, \boldsymbol{x}_{k-1} \rangle;$$

(b) the residuals $\boldsymbol{r}_k$ are pairwise orthogonal: $\boldsymbol{r}_k^T \boldsymbol{r}_j = \boldsymbol{0}$ for $j < k$;

(c) the directions $\boldsymbol{d}_k$ are pairwise $\boldsymbol{A}$-conjugate: $\boldsymbol{d}_k^T \boldsymbol{A} \boldsymbol{d}_j = \boldsymbol{0}$ for $j < k$.

4.8. **Convergence.**

Suppose the iterative model is $\mathbf{x}_{k+1} = \mathbf{B}\mathbf{x}_k + \mathbf{c}$.

Lemma 4.11. *If $\|\boldsymbol{B}\| < 1$, then $\boldsymbol{I} - \boldsymbol{B}$ is invertible and its inverse is given by*

$$(\boldsymbol{I} - \boldsymbol{B})^{-1} = \sum_{k=0}^{\infty} \boldsymbol{B}^k.$$

Proof. The idea is similar to finding the sum of a geometric sequence:

$$(\mathbf{I} - \mathbf{B}) \sum_{k=0}^{\infty} \mathbf{B}^k = \lim_{n \to \infty} \left(\sum_{k=0}^{n} \mathbf{B}^k - \sum_{k=0}^{n} \mathbf{B}^{k+1} \right) = \lim_{n \to \infty} \left(\mathbf{I} - \mathbf{B}^{n+1} \right) = \mathbf{I}.$$

We have $\mathbf{B}^\infty = \mathbf{0}$ because $\|\mathbf{B}\|^\infty = v^\infty = 0$ for some $v \in \mathbb{R}$ smaller than 1. □

Proposition 4.12. *If $\|\boldsymbol{B}\| < 1$, then $\boldsymbol{x}_n$ converges to $(\boldsymbol{I} - \boldsymbol{B})^{-1}\boldsymbol{c}$.*

Proof. By staring at the model, we have

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{B}\mathbf{x}_k + \mathbf{c} \\ &= \mathbf{B}(\mathbf{B}\mathbf{x}_{k-1} + \mathbf{c}) + \mathbf{c} \\ &= \mathbf{B}(\mathbf{B}(\mathbf{B}\mathbf{x}_{k-2} + \mathbf{c}) + \mathbf{c}) + \mathbf{c} \\ &\vdots \\ &= \mathbf{B}^{k+1}\mathbf{x}_0 + \sum_{r=0}^{k} \mathbf{B}^r \mathbf{c}.\end{aligned}$$

Since $\|\mathbf{B}\| < 1$, we have $\lim_{k\to\infty} \|\mathbf{B}\|^k = 0$ and $\lim_{k\to\infty} \sum_{r=0}^{k} \mathbf{B}^r = (\mathbf{I}-\mathbf{B})^{-1}$, which implies that

$$\lim_{k\to\infty} \mathbf{x}_{k+1} = \lim_{k\to\infty} \left(\mathbf{B}^{k+1}\mathbf{x}_0 + \sum_{r=0}^{k} \mathbf{B}^r \mathbf{c} \right) = \mathbf{0} + (\mathbf{I}-\mathbf{B})^{-1}\mathbf{c} = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{c}.$$

□

Definition 4.13. (Spectral radius) Let $\mathbf{B}$ be a matrix. The spectral radius of $\mathbf{B}$ is defined as

$$\rho(\mathbf{B}) = \max |\lambda_i|.$$

Remark 4.14. We are interested in the spectral radius of $\mathbf{B}$ because

$$|\lambda| \, \|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|\mathbf{B}\mathbf{x}\| \leq \|\mathbf{B}\| \, \|\mathbf{x}\| .$$

With the above ideas, we can prove theorem 4.5 and theorem 4.7. The detailed proofs can be found in the chapter 2.5.3 of the reference text.

4.9. **Preconditioning.**

Definition 4.15. (Preconditioner) A preconditioner to a $n \times n$ linear system is an invertible $n \times n$ matrix $\mathbf{M}$. The preconditioned form of linear system is $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$.

Remark 4.16. An effective preconditioner reduces the condition number of the problem by attempting to invert $\mathbf{A}$.

There are some examples of preconditioners:

Example 4.17. (Jacobi preconditioner) We choose $\mathbf{M} = \mathbf{D}$, where $\mathbf{D}$ is the diagonal of $\mathbf{A}$.

Example 4.18. When $\mathbf{A}$ is a symmetric positive-definite $n \times n$ matrix, we will choose a symmetric positive-definite matrix $\mathbf{M}$ as a preconditioner. The idea behind is to replace the system $\mathbf{Ax} = \mathbf{b}$ in Conjugate Gradient Method to $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$.

Example 4.19. (SSOR preconditioner) The symmetric successive over-relaxation preconditioner is defined by

$$\mathbf{M} = (\mathbf{D} + \omega\mathbf{L})\mathbf{D}^{-1}(\mathbf{D} + \omega\mathbf{U}),$$

where $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$. The special case $\omega = 1$ is called the Gauss-Seidel preconditioner.

5. Least Squares Methods

5.1. **Motivation.**

Let $\mathbf{Ax} = \mathbf{b}$ be linear system, where $\mathbf{A}$ is a $m \times n$.

Definition 5.1. (Over-determined systems) We say that a linear system is over-determined if $m > n$. This means, we have more equations than variables.

Definition 5.2. (Under-determined systems) We say that a linear system is under-determined if $n < m$. This means, we have less equations than variables.

Definition 5.3. (Consistency) We say that a system $\mathbf{Ax} = \mathbf{b}$ is consistent iff it has a solution. Otherwise, the system is inconsistent.

Remark 5.4. An over-determined system usually has no solution, while an under-determined system usually has infinitely many solutions.

If a system is inconsistent, then we are interested in what the minimum of $\|\mathbf{Ax} - \mathbf{b}\|$ can be. If a system has infinitely many solutions, we are interested in what the minimum of $\|\mathbf{x}\|$ is subject to $\mathbf{Ax} = \mathbf{b}$.

Also, suppose we have a system linear in the coefficients

$$y_i = \beta_k f_k(x_i) + \ldots + \beta_1 f_1(x_i) + \beta_0 f_0(x_i) + \varepsilon_i$$

for $1 \leq i \leq n$. We can write it as

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} f_k(x_1) & \ldots & f_1(x_1) & f_0(x_1) \\ f_k(x_2) & \ldots & f_1(x_2) & f_0(x_2) \\ \vdots & \ddots & \vdots & \vdots \\ f_k(x_n) & \ldots & f_1(x_n) & f_0(x_n) \end{pmatrix} \begin{pmatrix} \beta_k \\ \beta_1 \\ \vdots \\ \beta_0 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix} \iff \mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

which is also referred to as a linear model. We are interested in the question that given a set of $\mathbf{X}$ and the corresponding $\mathbf{Y}'$, what values of $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ we shall use such that $\|\mathbf{Y} - \mathbf{Y}'\|$ is minimised.

5.2. **Quadratic minimisation.**

For a general quadratic equation of n variables, we can write it as $q(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x} - 2\mathbf{x}^T\mathbf{v} + c$, where $\mathbf{x}, \mathbf{v}$ are column vectors of length n, c is a constant, and $\mathbf{M}$ is a symmetric positive-definite matrix of dimension n.

Example 5.5. Let $q(x, y) = x^2 + 3xy - y^2 + x + y - 2$. We can write it as

$$q(\mathbf{x}) = \mathbf{x}^T \begin{pmatrix} 1 & \frac{3}{2} \\ \frac{3}{2} & -1 \end{pmatrix} \mathbf{x} - 2\mathbf{x}^T \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \end{pmatrix} + (-2),$$

where $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$.

Example 5.6. Let $q(x, y, z) = x^2 - y^2 + z^2 + 3xy + 2xz + x + 3y + z + 2$. We can write it as

$$q(\mathbf{x}) = \mathbf{x}^T \begin{pmatrix} 1 & \frac{3}{2} & 1 \\ \frac{3}{2} & -1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \mathbf{x} - 2\mathbf{x}^T \begin{pmatrix} -\frac{1}{2} \\ -\frac{3}{2} \\ -\frac{1}{2} \end{pmatrix} + 2.$$

Remark 5.7. Generally speaking, the (i, j)-th entry of $\mathbf{M}$ stores half of the coefficient of $x_i x_j$ in $q(\mathbf{x})$ for $x \neq j$, and the (i, i)-th entry of $\mathbf{M}$ stores the coefficient of x_i^2 in $q(\mathbf{x})$.

We now consider a quadratic expression $q(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x} - 2\mathbf{x}^T\mathbf{v} + c$. With the above discussion, we know that $\mathbf{M}$ is symmetric positive-definite. Now, suppose $\mathbf{y}$ is a solution to the linear system $\mathbf{M}\mathbf{y} = \mathbf{v}$. Noticing that $\mathbf{x}^T\mathbf{v}$ is essentially a 1×1 matrix,

$$\begin{aligned}
q(\mathbf{x}) = \mathbf{x}^T\mathbf{M}\mathbf{x} - 2\mathbf{x}^T\mathbf{v} + c &= \mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{v} - \mathbf{v}^T\mathbf{x} + c && (\mathbf{x}^T\mathbf{v} = \mathbf{v}^T\mathbf{x}) \\
&= \mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{M}\mathbf{y} - \mathbf{y}^T\mathbf{M}^T\mathbf{x} + c && ((\mathbf{M}\mathbf{y})^T = \mathbf{y}^T\mathbf{M}^T) \\
&= \mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{M}\mathbf{y} - \mathbf{y}^T\mathbf{M}\mathbf{x} + c && (\mathbf{M} = \mathbf{M}^T) \\
&= \mathbf{x}^T\mathbf{M}(\mathbf{x} - \mathbf{y}) - \mathbf{y}^T\mathbf{M}\mathbf{x} + c + \mathbf{y}^T\mathbf{M}\mathbf{y} - \mathbf{y}^T\mathbf{M}\mathbf{y} \\
&= \mathbf{x}^T\mathbf{M}(\mathbf{x} - \mathbf{y}) - \mathbf{y}^T\mathbf{M}(\mathbf{x} - \mathbf{y}) - \mathbf{y}^T\mathbf{M}\mathbf{y} + c \\
&= (\mathbf{x} - \mathbf{y})^T\mathbf{M}(\mathbf{x} - \mathbf{y}) + \left[c - \mathbf{y}^T\mathbf{M}\mathbf{y}\right], && (\mathbf{x}^T - \mathbf{y}^T = (\mathbf{x} - \mathbf{y})^T)
\end{aligned}$$

where $c - \mathbf{y}^T\mathbf{M}\mathbf{y}$ is a constant. It is to be noted that this expression is well-defined in the sense that it is independent from the choice of $\mathbf{y}$ if $\mathbf{M}\mathbf{y} = \mathbf{v}$ has multiple solutions. This is because if $\mathbf{M}\mathbf{y} = \mathbf{v} = \mathbf{M}\mathbf{z}$, we have

$$\mathbf{y}^T\mathbf{M}\mathbf{y} = \mathbf{y}^T\mathbf{M}\mathbf{z} = (\mathbf{M}\mathbf{y})^T\mathbf{z} = (\mathbf{M}\mathbf{z})^T\mathbf{z} = \mathbf{z}^T\mathbf{M}\mathbf{z}.$$

Moreover, we have $\mathbf{z}^T\mathbf{M}\mathbf{z} \geq 0$. Hence, by taking $\mathbf{x} = \mathbf{y}$, we obtained the minimum of $q(\mathbf{x})$.

5.3. **Least square solution - 1.**

Definition 5.8. (Least square solution) Let $\mathbf{A}\mathbf{x} = \mathbf{b}$ be an inconsistent linear system. The least square solution $\overline{\mathbf{x}}$ to the linear system is the solution which minimises $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|$.

Theorem 5.9. *For any inconsistent linear systems $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ such that $\boldsymbol{A}$ is of full rank, there exists a unique least square solution $\overline{\boldsymbol{x}}$. Namely, $\overline{\boldsymbol{x}}$ is the unique solution of $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T\boldsymbol{b}$.*

Proof. Our proof consists of proving 1) the existence of $\overline{\mathbf{x}}$ and 2) the uniqueness of $\overline{\mathbf{x}}$.

Lemma 5.10. *Let $\boldsymbol{v}$ be arbitrary vector and its 2-norm be $\|\boldsymbol{v}\|_2$. We have $\boldsymbol{v}^T\boldsymbol{v} = \|\boldsymbol{v}\|_2^2 = (\|\boldsymbol{v}\|_2)^2$.*

Lemma 5.11. *Suppose $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two column vectors of length n whose entries are all from a commutative ring. We have $\boldsymbol{v}_1^T\boldsymbol{v}_2 = (\boldsymbol{v}_1^T\boldsymbol{v}_2)^T = \boldsymbol{v}_2^T\boldsymbol{v}_1$.*

Remark 5.12. A commutative ring is a special mathematical structure on which binary operations are commutative. For instance, $\mathbb{R}$ with the usual addition and multiplication is a commutative ring since $a \times b = b \times a$ is true for all $a, b \in \mathbb{R}$.

To prove the existence of $\overline{\mathbf{x}}$, we take $\mathbf{M} = \mathbf{A}^T\mathbf{A}$, $\mathbf{v} = \mathbf{A}^T\mathbf{b}$, and $c = \mathbf{b}^T\mathbf{b}$. It is easy to verify that $\mathbf{M}$ is a symmetric positive-definite matrix, $\mathbf{v}$ is a column vector, and c is a constant. We

can then have a quadratic expression

$$\begin{aligned} q(\mathbf{x}) &= \mathbf{x}^T\mathbf{M}\mathbf{x} - 2\mathbf{x}^T\mathbf{v} + c \\ &= \mathbf{x}^T\mathbf{M}\mathbf{x} - \mathbf{x}^T\mathbf{v} - \mathbf{v}^T\mathbf{x} + c \\ &= \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{b} \\ &= (\mathbf{x}^T\mathbf{A}^T - \mathbf{b}^T)(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= (\mathbf{A}\mathbf{x} - \mathbf{b})^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 . \end{aligned}$$

Since in the previous section, we have proven that $q(\mathbf{x})$ achieves its minimum at $\mathbf{x} = \mathbf{y}$, where $\mathbf{y}$ is a solution to the linear system $\mathbf{M}\mathbf{y} = \mathbf{v} \implies \mathbf{A}^T\mathbf{A}\mathbf{y} = \mathbf{A}^T\mathbf{b}$, we only need to prove that $\mathbf{A}^T\mathbf{A}\mathbf{y} = \mathbf{A}^T\mathbf{b}$ is consistent, which is true since $\text{range}(\mathbf{A}^T\mathbf{A}) = \text{range}(\mathbf{A}^T)$. Therefore, there exists $\mathbf{y}'$ such that $\mathbf{A}^T\mathbf{A}\mathbf{y}' = \mathbf{A}^T\mathbf{b}$. By taking $\overline{\mathbf{x}} = \mathbf{y}'$, we show the existence of $\overline{\mathbf{x}}$.

We now prove the uniqueness of $\overline{\mathbf{x}}$. Since $\mathbf{A}$ is full rank, then $\mathbf{A}^T\mathbf{A}$ is invertible, which suggests that the linear system $\mathbf{A}^T\mathbf{A}\mathbf{y}' = \mathbf{A}^T\mathbf{b}$ has a unique solution, i.e., $\mathbf{y}' = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$. This implies the uniqueness of $\overline{\mathbf{x}}$ to the original linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$. □

5.4. **Sensitivity and condition number.**

Previously, the condition number for solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ is given by

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \left\|\mathbf{A}^{-1}\right\| ,$$

where $\mathbf{A}$ is an invertible matrix.

Now, since $\mathbf{A}$ is no longer invertible, we define

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|_2 \cdot \left\|\mathbf{A}^{\dagger}\right\|_2 .$$

The sensitivity of the problem depends on $\mathbf{b}$, as the larger the residual is, the more sensitive the problem is.

5.5. **Error analysis.**

Since the accuracy of the least square solution to the linear system $\mathbf{Ax} = \mathbf{b}$ can be affected by the errors in $\mathbf{A}$ and $\mathbf{b}$, we now analyse how small perturbations in $\mathbf{A}$ and $\mathbf{b}$ affect the error in $\mathbf{x}$ respectively.

Suppose now $\mathbf{b}' = \mathbf{b} + \Delta\mathbf{b}$. We have

$$\begin{aligned}\mathbf{A}^T\mathbf{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{A}^T(\mathbf{b} + \Delta\mathbf{b}) \implies \mathbf{A}^T\mathbf{A}\Delta\mathbf{x} &= \mathbf{A}^T\Delta\mathbf{b} \\ \Delta\mathbf{x} &= \mathbf{A}^\dagger\Delta\mathbf{b},\end{aligned}$$

which then gives

$$\begin{aligned}\frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \frac{\left\|\mathbf{A}^\dagger\Delta\mathbf{b}\right\|_2}{\|\mathbf{x}\|_2} &\leq \left\|\mathbf{A}^\dagger\right\| \frac{\|\Delta\mathbf{b}\|_2}{\|\mathbf{x}\|_2} \\ &= \kappa(\mathbf{A})\frac{\|\mathbf{b}\|_2}{\|\mathbf{A}\|_2\|\mathbf{x}\|_2}\frac{\|\Delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}.\end{aligned}$$

Geometrically, $\mathbf{A}\overline{\mathbf{x}}$ is the projection of $\mathbf{Ax}$ onto $\mathbf{b}$, so we can define $\cos\theta = \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{b}\|_2} \leq \frac{\|\mathbf{A}\|_2\|\mathbf{x}\|_2}{\|\mathbf{b}\|_2}$, which gives us

$$\frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \frac{\kappa(\mathbf{A})}{\cos\theta}\frac{\|\Delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}.$$

Remark 5.13. There is a slight abuse of symbol, where $\mathbf{x}$ here means $\overline{\mathbf{x}}$ occasionally.

We now analyse the effect of the perturbations on $\mathbf{A}$. Suppose now $\mathbf{A}' = \mathbf{A} + \mathbf{E}$. We have

$$\frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \lessapprox \frac{\|\mathbf{E}\|_2}{\|\mathbf{A}\|_2}\left(\kappa(\mathbf{A})^2\tan\theta + \kappa(\mathbf{A})\right).$$

Remark 5.14. The writer is still trying to derive the above approximation.

5.6. **Some results on linear algebra.**

Lemma 5.15. $\operatorname{ran}(\boldsymbol{A}^T\boldsymbol{A}) = \operatorname{ran}(\boldsymbol{A}^T)$.

Lemma 5.16. *If the columns of $\boldsymbol{A}$ are linearly independent, then $\boldsymbol{A}^T\boldsymbol{A}$ is invertible.*

Remark 5.17. With the above two lemmas, we can conclude the existence and uniqueness of the least square solution. Namely, the first lemma guarantees the existence of solution, and the second lemma (assume that the system is over-determined) guarantees the uniqueness of the solution.

5.7. **Orthogonality.**

Definition 5.18. (Orthogonality) Two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are said to be orthogonal to each other if

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i = 0.$$

Definition 5.19. (Angle) Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be two vectors. The angle between these two vectors is defined as

$$\theta = \cos^{-1}\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \, \|\mathbf{y}\|} \right).$$

Remark 5.20. In $\mathbb{R}^2$ (the normal plane), the above definition is consistent with the law of cosines:

$$\|\mathbf{x} - \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - 2\|\mathbf{x}\| \, \|\mathbf{y}\| \cos\theta \iff c^2 = x^2 + y^2 - 2xy\cos\theta.$$

Definition 5.21. (Orthogonal complement) Given a subspace U of $\mathbb{R}^n$, its orthogonal complement $U^\perp$ is defined as the set of all $\mathbf{x} \in \mathbb{R}^n$ that are orthogonal to every $\mathbf{u} \in U$.

Example 5.22. $\mathrm{Span}((1,0,0)^T)^\perp = \mathrm{Span}((0,1,0)^T, (0,0,1)^T)$.

Lemma 5.23. *Every $\boldsymbol{x} \in \mathbb{R}^n$ can be written uniquely as $\boldsymbol{x} = \boldsymbol{u} + \boldsymbol{w}$, where $\boldsymbol{u} \in U$ and $\boldsymbol{w} \in U^\perp$.*

Definition 5.24. (Orthogonal projection) The orthogonal projection onto U is a linear map defined by

$$\mathbf{P}(\mathbf{x}) = \mathbf{P}(\mathbf{u} + \mathbf{w}) = \mathbf{u}.$$

Remark 5.25. It is to be noted that $\mathbf{x} - \mathbf{P}(\mathbf{x}) = \mathbf{w} \in U^\perp$, which implies that $\mathbf{x} - \mathbf{P}(\mathbf{x})$ is orthogonal to $\mathbf{P}(\mathbf{x})$.

Definition 5.26. (Projection matrix) Let $\mathcal{B}_U = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k\}$ be a basis for U. We form

$$\mathbf{A} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \ldots & \mathbf{x}_k \\ | & & | \end{pmatrix}.$$

The orthogonal projection onto the range of $\mathbf{A}$ is given by

$$\mathbf{P}(\mathbf{x}) = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\mathbf{x}).$$

Remark 5.27. The idea behind is that, $\mathbf{Ax}$, where $\mathbf{x}$ is the solution to the normal equation $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$, is the projection of $\mathbf{b}$ onto $\text{ran}(\mathbf{A}) = \text{span}(\mathcal{B}_U) = U$. Since $\mathbf{x}$ is given by

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b},$$

pre-multiplying both sides by $\mathbf{A}$ gives

$$\mathbf{Ax} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{P}(\mathbf{b}).$$

Definition 5.28. (Orthogonal matrix) A matrix $\mathbf{Q}$ is said to be orthogonal if $\mathbf{Q}^T\mathbf{Q} = \mathbf{I} = \mathbf{QQ}^T$.

Remark 5.29. It is important to ensure that both $\mathbf{QQ}^T$ and $\mathbf{Q}^T\mathbf{Q}$ equal $\mathbf{I}$ since matrix multiplication is generally not commutative.

Lemma 5.30. *If $\boldsymbol{Q}$ is an orthogonal matrix, then the columns of $\boldsymbol{Q}$ form an orthonormal basis.*

Definition 5.31. (Isometry) If $\mathbf{QQ}^T = \mathbf{I}$, then we say that $\mathbf{Q}$ is an isometry.

Lemma 5.32. *If $\boldsymbol{Q}$ is an isometry, then the orthogonal projection onto* $\text{ran}(\boldsymbol{Q})$ *is $\boldsymbol{QQ}^T$.*

5.8. **Gram-Schmidt Process.**

Theorem 5.33. (Gram-Schmidt Process) *Given a linearly independent set $X := \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, we can construct a linearly independent orthonormal set $Q := \{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n\}$ such that* $\text{Span}(X) =$

Span(Q) *by setting*

$$\boldsymbol{y}_1 = \boldsymbol{x}_1 \quad \boldsymbol{q}_1 = \frac{\boldsymbol{y}_1}{\|\boldsymbol{y}_1\|_2};$$
$$\boldsymbol{y}_j = \boldsymbol{x}_j - \sum_{i=1}^{j-1} \langle \boldsymbol{q}_i, \boldsymbol{x}_j \rangle \boldsymbol{q}_i \quad \boldsymbol{q}_j = \frac{\boldsymbol{y}_j}{\|\boldsymbol{y}_j\|_2} \text{ for } j \geq 2.$$

Remark 5.34. When calculating $\mathbf{y}_j$, we ensure that $\mathbf{y}_j$ is orthogonal to all $\mathbf{q}_i$ for $1 \leq i < j$ by subtracting the projections of $\mathbf{x}_j$ onto $\mathbf{q}_i$ for $1 \leq i < j$. Then, we normalise $\mathbf{y}_j$ by dividing it with its norm.

Lemma 5.35. *Alternatively, if* $U_{j-1} = \text{Span}(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{j-1})$ *and* $\boldsymbol{P}_{j-1}$ *is the orthogonal projection onto* U_{j-1}*, we have* $\boldsymbol{y}_j = \boldsymbol{x}_j - \boldsymbol{P}_{j-1}\boldsymbol{x}_j$.

Lemma 5.36. *Alternatively, if* $\boldsymbol{Q}_{j-1}$ *is the matrix whose columns are* $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{j-1}$*, then* $\boldsymbol{y}_j = \boldsymbol{x}_j - (\boldsymbol{Q}_{j-1}\boldsymbol{Q}_{j-1}^T)\boldsymbol{x}_j$.

Thinking of matrices, suppose we have

$$\mathbf{A} = \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \ldots & \mathbf{x}_k \\ | & & | \end{pmatrix},$$

whose columns are all linearly independent. By Gram-Schmidt process, we have

$$\mathbf{q}_1 = \frac{\mathbf{x}_1}{\|\mathbf{x}_1\|} \iff \mathbf{x}_1 = \|\mathbf{x}_1\|\,\mathbf{q}_1 = \langle \mathbf{q}_1, \mathbf{x}_1 \rangle \mathbf{q}_1.$$

In general, we have

$$\mathbf{x}_j = \sum_{i=1}^{j} \langle \mathbf{q}_i, \mathbf{x}_j \rangle \mathbf{q}_i,$$

which allows us to have the **QR** factorisation.

Theorem 5.37. (**QR** factorisation) *Let $\boldsymbol{A}$ be a square matrix of dimension k whose columns are linearly independent, we can write*

$$\boldsymbol{A} = \boldsymbol{QR} \iff \begin{pmatrix} | & & | \\ \boldsymbol{x}_1 & \dots & \boldsymbol{x}_k \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \boldsymbol{q}_1 & \dots & \boldsymbol{q}_k \\ | & & | \end{pmatrix} \begin{pmatrix} \langle \boldsymbol{q}_1, \boldsymbol{x}_1 \rangle & \langle \boldsymbol{q}_1, \boldsymbol{x}_2 \rangle & \dots & \langle \boldsymbol{q}_1, \boldsymbol{x}_k \rangle \\ 0 & \langle \boldsymbol{q}_2, \boldsymbol{x}_2 \rangle & \dots & \langle \boldsymbol{q}_2, \boldsymbol{x}_k \rangle \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \langle \boldsymbol{q}_k, \boldsymbol{x}_k \rangle \end{pmatrix},$$

where $\boldsymbol{Q}$ is an orthogonal matrix and $\boldsymbol{R}$ is an upper triangular matrix.

5.9. **The real spectral theorem.**

Let $\mathbf{V}$ be a vector over field $\mathbf{F}$.

Definition 5.38. (Eigenspace) Let $\mathbf{A}$ be (a representation) of a linear transformation (from V to itself). We define

$$\mathbf{E}_\lambda := \{\mathbf{x} \in V : \mathbf{A}\mathbf{x} = \lambda \mathbf{x}\}$$

for all $\lambda \in \mathbf{F}$. If $\mathbf{E}_\lambda \neq \{\mathbf{0}_V\}$, i.e., there exists some non-zero vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$, we say that $\mathbf{E}_\lambda$ is an eigenspace of $\mathbf{A}$. In addition, we say that λ is an eigenvalue of $\mathbf{A}$, and all non-zero vectors in $\mathbf{E}_\lambda$ are eigenvectors of $\mathbf{A}$.

Remark 5.39. It is possible for a matrix to have no real eigenvalues.

Lemma 5.40. *If $\boldsymbol{A}$ is symmetric, then all eigenvalues of $\boldsymbol{A}$ must be real.*

Lemma 5.41. *If $\boldsymbol{A}$ is symmetric, then we can select some eigenvectors of $\boldsymbol{A}$ to form an orthonormal basis of* $\operatorname{ran}(\boldsymbol{A})$.

Theorem 5.42. (The real spectral theorem) *If $\boldsymbol{A}$ is symmetric and has eigenvalues $\lambda_1, \dots, \lambda_n$, then we can select some orthonormal eigenvectors $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ corresponding to eigenvalues $\lambda_1, \dots, \lambda_n$ to form $\boldsymbol{Q}$ and Λ such that*

$$\boldsymbol{Q}^T \boldsymbol{A}\, \boldsymbol{Q} = \boldsymbol{\Lambda} \iff \boldsymbol{A} = \boldsymbol{Q}\boldsymbol{\Lambda}\, \boldsymbol{Q}^T,$$

where

$$\boldsymbol{Q} = \begin{pmatrix} | & & | \\ \boldsymbol{x}_1 & \dots & \boldsymbol{x}_n \\ | & & | \end{pmatrix}, \quad \boldsymbol{\Lambda} = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}.$$

5.10. **Singular values and SVD.**

Let $\mathbf{A}$ be an $m \times n$ real matrix. From the previous lectures, we know that $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are both symmetric positive semi-definite matrices. Therefore, they both have non-negative real eigenvalues.

Now, suppose $\mathbf{A}^T\mathbf{A}$ has eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, and $\mathbf{A}\mathbf{A}^T$ has eigenvalues $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_m \geq 0$. Let $k = \min(m, n)$.

Lemma 5.43. *If $i \leq k$, then $\lambda_i = \mu_i$. If $i > k$, the remaining eigenvalues are all* 0.

With the above lemma, we can focus on eigenvalues λ_i for $1 \leq i \leq k$.

Definition 5.44. (Singular values) The singular values of $\mathbf{A}$ are defined by

$$\sigma_i = \sqrt{\lambda_i}$$

for $1 \leq i \leq k$.

Let $\boldsymbol{\Sigma}$ be the $m \times n$ diagonal matrix whose diagonal entries are $\sigma_1, \ldots, \sigma_k$. It is to be noted that $\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}}$ is an $m \times m$ matrix whose diagonal entries are $\mu_1, \ldots, \mu_m$, and $\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{\Sigma}$ is an $n \times n$ matrix whose diagonal entries are $\lambda_1, \ldots, \lambda_n$.

Lemma 5.45. *There exist orthogonal matrices $\boldsymbol{V}$ and $\boldsymbol{U}$ such that*

$$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{V}\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{\Sigma}\,\boldsymbol{V}^T, \quad \boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}}\,\boldsymbol{U}^T.$$

Proof. This is a direct implication of the real spectral theorem. □

Theorem 5.46. (Singular value decomposition) *We have $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\,\boldsymbol{V}^T$.*

5.11. **Pseudo-inverse.**

Given that most of the matrices are not invertible, we want to find something which has a similar property to inverse matrix.

Suppose $\mathbf{A}$ has independent columns. Since $\mathbf{A}$ has independent columns implies that $\mathbf{A}^T\mathbf{A}$ is invertible, we can define the pseudo-inverse of $\mathbf{A}$ as

$$\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T.$$

With this definition, we have $\mathbf{A}^\dagger\mathbf{A} = \mathbf{I}$. This is useful in solving the normal equations:

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \iff \mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^\dagger\mathbf{b}.$$

If we write $\mathbf{A} = \mathbf{QR}$, then we have

$$\mathbf{A}^T\mathbf{A} = \mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R} = \mathbf{R}^T\mathbf{R}$$

since $\mathbf{Q}$ is orthogonal, i.e., $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$. This translates the question into finding the Cholesky factorisation.

For general $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, we take

$$\mathbf{A}^\dagger = \mathbf{V}\mathbf{\Sigma}^\dagger\mathbf{U}^T.$$

5.12. **Least square solution - 2.**

When the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ has no solution, we want to minimise $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ instead. By the previous discussion, this amounts to solving the normal equation $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$. One way to do so is to perform Cholesky factorisation as $\mathbf{A}^T\mathbf{A}$ is a symmetric positive-definite matrix.

However, solving $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ may not be ideal since the question can be ill-conditioned. One such example can be seen from the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ \varepsilon & 0 \\ 0 & \varepsilon \end{pmatrix} \implies \mathbf{A}^T\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Another reason is that the condition number $\kappa(\mathbf{A}^T\mathbf{A}) = \kappa(\mathbf{A})$ can be quite large.

Therefore, we need some other methods to find the least square solution to the linear system $\mathbf{Ax} = \mathbf{b}$. When $\mathbf{A}$ is sparse, we can consider the augmented system

$$\begin{cases} \mathbf{r} + \mathbf{Ax} = \mathbf{b} \\ \mathbf{A}^T\mathbf{r} = \mathbf{0} \end{cases},$$

which translates the question of solving

$$\begin{pmatrix} \mathbf{I} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{r} \\ \mathbf{x} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \end{pmatrix}.$$

5.13. **Triangular least squares.**

From our previous experience, triangular systems can be solved relatively easily. In general, we can write triangular over-determined system in the form of

$$\begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix},$$

where $\mathbf{R}$ is an upper triangular matrix.

Example 5.47. Consider the system

$$(\mathbf{A}|\mathbf{b}) = \left(\begin{array}{cc|c} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \end{array}\right).$$

Clearly, the system is in REF form and has no solution. As such, we can define

$$\mathbf{R} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \mathbf{c}_1 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \mathbf{c}_2 = \begin{pmatrix} 3 \end{pmatrix}$$

and rewrite the system as

$$\begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix}.$$

The residual of the system thus becomes

$$\|\mathbf{r}\|_2^2 = \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 = \|\mathbf{c}_1 - \mathbf{R}\mathbf{x}\|_2^2 + \|\mathbf{c}_2\|_2^2,$$

whose minimum is achieved when $\mathbf{R}\mathbf{x} = \mathbf{c}_1$.

5.14. **QR transformation.**

When converting $\mathbf{A}\mathbf{x} = \mathbf{b}$ to an upper triangular system, one common approach is to use $\mathbf{LU}$ factorisation. Yet, the issue with $\mathbf{LU}$ factorisation is that it does not preserve the norm (given that our aim is to minimise $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$). This makes us wonder what property of a matrix $\mathbf{Q}$ would guarantee $\|\mathbf{Q}\mathbf{y}\|_2 = \|\mathbf{y}\|_2$.

Back to the property of norm, we have

$$\|\mathbf{Q}\mathbf{y}\|_2 = \|\mathbf{y}\|_2 \implies (\mathbf{Q}\mathbf{y})^T(\mathbf{Q}\mathbf{y}) = \mathbf{y}^T\mathbf{Q}^T\mathbf{Q}\mathbf{y} = \mathbf{y}^T\mathbf{y},$$

so we need $\mathbf{Q}$ to have the property $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$, i.e., to be orthogonal.

Therefore, we are interested in the $\mathbf{QR}$ factorisation. Suppose $\mathbf{A}$ is an $m \times n$ matrix with independent columns and with $m > n$, we can find $\mathbf{Q}$ and $\mathbf{R}$ such that

$$\mathbf{A} = \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix},$$

where $\mathbf{Q}$ is $m \times m$, $\mathbf{R}$ is $n \times n$, and $\mathbf{0}$ is $(m - n) \times n$.

Recall that if $m = n$, we have

$$\mathbf{A} = \mathbf{QR}' \iff \begin{pmatrix} | & & | \\ \mathbf{x}_1 & \dots & \mathbf{x}_m \\ | & & | \end{pmatrix} = \begin{pmatrix} | & & | \\ \mathbf{q}_1 & \dots & \mathbf{q}_m \\ | & & | \end{pmatrix} \begin{pmatrix} \langle \mathbf{q}_1, \mathbf{x}_1 \rangle & \langle \mathbf{q}_1, \mathbf{x}_2 \rangle & \dots & \langle \mathbf{q}_1, \mathbf{x}_m \rangle \\ 0 & \langle \mathbf{q}_2, \mathbf{x}_2 \rangle & \dots & \langle \mathbf{q}_2, \mathbf{x}_m \rangle \\ 0 & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \langle \mathbf{q}_m, \mathbf{x}_m \rangle \end{pmatrix}.$$

Since now we have $m > n$, we first add $(m - n)$ columns in $\mathbf{A}$ such that the columns of $\mathbf{A}$ spans $\mathbb{R}^m$. Then, we carry out the Gram-Schmidt process as per normal to find $\mathbf{Q}$ and $\mathbf{R}'$. After that, we only take the first n columns of $\mathbf{R}'$, which gives us $\binom{\mathbf{R}}{\mathbf{0}}$. This is known as the full $\mathbf{QR}$ factorisation.

Now, the residual becomes

$$\begin{aligned} \|\mathbf{r}\|_2^2 = \left\| \mathbf{b} - \mathbf{Q} \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{x} \right\|_2^2 &= \left\| \mathbf{Q}^T \mathbf{b} - \begin{pmatrix} \mathbf{R} \\ \mathbf{0} \end{pmatrix} \mathbf{x} \right\|_2^2 \\ &= \|\mathbf{c}_1 - \mathbf{R}\mathbf{x}\|_2^2 + \|\mathbf{c}_2\|_2^2, \end{aligned}$$

where

$$\mathbf{Q}^T \mathbf{b} = \begin{pmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{pmatrix}.$$

5.15. **Householder transformations.**

So far, our intention is to perform orthogonal transformations, which are to transform vectors such that the resulting vectors are orthogonal to each other with the original lengths preserved. It can be done in multiple ways, such as (classical/modified) Gram-Schmidt process, rotations, and reflections. In practical, we use reflections the most (at least in this chapter of this module).

Definition 5.48. (Householder reflector) A Householder reflector is an orthogonal matrix that reflects all m-vectors through an $m-1$ dimensional plane. If we want to transform a vector $\mathbf{x}$ to a vector $\mathbf{w}$ of equal length, then we need to find a reflector $\mathbf{H}$ such that

$$\mathbf{Hx} = \mathbf{w}.$$

Lemma 5.49. *If $\boldsymbol{x}$ and $\boldsymbol{w}$ are vectors of the same Euclidean length, then $\boldsymbol{w}-\boldsymbol{x}$ and $\boldsymbol{w}+\boldsymbol{x}$ are perpendicular.*

Proof. We have $(\mathbf{w}-\mathbf{x})^T(\mathbf{w}-\mathbf{x}) = \mathbf{w}^T\mathbf{w} - \mathbf{x}^T\mathbf{w} + \mathbf{w}^T\mathbf{x} - \mathbf{x}^T\mathbf{x} = \|\mathbf{w}\|_2^2 - \|\mathbf{x}\|_2^2 = 0.$ □

Let $\mathbf{v}$ be a non-zero vector. Recalling that the projection map onto the range of $\mathbf{A}$ is

$$\mathbf{P_A} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T,$$

by taking $\mathbf{A} = (\mathbf{v})$, a single column vector, we have

$$\mathbf{P_v} = \mathbf{v}(\mathbf{v}^T\mathbf{v})^{-1}\mathbf{v}^T = \mathbf{v}(\|\mathbf{v}\|_2^2)^{-1}\mathbf{v}^T = \frac{\mathbf{v}\mathbf{v}^T}{\|\mathbf{v}\|_2^2} = \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}.$$

Furthermore, the projection map onto the orthogonal complement of Span($\mathbf{v}$) is

$$\mathbf{P_{v^\perp}} = \mathbf{I} - \mathbf{P_v} = \mathbf{I} - \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}.$$

If we set $\mathbf{v} = \mathbf{x} - \mathbf{w}$, then we notice that $\mathbf{x} - 2\mathbf{P}_{\mathbf{v}}\mathbf{x} = \mathbf{w}$. This is because

$$\begin{aligned}
\mathbf{x} - 2\mathbf{P}_{\mathbf{v}}\mathbf{x} &= \mathbf{w} - \mathbf{v} - \frac{2\mathbf{v}\mathbf{v}^T\mathbf{x}}{\mathbf{v}^T\mathbf{v}} \\
&= \mathbf{w} - \mathbf{v} - \frac{\mathbf{v}\mathbf{v}^T\mathbf{x}}{\mathbf{v}^T\mathbf{v}} - \frac{\mathbf{v}\mathbf{v}^T(\mathbf{w}-\mathbf{v})}{\mathbf{v}^T\mathbf{v}} \\
&= \mathbf{w} - \frac{\mathbf{v}\mathbf{v}^T(\mathbf{w}+\mathbf{x})}{\mathbf{v}^T\mathbf{v}} \\
&= \mathbf{w},
\end{aligned}$$

as $\mathbf{x} + \mathbf{w}$ is orthogonal to $\mathbf{x} - \mathbf{w}$ by our lemma 5.49.

This gives us a general idea on how to define $\mathbf{H}$, which is to set $\mathbf{H} = \mathbf{I} - 2\mathbf{P}_{\mathbf{v}}$. It is to be noted that by definition, we have

(1) $\mathbf{H} = \mathbf{H}^T$; and
(2) $\mathbf{H}\mathbf{H} = \mathbf{I}$.

It is to be noted that (2) is true because $\mathbf{P}_{\mathbf{v}}^2 = \mathbf{P}_{\mathbf{v}}$. Geometrically, we can understand it as the projection of (projection onto $\mathbf{v}$) onto $\mathbf{v}$ is just the projection onto $\mathbf{v}$.

The question is how we choose $\mathbf{v}$ in general.

Since we want to transform $\mathbf{A}$ to an upper triangular matrix, if $\mathbf{a}_1$ is the first column of $\mathbf{A}$, we will choose $\mathbf{w} = \pm \|\mathbf{a}_1\|_2 \mathbf{e}_1$ (either sign will work, but usually use the opposite sign of first entry of $\mathbf{a}_1$) and define $\mathbf{H}_1 = \mathbf{I} - 2\mathbf{P}_{\mathbf{v}_1}$ for $\mathbf{v}_1 = \mathbf{a}_1 - \mathbf{w}$. Clearly, $\|\mathbf{a}_1\|_2 = \|\mathbf{w}\|_2$. Then, we have

$$\mathbf{H}_1\mathbf{A} = \begin{pmatrix} \times & \times & \dots & \times \\ 0 & \times & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \times & \dots & \times \end{pmatrix},$$

where $\times$ is some arbitrary value which we don't care.

Now, we dissect the matrix as

$$\mathbf{H}_1\mathbf{A} = \left(\begin{array}{c|ccc} \times & \times & \dots & \times \\ \hline \mathbf{0} & & \mathbf{A}_2 & \end{array}\right),$$

where $\mathbf{A}_2$ is a $(m-1)\times(n-1)$ matrix. We now want to move the first column of $\mathbf{A}_2$, denoted by $\mathbf{a}_2'$, i.e., the lower $(m-1)$ entries of the second column of $\mathbf{A}$, to $\mathbf{w} = \|\mathbf{a}_2'\|_2\,\mathbf{e}_1$. This is done by defining a $(m-1)\times(m-1)$ Householder reflector $\hat{\mathbf{H}}_2$ with a similar process. We then further define that

$$\mathbf{H}_2 = \left(\begin{array}{c|c} 1 & \mathbf{0} \\ \hline \mathbf{0} & \hat{\mathbf{H}}_2 \end{array}\right).$$

This then gives us

$$\mathbf{H}_2\mathbf{H}_1\mathbf{A} = \begin{pmatrix} \times & \times & \dots & \times \\ 0 & \times & \dots & \times \\ 0 & 0 & \dots & \times \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \times \end{pmatrix}.$$

With a similar process, we can define a series of $\mathbf{H}_i$ such that

$$\left(\prod_{i=1}^{n-1} \mathbf{H}_{(n-i)}\right)\mathbf{A} = \mathbf{R}$$

for some upper triangular matrix $\mathbf{R}$. The corresponding $\mathbf{Q}$ is thus defined by

$$\mathbf{Q}^T = \prod_{i=1}^{n-1} \mathbf{H}_{(n-i)}.$$

It is to be noted that $\mathbf{Q}$ defined in this way may not be unique.

5.16. **Givens gotation.**

In $\mathbb{R}^2$, if we want to rotate a vector by θ anticlockwise, the rotation matrix is given by

$$\mathbf{R}_\theta = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}.$$

By staring at $\mathbf{R}_\theta$, we know that it is orthogonal.

Now, we want to determine θ which transforms (x_1, x_2) to $(x, 0)$. This amounts to solving

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x \\ 0 \end{pmatrix} \iff \begin{pmatrix} x_1 & x_2 \\ x_2 & -x_1 \end{pmatrix} \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} = \begin{pmatrix} x \\ 0 \end{pmatrix},$$

which gives us

$$\tan\theta = \frac{x_2}{x_1}.$$

By some seemingly complex trigonometric operations, we can derive the two expressions given in the lecture slides:

$$\cos\theta = \frac{x_1}{\sqrt{x_1^2 + x_2^2}}, \quad \sin\theta = \frac{x_2}{\sqrt{x_1^2 + x_2^2}}.$$

If we want to do rotations in higher-dimensional spaces, we can reduce the space to $x_a - x_b$ plane and do rotations on this plane for some $1 \le a < b \le n$. The corresponding rotation matrix is given by

$$\mathbf{R}_\theta = (r_{ij})_{n\times n} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i \neq a \text{ or } j \neq b); \\ 1 & \text{if } i = j \text{ and } i \neq a \text{ and } j \neq b; \\ \cos\theta & \text{if } i = j \text{ and } (i = a \text{ or } j = b); \\ \sin\theta & \text{if } i = a \text{ and } j = b; \\ -\sin\theta & \text{if } i = b \text{ and } j = a. \end{cases}$$

5.17. **Singular value decomposition.**

So far, our assumption on $\mathbf{A}$ is that we need $\mathbf{A}$ to have independent columns, i.e., $\mathbf{A}$ is of full rank. Yet, this may not hold in reality, especially when there are some kinds of rounding errors. In this case, we can perform singular value decomposition on $\mathbf{A}$. During the process, we can drop zero or small singular values, which allows us to handle rank deficiency or near rank deficiency problems.

For instance, consider a 4×3 matrix $\mathbf{A}$ with SVD

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \begin{pmatrix} | & | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 \\ | & | & | & | \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \\ - & \mathbf{v}_3^T & - \end{pmatrix}.$$

In this case, the matrix $\mathbf{A}$ we are dealing with is not of full rank. However, eventually, we have

$$\mathbf{A} = \sigma \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T.$$

This allows us to form

$$\mathbf{A} = \mathbf{U}'\mathbf{\Sigma}'(\mathbf{V}')^T = \begin{pmatrix} | & | \\ \mathbf{u}_1 & \mathbf{u}_2 \\ | & | \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} - & \mathbf{v}_1^T & - \\ - & \mathbf{v}_2^T & - \end{pmatrix}.$$

In this case, we still have $(\mathbf{U}')^T\mathbf{U}' = \mathbf{I}$ and $(\mathbf{V}')^T\mathbf{V}' = \mathbf{I}$. Also, we can still define

$$\mathbf{A}^\dagger = \mathbf{V}'(\mathbf{\Sigma}')^{-1}(\mathbf{U}')^T,$$

which allows us to minimise $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$ even if $\mathbf{A}$ is not full rank.

If we want to analyse the whole minimisation process more carefully, then we have

$$\begin{aligned}
\|\mathbf{Ax}-\mathbf{b}\|_2 &= \left\|\mathbf{U\Sigma V}^T\mathbf{x}-\mathbf{b}\right\|_2 \\
&= \left\|\mathbf{\Sigma V}^T\mathbf{x}-\mathbf{U}^T\mathbf{b}\right\|_2 && (\mathbf{U} \text{ is orthogonal}) \\
&= \|\mathbf{w}-\mathbf{v}\|_2 && (\mathbf{w}=\mathbf{\Sigma V}^T\mathbf{x} \text{ and } \mathbf{v}=\mathbf{U}^T\mathbf{b}) \\
&= \sum_{i=1}^{r}(w_i-v_i)^2+\sum_{i=r+1}^{m}(w_i-v_i)^2 && (\mathbf{w}-\mathbf{v} \text{ is an } m\text{-vector}, r=\operatorname{rank}(\mathbf{A})) \\
&= \sum_{i=1}^{r}(w_i-v_i)^2+\sum_{i=r+1}^{m}v_i^2. && (\mathbf{\Sigma} \text{ has at most } \operatorname{rank}(\mathbf{A}) \text{ non-zero rows})
\end{aligned}$$

Therefore, if we have $w_i = v_i$ for all $1 \le i \le r$, then we have minimised the $\|\mathbf{Ax}-\mathbf{b}\|_2$.

By looking at the product carefully, we have

$$w_i = (\mathbf{\Sigma V}^T\mathbf{x})_i = \sigma_i\mathbf{v}_i^T\mathbf{x}, \quad v_i = \mathbf{u}_i^T\mathbf{b},$$

which then translates the question to finding an $\mathbf{x}$ such that $\sigma_i\mathbf{v}_i^T\mathbf{x} = \mathbf{u}_i^T\mathbf{b}$ for all $1 \le i \le r$.

The proposed $\mathbf{x}$ is

$$\mathbf{x} = \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i,$$

which gives us

$$w_i = \sigma_i\mathbf{v}_i^T\mathbf{x} = \mathbf{v}_i^T(\mathbf{u}_i^T\mathbf{b})\mathbf{v}_i = v_j(\mathbf{v}_i^T\mathbf{v}_i) = v_j,$$

as $\mathbf{V}$ is orthogonal.

SVD also has a lot of applications, some of which are

- minimum norm solution;
- condition number;
- rank determination;
- pseudo-inverse;
- low-rank approximation.

6. Eigenvalue problems

6.1. **Eigenvalues and eigenvectors.**

Definition 6.1. (Eigenspace) Let $\mathbf{A}$ be an $n \times n$ square matrix. We define

$$E_\lambda^{\mathbf{A}} := \{\mathbf{v} \in \mathbb{R}^n : \mathbf{A}\mathbf{v} = \lambda\mathbf{v}\}$$

as the eigenspace of $\mathbf{A}$ for all $\lambda \in \mathbb{R}$.

Remark 6.2. Based on the above definition, it should be clear that $\mathbf{0} \in E_\lambda$ for all $\lambda \in \mathbb{R}$. Hence, all eigenspaces of any square matrix is non-empty.

Definition 6.3. (Eigenvalue) If $E_\lambda \neq \{\mathbf{0}\}$, then we say that λ is an eigenvalue of $\mathbf{A}$.

Definition 6.4. (Eigenvector) If $E_\lambda \neq \{\mathbf{0}\}$, then we say that every vector in $E_\lambda \backslash \{\mathbf{0}\}$ is an eigenvector of $\mathbf{A}$ corresponding to the eigenvalue $\lambda \in \mathbb{R}$.

Definition 6.5. (Characteristic polynomial) Let $\mathbf{A}$ be a square matrix. The characteristic polynomial $p_{\mathbf{A}}(x)$ of a matrix is defined by $p_{\mathbf{A}}(x) = \det(\mathbf{A} - x\mathbf{I})$.

Proposition 6.6. *The following statements are equivalent:*

(1) λ *is an eigenvalue of* $\boldsymbol{A}$*;*
(2) $E_\lambda \backslash \{\boldsymbol{0}\} \neq \emptyset$*;*
(3) *there exists non-zero vector* $\boldsymbol{v}$ *such that* $\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$*;*
(4) $(\boldsymbol{A} - \lambda\boldsymbol{I})\boldsymbol{x} = \boldsymbol{0}$ *has non-trivial solution;*
(5) $(\boldsymbol{A} - \lambda\boldsymbol{I})$ *is invertible;*
(6) λ *is a root to the characteristic polynomial of* $\boldsymbol{A}$*, i.e.,* $p_{\boldsymbol{A}}(\lambda) = 0$.

Proof. They are standard results from MA2001. □

Definition 6.7. (Left eigenvector) A left-eigenvector of a square matrix $\mathbf{A}$ is a non-zero vector $\mathbf{y}$ satisfying $\mathbf{y}^T\mathbf{A} = \lambda\mathbf{y}^T$, or equivalently, $\mathbf{A}^T\mathbf{y} = \lambda\mathbf{y}$.

Definition 6.8. (Algebraic multiplicity) Suppose $p_{\mathbf{A}}(x) = \prod_{i=1}^{n}(x - \lambda_i)^{k_i}$ for some $\lambda_i \in \mathbb{R}$, $k_i \in \mathbb{Z}^+$, $1 \leq i \leq n$. By proposition 6.6, the eigenvalues of $\mathbf{A}$ are just λ_i. We define the algebraic multiplicity of λ_i as the value of k_i, i.e., $\mathrm{AM}(\lambda_i) = k_i$.

Definition 6.9. (Geometric multiplicity) The geometric multiplicity of an eigenvalue of $\mathbf{A}$ is defined as the dimension of its eigenspace, i.e., $\mathrm{GM}(\lambda_i) = \dim(E_{\lambda_i})$.

Lemma 6.10. $\mathrm{GM}(\lambda_i) \leq \mathrm{AM}(\lambda_i)$.

Remark 6.11. The author only knows how to prove the above lemma by using Cayley-Hamilton theorem, which requires some effort to establish. Therefore, we just assume the above lemma as what is given.

Definition 6.12. (Defective) Let $\mathbf{A}$ be a square matrix. If $\mathrm{GM}(\lambda_i) < \mathrm{AM}(\lambda_i)$ for some eigenvalue λ_i of $\mathbf{A}$, then we say that $\mathbf{A}$ is defective.

Proposition 6.13. *Non-defective matrices are diagonalisable. This means, if for all eigenvalues of $\boldsymbol{A}$, $AM(\lambda_i) = GM(\lambda_i)$, then $\boldsymbol{A}$ is diagonalisable.*

Proposition 6.14. *Let $\boldsymbol{A}$ be a square matrix and λ be one of its eigenvalue. Then,*

(1) $\boldsymbol{Ax} = \lambda\boldsymbol{x} \iff (\boldsymbol{A} - \mu I)\boldsymbol{x} = (\lambda - \mu)\boldsymbol{x}$;
(2) $\boldsymbol{Ax} = \lambda\boldsymbol{x} \iff \boldsymbol{A}^{-1}\boldsymbol{x} = \lambda^{-1}\boldsymbol{x}$;
(3) $\boldsymbol{Ax} = \lambda\boldsymbol{x} \iff \boldsymbol{A}^{k}\boldsymbol{x} = \lambda^{k}\boldsymbol{x}$;
(4) $\boldsymbol{B}$ *is similar to* $\boldsymbol{A} \implies \lambda$ *is an eigenvalue of* $\boldsymbol{B}$.

6.2. **Power iteration.**

Throughout this section, let $\mathbf{A}$ be a $n \times n$ square matrix.

Definition 6.15. (Dominant eigenvalue) A dominant eigenvalue of $\mathbf{A}$ is an eigenvalue λ whose magnitude is greater than all other eigenvalues of $\mathbf{A}$. If it exists, an eigenvector associated to λ is called a dominant eigenvector.

Remark 6.16. Here, the magnitude of an eigenvalue refers to its absolute value.

Since if λ is an eigenvalue of $\mathbf{A}$, there exists $\mathbf{v}$ such that $\mathbf{Av} = \lambda\mathbf{v}$, we then have

$$\mathbf{v}^T\mathbf{Av} = \lambda\mathbf{v}^T\mathbf{v},$$

which implies

$$\lambda = \frac{\mathbf{v}^T\mathbf{A}\mathbf{v}}{\mathbf{v}^T\mathbf{v}}.$$

Definition 6.17. (Rayleigh quotient) If $\mathbf{v}$ is an approximated eigenvector, its approximated corresponding eigenvalue can be determined by

$$\lambda = \frac{\mathbf{v}^T\mathbf{A}\mathbf{v}}{\mathbf{v}^T\mathbf{v}},$$

which is called the Rayleigh quotient.

Definition 6.18. (Power iteration) We arbitrarily choose $\mathbf{v}_0$ as the initial vector. The power iteration can be performed in the following way:

(1) normalise the vector, i.e., define $\mathbf{u}_i := \dfrac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$;
(2) calculate a new vector $\mathbf{x}_i$ by $\mathbf{x}_{i+1} = \mathbf{A}\mathbf{u}_i$;
(3) use Rayleigh quotient to approximate the eigenvalue of $\mathbf{x}_i$.

When $\|\mathbf{v}_{i+1} - \mathbf{v}_i\|$ is small enough, we stop the power iteration and get an eigenvector $\mathbf{u} = \dfrac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$ for the stopping $\mathbf{v}_i$.

Remark 6.19. Since $\mathbf{A}$ is diagonalisable, we can choose eigenvectors of $\mathbf{A}$ such that they form a basis of $\mathbb{R}^n$. Consequently, for any $\mathbf{v}_0$, we can write

$$\mathbf{v}_0 = \sum_{i=1}^{n} c_i\mathbf{b}_i,$$

where $c_i \in \mathbb{R}$ and $\mathbf{b}_i$ are eigenvectors of $\mathbf{A}$. By performing power iteration, we have

$$\mathbf{v}_1 = \mathbf{A}\mathbf{v}_0 = \mathbf{A}\left(\sum_{i=1}^{n} c_i\mathbf{b}_i\right) = \sum_{i=1}^{n} c_i\mathbf{A}(\mathbf{b}_i) = \sum_{i=1}^{n} c_i\lambda_i\mathbf{b}_i$$

$$\mathbf{v}_2 = \mathbf{A}\mathbf{v}_1 = \mathbf{A}\left(\sum_{i=1}^{n} c_i\lambda_i\mathbf{b}_i\right) = \sum_{i=1}^{n} c_i\lambda_i\mathbf{A}(\mathbf{b}_i) = \sum_{i=1}^{n} c_i\lambda_i^2\mathbf{b}_i$$

$$\vdots$$

$$\mathbf{v}_n = \mathbf{A}\mathbf{v}_{n-1} = \mathbf{A}\left(\sum_{i=1}^{n} c_i\lambda_i^{n-1}\mathbf{b}_i\right) = \sum_{i=1}^{n} c_i\lambda_i^{n-1}\mathbf{A}(\mathbf{b}_i) = \sum_{i=1}^{n} c_i\lambda_i^{n}\mathbf{b}_i.$$

Since $\mathbf{A}$ is diagonalisable, we can arrange its eigenvalues such that

$$|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

Therefore, if we have a dominant eigenvalue λ_1, as compared to λ_1^n, all other λ_i^n will be negligible, so $\mathbf{A}\mathbf{v}_n$ will be dominated by the term $\lambda_1^n \mathbf{b}_1$. Hence, the eigenvector power iteration converges to will be $\mathbf{b}_1$, or an eigenvector corresponding to the dominant eigenvalue.

Remark 6.20. From the above reasoning, if $\lambda_1 > 1$, as the number of iteration increases, λ_1^n will be very large. Similarly, if $\lambda < 1$, λ_1^n will be very small. Both of these situations will affect the accuracy of the computed eigenvector. To avoid these issues, we need to normalise the eigenvector before each iteration.

Theorem 6.21. *For almost every initial vector, power iteration converges linearly to an eigenvector associated to λ_i with convergence rate constant $S = \left|\dfrac{\lambda_2}{\lambda_1}\right|$.*

Remark 6.22. For the term 'for almost every initial vector', it means that the theorem is generally true except for some exceptions. In this case, if x_0 is contained in the union of $\{\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_n\}$ and $\{\mathbf{v}_1, \mathbf{v}_3, \ldots, \mathbf{v}_n\}$, where $\mathbf{v}_i$ is an eigenvector corresponding to eigenvalue λ_i, then the power iteration may not converge to an eigenvector associated to λ_i or converge at this rate.

6.3. **Inverse power iteration.**

Lemma 6.23. *If λ is an eigenvalue of $\boldsymbol{A}$, then $\dfrac{1}{\lambda}$ is an eigenvalue of $\boldsymbol{A}^{-1}$.*

Proposition 6.24. *Applying power iteration to $\boldsymbol{A}^{-1}$ will lead to an eigenvector corresponding to the eigenvalue smallest in magnitude of $\boldsymbol{A}$ (as its reciprocal is the largest).*

Lemma 6.25. *If λ is an eigenvalue of $\boldsymbol{A}$, then $\lambda - \delta$ is an eigenvalue of $\boldsymbol{A} - \delta\boldsymbol{I}$.*

Proposition 6.26. *Applying inverse power iteration to $\boldsymbol{A} - \delta\boldsymbol{I}$ will lead to an eigenvector corresponding to the eigenvalue closest to δ. The closer δ is to λ, the faster the convergence.*

6.4. **Rayleigh quotient iteration.**

Consider the linear system $\mathbf{x}(\lambda) = \mathbf{A}\mathbf{x}$. It can be seen as an over-determined system as $\mathbf{x}$ is an $n \times 1$ matrix. By the least square method, we have

$$\mathbf{x}(\lambda) = \mathbf{A}\mathbf{x} \implies \mathbf{x}^T\mathbf{x}\lambda = \mathbf{x}^T\mathbf{A}\mathbf{x} \implies \lambda = \frac{\mathbf{x}^T\mathbf{A}\mathbf{x}}{\mathbf{x}^T\mathbf{x}},$$

which is another way to derive the Rayleigh quotient. With this quotient, recalling that $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T\mathbf{y}$, we have

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \implies \frac{\langle \mathbf{A}\mathbf{x}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} = \frac{\langle \lambda\mathbf{x}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} = \frac{\lambda\langle \mathbf{x}, \mathbf{x} \rangle}{\langle \mathbf{x}, \mathbf{x} \rangle} = \lambda.$$

Combining this with inverse power iteration, we have the Rayleigh quotient iteration.

Definition 6.27. (Rayleigh quotient iteration) We arbitrarily choose $\mathbf{v}_0$ as the initial vector. The Rayleigh quotient iteration can be performed in the following way:

(1) normalise the vector, i.e., define $\mathbf{u}_i := \dfrac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$;
(2) calculate the Rayleigh quotient of $\mathbf{u}_i$, i.e. $\sigma_i = \dfrac{\mathbf{u}_i^T\mathbf{A}\mathbf{u}_i}{\mathbf{u}_i^T\mathbf{u}_i}$;
(3) calculate $\mathbf{v}_{i+1}$ by solving $(\mathbf{A} - \sigma_i\mathbf{I})\mathbf{v}_{i+1} = \mathbf{u}_i$.

When $\|\mathbf{v}_{i+1} - \mathbf{v}_i\|$ is small enough, we stop the power iteration and get an eigenvector $\mathbf{u} = \dfrac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2}$ for the stopping $\mathbf{v}_i$.

6.5. **Simultaneous iteration.**

For a general matrix $\mathbf{A}$, we wonder if there is a way to find a set of n eigenvectors of $\mathbf{A}$, which leads to simultaneous iteration.

When using this method, we cannot directly use power iteration or inverse power iteration since if so, the eigenvectors we get will all be corresponding to the largest/smallest eigenvalue. We thus start to consider the $\mathbf{QR}$ factorisation. Writing it in symbol, there exists an orthogonal matrix $\overline{\mathbf{Q}}_1$ and an upper triangular matrix $\mathbf{R}_1$ such that $\mathbf{A}\mathbf{I} = \overline{\mathbf{Q}}_1\mathbf{R}_1$, where $\overline{\mathbf{Q}}_1 = (\mathbf{q}_1, \ldots, \mathbf{q}_n)$.

On the one hand, this is an $\mathbf{QR}$ factorisation. On the other hand, we can consider $\mathbf{A}\mathbf{I}$ as applying power iteration once to the identity matrix. To explain this idea further, if we consider the identity matrix as a $1 \times n$ matrix, i.e., $\mathbf{I} = (\mathbf{e}_i)$, then each column of $\mathbf{A}$ is given by $\mathbf{A}\mathbf{e}_i$,

which is to apply power iteration once to the standard basis vector. Therefore, each column of $\overline{\mathbf{Q}}_1$ will be an approximation of an eigenvector, and the corresponding diagonal entry of $\mathbf{R}_1$ will be the approximated eigenvalues.

Definition 6.28. (Normalised simultaneous iteration) Set $\overline{\mathbf{Q}}_0 = \mathbf{I}$. The normalised simultaneous iteration is performed in the following manner:

(1) $\mathbf{A}\overline{\mathbf{Q}}_j = \overline{\mathbf{Q}}_{j+1}\mathbf{R}_{j+1}$.

6.6. QR iteration.

If we want to write the normalised simultaneous iteration in a more compact manner, we can use **QR** iteration.

Definition 6.29. (Unshifted **QR** iteration) Set $\mathbf{A}_0 = \mathbf{A}$ and $\overline{\mathbf{Q}}_0 = \mathbf{Q}_0 = \mathbf{I}$. The **QR** iteration is performed in the following manner.

(1) $\mathbf{A}_{i+1} = \mathbf{Q}_{i+1}\mathbf{R}_{i+1} = \mathbf{R}_i\mathbf{Q}_i$;
(2) $\overline{\mathbf{Q}}_{i+1} = \overline{\mathbf{Q}}_i\mathbf{Q}_{i+1}$.

We will show the calculation of the first few iterations for readers to make sense of this algorithm. Since $\mathbf{Q}$ is orthogonal, we have $\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$ and thus

$$\begin{aligned}
&\mathbf{A}_1 = \mathbf{A} = \mathbf{Q}_1\mathbf{R}_1,\\
&\mathbf{A}_1 = \mathbf{R}_1\mathbf{Q}_1 \implies \mathbf{Q}_2\mathbf{R}_2 = \mathbf{A}_1 = \mathbf{Q}_1^T\mathbf{Q}_1\mathbf{R}_1\mathbf{Q}_1 = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1,\\
&\mathbf{A}_2 = \mathbf{R}_2\mathbf{Q}_2 \implies \mathbf{Q}_3\mathbf{R}_3 = \mathbf{A}_2 = \mathbf{Q}_2^T\mathbf{Q}_2\mathbf{R}_2\mathbf{Q}_2 = \mathbf{Q}_2^T\mathbf{A}_2\mathbf{Q}_2 = \mathbf{Q}_2^T\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2.
\end{aligned}$$

Comparing this with the normalised simultaneous iteration:

$$\begin{aligned}
\mathbf{A} = \mathbf{A}\overline{\mathbf{Q}}_0 = \overline{\mathbf{Q}}_1\mathbf{R}_1' &\implies \overline{\mathbf{Q}}_1 = \mathbf{Q}_1 \text{ and } \mathbf{R}_1' = \mathbf{R}_2,\\
\overline{\mathbf{Q}}_2\mathbf{R}_2' = \mathbf{A}\overline{\mathbf{Q}}_1 \implies \overline{\mathbf{Q}}_2\mathbf{R}_2' = \overline{\mathbf{Q}}_1\mathbf{R}_1'\overline{\mathbf{Q}}_1 = \mathbf{Q}_1\mathbf{R}_1'\mathbf{Q}_1 = \mathbf{Q}_1\mathbf{Q}_2\mathbf{R}_2' &\implies \overline{\mathbf{Q}}_1 = \mathbf{Q}_1\mathbf{Q}_2 \text{ and } \mathbf{R}_2' = \mathbf{R}_2,\\
&\vdots\\
\overline{\mathbf{Q}}_n = \mathbf{Q}_1 \dots \mathbf{Q}_n &\text{ and } \mathbf{R}_n' = \mathbf{R}_n.
\end{aligned}$$

Proposition 6.30. *If $\boldsymbol{A}$ is real symmetric, then $\boldsymbol{A}_k$ converges to a diagonal matrix. Otherwise, $\boldsymbol{A}_k$ converges to an upper-triangular or block upper-triangular matrix.*

By shifting and deflating, we can improve the convergence of **QR** iteration.

Definition 6.31. (Real Schur form) A matrix **T** has real Schur form if it is upper triangular, except possibly for 2×2 blocks on the main diagonal.

Remark 6.32. If one is familiar with Jordan canonical form, one may find some common behaviour between the Schur form and the Jordan form.

Theorem 6.33. *Let $\boldsymbol{A}$ be a square matrix with real entries. Then there exists an orthogonal matrix $\boldsymbol{Q}$ and a matrix $\boldsymbol{T}$ in real Schur form such that $\boldsymbol{A} = \boldsymbol{Q}^T \boldsymbol{T} \boldsymbol{Q}$.*

Theorem 6.34. *For a real Schur matrix, its eigenvalues are eigenvalues of the diagonal blocks.*

Example 6.35. The matrix below is clearly a matrix in real Schur form:

$$\left(\begin{array}{c|cc|c} 1 & 1 & 1 & 1 \\ \hline 0 & 7 & 1 & 0 \\ 0 & 1 & 7 & 0 \\ \hline 0 & 0 & 0 & 3 \end{array}\right).$$

In this case, its eigenvalues are just $1, 3$ (the two diagonal entries) together with the eigenvalues of $\begin{pmatrix} 7 & 1 \\ 1 & 7 \end{pmatrix}$.

Remark 6.36. With this regard, the real Schur matrix reveals the eigenvalues of **A**, and the corresponding eigenvectors are revealed in **Q**.

The full **QR** algorithm iteratively moves an arbitrary matrix **A** toward its Schur factorisation by a series of similarity transformations.

The details of shifted QR algorithm is omitted since

6.7. **Upper Hessenberg form.**

Efficiency of the **QR** algorithm increases considerably if we first put **A** into upper Hessenberg form.

Definition 6.37. The $m \times n$ matrix **A** is in upper Hessenberg form if $a_{ij} = 0$ for $i > j + 1$.

Theorem 6.38. *Let* $\boldsymbol{A}$ *be a square matrix. There exists an orthogonal matrix* $\boldsymbol{Q}$ *such that* $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{B}\boldsymbol{Q}^T$ *and* $\boldsymbol{B}$ *is in upper Hessenberg form.*

Remark 6.39. To calculate the upper Hessenberg form, we can use the Householder reflectors. The details can be found in the section 12.2.3 of the reference book, which are lengthy but can be understood easily.

Remark 6.40. This is the author's first revision, so some details will be filled in the future drafts.

7. Non-linear equations

7.1. **Conditioning.**

For evaluating a function, the condition number is approximately

$$\text{condition number} \approx \left| \frac{xf'(x)}{f(x)} \right|.$$

The detailed derivation can be found in the first chapter of this notes.

With a similar argument, for root finding, we will use $\frac{1}{|f'(x)|}$. In higher dimensions, we will use Jacobian.

7.2. **Convergence and stopping criteria.**

Definition 7.1. (Convergence rate) We say that an algorithm has a convergence rate r when

$$\lim_{k\to\infty} \frac{\|\mathbf{e}_{k+1}\|}{\|\mathbf{e}_k\|^r} = C > 0.$$

When

- $r = 1$ and $C < 1$, the algorithm converges linearly;
- $r > 1$, the algorithm converges superlinearly;
- $r = 2$, the algorithm converges quadratically;
- $r = 3$, the algorithm converges cubically.

When deciding when the algorithm should stop, we usually consider the relative change in iterates, i.e., $\frac{\|x_{k+1} - x_k\|}{\|x_k\|}$.

7.3. **Bisection.**

Theorem 7.2. (Intermediate value theorem) *If f is continuous on $[a, b]$, then there exists a $c \in \mathbb{R}$ such that $f(c) = m$ for all $m \in [\min(f(a), f(b)), \max(f(a), f(b))]$.*

Therefore, if we have a continuous nonlinear equation, we can use bisection to determine its root.

Definition 7.3. (Bisection) The bisection method is performed in the following manner:

(1) Start with an initial interval $[a, b]$ such that $f(a)$ and $f(b)$ have different signs. Without loss of generality, we assume $f(a) < 0 < f(b)$;
(2) calculate the value of $f\left(\frac{a+b}{2}\right)$;
(3) if $f\left(\frac{a+b}{2}\right) > 0$, update the interval to $\left[a, \frac{a+b}{2}\right]$; $f\left(\frac{a+b}{2}\right) < 0$, then update the interval to $\left[\frac{a+b}{2}, b\right]$;
(4) continue steps (2) and (3) until we get a root of the desired precision.

7.4. **Fixed-point iteration.**

Suppose we are solving $f(x) = 0$ for some non-linear equation $f(x)$. We can rewrite it in the form of $g(x) = x$ for some $g(x)$ related to $f(x)$. It is to be noted that the choice of $g(x)$ is not unique.

For instance, we want to solve for $x^2 + \cos x = 0$. In this case, if we define $f(x) = x^2 + \cos x$, then some of the possible choices of $g(x)$ such that $f(x) = 0 \implies g(x) = x$ are:

$$g_1(x) = x^2 + \cos x + x,$$
$$g_2(x) = x^3 + x\cos x + x.$$

We now perform the iteration as $x_{n+1} = g(x_n)$. If the algorithm converges, we will have $x_{n+1} = x_n$, which is equivalent to $x = g(x)$. The question is, when the algorithm converges?

Since $x_{k+1} = g(x_k)$, the error term is given by

$$e_{k+1} = x_{k+1} - x = g(x_k) - g(x).$$

for all $k \in \mathbb{Z}^+$. By the Mean Value theorem, there exists c_k between x_k and x such that

$$g'(c_k)e_k = g'(c_k)(x_k - x) = g(x_k) - g(x) = e_{k+1}.$$

If we expect the algorithm to converge, then x_k is expected to be very close to x, so $g'(c_k)$ can be approximately taken to be equal to $g'(x)$, as c_k is between x_k and x. Here, x is an arbitrarily chosen point throughout the argument.

Hence, the convergence condition for the fixed point iteration depends on $g'(x)$. If $|g(x)| \geq 1$, then the iteration will not converge. If $0 < |g'(x)| < 1$, then the iteration will converge. If $g'(x) = 0$, we will need to consider higher derivatives as

$$g(x_k) - g(x) = g''(c_k')\frac{(x_k - x)^2}{2}$$

for some c_k' between x_k and x.

7.5. **Newton's method.**

By Taylor's theorem, we have $f(x_n) = f(x + h) \approx f(x) + f'(x)h$, where $x_n = x + h$.

If we take $f(x) + f'(x)h = 0$, then we have

$$h = -\frac{f(x)}{f'(x)}.$$

Hence, we can take an initial guess x_0, calculate h, compute $x_1 = x_0 + h$, and repeat this process.

Since Newton's method is still a type of fixed-point method, its convergence rate depends on $|g'(x)|$. Here, $g(x)$ is defined by

$$g(x) = x + h = x - \frac{f(x)}{f'(x)},$$

which gives us

$$g'(x) = \frac{f(x)f''(x)}{(f'(x))^2}.$$

If a is a simple root, then $f(a) = 0$ and $f'(a) \neq 0$, which implies that $g'(a) = 0$. By Taylor's theorem, this implies that local convergence to simple roots should be quadratic.

For multiple roots, the convergence is linear with constant $C = 1 - \frac{1}{m}$.

7.6. **Secant method.**

Computing derivatives can be inconvenient or computationally expensive, so we use

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

instead.

7.7. **Singular value decomposition.**

One may refer to section 5.10 for more information of SVD. Here, we only explain a method to compute SVD.

Suppose $\mathbf{A}$ is an $m \times n$ matrix. The SVD of $\mathbf{A}$ can be computed in the following manner:

(1) Compute $\mathbf{A}^T\mathbf{A}$, a $n \times n$ matrix, or $\mathbf{A}\mathbf{A}^T$, a $m \times m$ matrix, whichever is easier to be calculated. Without loss of generality, we assume that we computed $\mathbf{A}^T\mathbf{A}$.

(2) Find a set of orthonormal eigenvectors of $\mathbf{A}^T\mathbf{A}$. The existence of this orthonormal set of eigenvectors is guaranteed by the fact that $\mathbf{A}^T\mathbf{A}$ is real symmetric. We define

$$\mathbf{V} := (\mathbf{v}_1 \ \dots \ \mathbf{v}_n)$$

to be the matrix whose columns are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ we found. Similarly, we define $\mathbf{U} := (\mathbf{u}_1 \ \dots \ \mathbf{u}_n)$ to be the matrix whose columns form an orthonormal set and are all the eigenvectors of $\mathbf{A}\mathbf{A}^T$.

(3) Compute the singular values of $\mathbf{A}^T\mathbf{A}$. Define a matrix $\mathbf{\Sigma}$ whose diagonal entries are singular values of $\mathbf{A}^T\mathbf{A}$.

(4) Use the relation $\mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$ or $\mathbf{A}^T\mathbf{U} = \mathbf{V}\mathbf{\Sigma}$ to compute the SVD.

8. Interpolation

8.1. **General problem.**

The general problem of interpolation is, for a set of data points (t_i, y_i) for $1 \leq i \leq m$ and a collection of basis functions $\{f_1, \ldots, f_n\}$, we want to find an interpolating function

$$f(t) = x_1 f_1(t) + \ldots + x_n f_n(t)$$

such that $f(t_i) = y_i$. Equivalently, this is a system of linear equation given by

$$(*) \qquad \begin{pmatrix} f_1(t_1) & f_2(t_1) & \ldots & f_n(t_1) \\ f_1(t_2) & f_2(t_2) & \ldots & f_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(t_m) & f_2(t_m) & \ldots & f_n(t_m) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} f(t_1) \\ f(t_2) \\ \vdots \\ f(t_m) \end{pmatrix}.$$

Here, we view the set of polynomials as a vector space, and basis function refers to functions in the basis of this vector space.

With this understanding, a natural question to ask is how we can solve $(*)$ quickly to find the desired interpolating function. Since the coefficient matrix in $(*)$ explicitly depends on our choice of basis, we will explore some common sets of basis functions to see their advantages and disadvantages.

8.2. **Monomial basis.**

The monomial basis is the easiest one to think about, defined as $\{1, t, t^2, \ldots, t^n\}$. 'Mono' means one, and monomial simply means functions in the form of t^k for some $k \in \mathbb{Z}_0^+$. If our choice of basis is monomial basis, then the coefficient matrix in $(*)$ will be in the form of an

Vandermonde matrix, i.e.,

$$\begin{pmatrix} 1 & t_1 & \dots & t_1^n \\ 1 & t_2 & \dots & t_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_m & \dots & t_m^n \end{pmatrix}.$$

8.3. **Lagrange basis.**

Given a set of data points (t_i, y_i), $1 \leq i \leq n$, the Lagrange basis functions are given by

$$\mathcal{L}_j(t) = \frac{\prod_{k \neq j}(t - t_k)}{\prod_{k \neq j}(t_j - t_k)}.$$

It seems to be a very strange construction, but it is actually very natural. One may sub into the given set of data points to see how these basis functions work.

8.4. **Newton basis.**

Given a set of data points (t_i, y_i) for $1 \leq i \leq n$, the Newton basis functions are

$$\pi_j(t) = \prod_{k=1}^{j-1}(t - t_k).$$

Again, there is some similarities between Newton basis and Lagrange basis. Yet, as compared to the Lagrange basis, we can write the coefficient matrix of Newton basis out.

Example 8.1. Suppose we have $(t_1, y_1), (t_2, y_2), (t_3, y_3)$ as our given data points. Suppose the desired interpolation function is $p(t) = x_0 + x_1 t + x_2 t^2$. The coefficient matrix is given by

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & t_2 - t_1 & 0 \\ 1 & t_3 - t_1 & (t_3 - t_1)(t_3 - t_2) \end{pmatrix}.$$

As compared to the monomial basis, this coefficient matrix is lower-triangular, which is easier to be computed.

There is an even faster way of computing Newton interpolating function, which is known as the method of divided differences.

We define $f[t_i] = y_i$ as our base cases for $1 \leq i \leq n$. Then, we define

$$f[t_a, t_{a+1}, \ldots, t_b] = \frac{f[t_{a+1}, \ldots, t_b] - f[t_a, t_{a+1}, \ldots, t_{b-1}]}{t_b - t_a}$$

inductively for all $a, b \in \mathbb{Z}_0^+$, $a < b$. The Newton interpolation function is given by

$$p(t) = f[t_1] + f[t_1, t_2]\pi_2(t) + \ldots + + f[t_1, t_2, \ldots, t_n]\pi_n(t) = \sum_{i=1}^{n} f[t_1, \ldots, t_i]\pi_i(t).$$

8.5. **Orthogonal polynomials.**

Definition 8.2. (Inner product) Given a vector space V over a field F, an inner product is a function $\langle \cdot, \cdot \rangle : V \times V \mapsto F$ satisfying

(1) $\langle v, v \rangle \geq 0$ for all $v \in V$;
(2) $\langle v, v \rangle = 0 \iff v = 0$;
(3) $\langle u + \lambda v, w \rangle = \langle u, w \rangle + \lambda \langle v, w \rangle$ for all $u, v, w \in V$, $\lambda \in F$;
(4) $\langle u, v \rangle = \overline{\langle v, u \rangle}$ for all $u, v \in V$.

Example 8.3. The dot product on $\mathbb{R}^n$ defined by

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$$

is an inner product.

Definition 8.4. (Orthonormal set) An orthonormal set is a non-empty set such that for all vectors x, y contained in the set, we have

$$\langle x, x \rangle = 1 \text{ and } \langle x, y \rangle = 0$$

for all $x \neq y$.

Remark 8.5. We can imagine that an orthonormal basis of a vector space is a set of axes established in this vector space, all of which are mutually orthogonal to each other.

Proposition 8.6. *Let $\mathcal{B} := \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n\}$ be an orthonormal basis of V. Then for all $\boldsymbol{v} \in V$, we can decompose $\boldsymbol{v}$ as*

$$\boldsymbol{v} = \langle \boldsymbol{v}, \boldsymbol{b}_1 \rangle \boldsymbol{b}_1 + \ldots + \langle \boldsymbol{v}, \boldsymbol{b}_n \rangle \boldsymbol{b}_n.$$

Furthermore, by the properties of basis, this decomposition is unique up to the reordering of summands. This means, this expression is unique if we do not exchange the positions of summands.

Example 8.7. (Inner product on polynomial space) Let $\mathbb{R}[x]$ be the set of polynomials with real coefficients. We can define an inner product on it by

$$(\forall f, g \in \mathbb{R}[x]) \left(\langle f, g \rangle = \int_a^b f(x)g(x)\ dx \right)$$

for some arbitrarily chosen $a, b \in \mathbb{R}$, depending on the context of the question. In this set of notes, we just choose $a = -1$ and $b = 1$ to have a sense of symmetry, i.e.,

$$\langle f, g \rangle = \int_{-1}^1 f(x)g(x)\ dx.$$

Since $\mathbb{R}[x]$ is now a vector space with a natural monomial basis $\{1, x, x^2, \ldots\}$ (this is an infinite-dimensional vector space) and we have defined the notion of distance (inner product) on it, we can try to find an orthonormal basis by the Gram-Schmidt process for polynomials. This will give rise to orthonormal bases of $\mathbb{R}[x]$.

Definition 8.8. (Legendre polynomials) We define

$$\mathbb{R}_n[x] := \{\text{polynomials with real coefficients of degree at most } n\}$$

with the monomial basis $\{1, x, x^2, \ldots, x^n\}$ and the inner product $\langle p, q \rangle = \int_{-1}^1 p(x)g(x)\ dx$. Apply the Gram-Schmidt process to this setting will lead to a list of orthogonal polynomials $\{q_i(x)\}$ known as Legendre polynomials. In general,

$$q_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n}(x^2 - 1)^n.$$

Proposition 8.9. *Let $q_i(x)$ be the i-th Legendre polynomial. We have*

$$nq_n(x) = (2n-1)xq_{n-1}(x) - (n-1)q_{n-2}(x).$$

Definition 8.10. (Chebyshev polynomials) If we change the inner product to

$$\langle p, q\rangle = \int_{-1}^{1} p(x)q(x)\frac{1}{\sqrt{1-x^2}}\,dx,$$

then we will have a set of orthogonal polynomials $\{T_i(x)\}$ known as the Chebyshev polynomials. In general, we have

$$T_n(x) = \cos(n\arccos x).$$

Proposition 8.11. $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$.

Remark 8.12. As we mentioned earlier, the concept of 'inner product' is a generalised idea of distance. Under different question settings and real-life situations, using different inner products may help better understand the problems. Yet, a sad thing to know is that there is no universally correct way of identifying what inner products we shall choose, and any discussion on this, in the author's opinion, is far beyond the scope of this module. Therefore, readers just need to accept the idea that 'distance', in general, is not something universal just like what we are accustomed to in the real world.

8.6. **Chebyshev points.**

For the n-th Chebyshev polynomials $T_n(x)$, its roots are given by

$$\begin{aligned}\cos(n\arccos x) = 0 &\implies n\arccos x = (2k-1)\frac{\pi}{2} \text{ for } k \in \mathbb{Z}\\ &\implies x = \cos\left(\frac{(2k-1)\pi}{2n}\right) \text{ for } k \in \mathbb{Z}.\end{aligned}$$

These roots are known as Chebyshev points or Chebyshev nodes.

Remark 8.13. Clearly, Chebyshev points are not evenly spaced, i.e., the distance between any two consecutive points is not a constant. This provides a good example of why we should not have the mindset that interpolating functions can only be done by selecting sample points that are evenly spaced.

Remark 8.14. Based on the author's memory, during the lecture, the lecturer demonstrated how powerful these points are when we try to choose sample points to interpolate functions. Yet, as the author did not pay attention to the lecture, he is not able to give further details as of this draft.

8.7. **Taylor polynomial.**

To some extent, it interpolates $(a, f(a))$, $(a, f'(a))$, ..., $(a, f^{(n)}(a))$.

8.8. **Laguerre polynomial.**

These are bonus materials and will not be tested, so the author excludes it in the first draft.

8.9. **Hermite polynomial.**

Same as the previous subsection.

8.10. **Piecewise interpolation.**

In the previous sections, what we are dealing with are continuous functions. In fact, the condition of functions being 'continuous' is a really nice condition to have. It is so nice to the extent that we should not expect the functions we will be dealing with are continuous in general. In fact, there is a very interesting result from set theory: the number of continuous functions from $\mathbb{R}$ to $\mathbb{R}$ is as many as the size of the set of real numbers.

You may refer to lecture 17 for some examples on what will happen if the function is not continuous. Currently, what we do is to use one polynomial to interpolate the whole function, which fails as per example in the lecture slides. If we cannot use one polynomial for the whole function, what will happen if we use multiple polynomials, each with a low-degree, to pass through all the data points? This idea gives rise to the idea of splines.

The easiest method of employing splines is to use linear lines. For any two consecutive sample points, we just use a straight line to connect them. Thus for a set of n points, we just need to use $n - 1$ straight lines to interpolate them.

While this is a viable method, there is a lack of 'smoothness'. By the term 'smoothness', it commonly refers to the ability to differentiate a function in mathematics. Cubic splines are meant to address this lack of 'smoothness', which is to replace linear functions with cubic

polynomials to interpolate the given data points. In other words, we now need to find $n-1$ cubic polynomials, each with 4 parameters, to interpolate the given data points.

The general formula for a cubic polynomial is $p_i(t) = a_i + b_i(t-t_i) + c_i(t-t_i)^2 + d_i(t-t_1)^3$, and we need to determine a_i, b_i, c_i, and d_i. To determine these four parameters, we impose the following requirements on the spline:

[1] $p_i(t_i) = y_i$ and $p_i(t_{i+1}) = y_{i+1}$, which fixes one parameter a_i and gives a condition on the other three parameters for each polynomial;

[2] To ensure the smoothness, we take $p'_{i-1}(t_i) = p'_i(t_i)$ and $p''_{i-1}(t_i) = p''_i(t_i)$ to have two more conditions on the remaining three parameters, which along with (1) allows us to solve for all parameters except for those of p_1 and p_{n-1}.

[3] We cannot calculate p_1 and p_{n-1} since they both contain an endpoint. Hence, we need two more endpoint conditions. A commonly chosen one is $p''_1(t_1) = 0 = p''_{n-1}(t_n)$.

Definition 8.15. (Natural spline) If a cubic spline satisfies the requirement (3), it is called a natural spline.

We now analyse what these requirements give us.

The first requirement tells us that

$$y_{i+1} = y_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \tag{1}$$

for $1 \le i \le n-1$, where $h_i = t_{i+1} - t_i$.

The first half of the requirement [2] tells us that

$$0 = b_i + 2c_i h_i + 3d_i h_i^2 - b_{i+1} \tag{2}$$

for $1 \le i \le n-2$.

The later half of the requirement [2] tells us that

$$0 = 2c_i + 6d_i h_i - 2c_{i+1} \tag{3}$$

for $1 \le i \le n-2$.

From (3), we know that $d_i = \dfrac{c_{i+1} - c_i}{3h_i}$ for $1 \le i \le n-1$.

Sub this into (1). Define $\Delta_i = y_{i+1} - y_i$. We have

$$\begin{aligned} h_i b_i &= \Delta_i - c_i h_i^2 - d_i h_i^3 \\ b_i &= \frac{\Delta_i}{h_i} - c_i h_i - d_i h_i^2 \\ b_i &= \frac{\Delta_i}{h_i} - c_i h_i - \frac{h_i(c_{i+1} - c_i)}{3} \\ b_i &= \frac{\Delta_i}{h_i} - \frac{h_i}{3}(2c_i + c_{i+1}) \end{aligned}$$

for $1 \leq i \leq n-1$.

Therefore, as long as we can solve c_i for $1 \leq i \leq n-1$, we can get the corresponding b_i and d_i. Sub b_i and d_i into (2), we have

$$h_i c_i + 2(h_i + h_{i+1})c_{i+1} + h_{i+1}c_{i+2} = 3\left(\frac{\Delta_{i+1}}{h_{i+1}} - \frac{\Delta_i}{h_i}\right)$$

for $1 \leq i \leq n-3$.

Along with the natural cubic spline condition, we can write

$$\begin{pmatrix} 1 & 0 & 0 & & & \\ h_i & 2h_1 + 2h_2 & h_2 & \ddots & & \\ 0 & h_2 & 2h_2 + 2h_3 & h_3 & \ddots & \\ 0 & \ddots & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & 2h_{n-2} + 2h_{n-1} & h_{n-1} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} 0 \\ 3\left(\frac{\Delta_2}{h_2} - \frac{\Delta_1}{h_1}\right) \\ \vdots \\ 3\left(\frac{\Delta_{n-1}}{h_{n-1}} - \frac{\Delta_{n-2}}{h_{n-2}}\right) \\ 0 \end{pmatrix},$$

solving which gives us c_i, thus b_i and d_i.

Remark 8.16. This tridiagonal system is developed with reference to the textbook. The lecture notes gives a different system formulation, which the author doubts is equivalent to the above system. Yet, the author has not verified it.

9. Numerical differentiation

9.1. Overview.

Definition 9.1. (Derivative) Let f be a real-valued function. The derivative of $f(x)$, denoted by $f(x)$, is said to exist if the limit

$$\lim_{h\to 0} \frac{f(x+h)-f(x)}{h}$$

exists.

For computational purposes, we need to replace the limit with some small $h \in \mathbb{R}$, which leads to the first way of calculating derivative, i.e.,

$$f'(x) \approx \frac{f(x+h)-f(x)}{h},$$

which is known as the forward difference formula. An error analysis for the above method can be found at example 1.11.

Another way of looking at the formula and the sign of h will give

$$f'(x) \approx \frac{f(x)-f(x-h)}{h},$$

which is known as the backward difference formula.

9.2. Interpolation and derivative.

The above two formulae make use of two points to approximate a derivative, which gives fairly good results. Yet, since differentiation is generally ill-conditioned, we want to see whether there is a way to improve the methods.

If we use Newton basis functions to fit the points $(x-h, f(x-h)), (x, f(x)), (x+h, f(x+h))$, we will have a three-point centred difference formula, given by:

Definition 9.2. (Three-point centred difference formula) The centred difference formula for computing derivatives is

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

for some choice of h.

If we used Lagrange interpolation instead, we could obtain an alternative three-point formula

$$f'(x) \approx \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}.$$

Using similar techniques, we can derive formulae for more points. The more points we consider in a formula, the more accurate the result will be. Yet, the computation will also be more expensive. There is thus a trade-off between the accuracy and the computational costs.

9.3. **Extrapolation.**

An alternative way of obtaining higher-order approximations is Richardson extrapolation.

By Taylor's theorem, we have

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} - f'''(a_h)\frac{h^2}{6}$$

for some $a_h \in (x-h, x+h)$.

If we assume f''' is roughly constant near x, then it is in the same form of $Q \approx F(h) + Kh^2$, where K is a constant. We say this is an order 2 formula since the power of h in Kh^2 is 2. With some algebra, we have

$$\begin{aligned}
Q - F(h) \approx Kh^2 &\implies Q - F\left(\frac{h}{2}\right) \approx \frac{Kh^2}{4} \\
&\implies Q - F\left(\frac{h}{2}\right) \approx \frac{Q - F(h)}{4} \\
&\implies 4Q - 4F\left(\frac{h}{2}\right) \approx Q - F(h) \\
&\implies Q \approx \frac{4F\left(\frac{h}{2}\right) - F(h)}{3}.
\end{aligned}$$

Applying this to centred difference formula gives us

$$f'(x) \approx \frac{f(x-h) - 8f\left(x - \frac{h}{2}\right) + 8f\left(x + \frac{h}{2}\right) - f(x+h)}{6h},$$

which is a five-point formula (with the coefficient of $f(x)$ being 0).

10. Numerical integration

10.1. **Overview.**

Definition 10.1. (Riemann Sum) On the interval $[a, b]$, we define $h = \dfrac{b-a}{h}$ and set $x_k = a+hk$ for $0 \leq k \leq n$. We define

$$L_n = \sum_{k=0}^{n-1} hf(x_k)$$

$$R_n = \sum_{k=1}^{n} hf(x_k)$$

for $n \in \mathbb{Z}^+$. If we have

$$\lim_{n\to\infty} L_n = \lim_{n\to\infty} R_n = J,$$

we say that f is Riemann integrable and write $\displaystyle\int_a^b f(x)\ dx = J$.

Remark 10.2. There are multiple ways to define a definite integral, and what is presented here follows the lecture notes.

In this course, we are more interested in approximating a definite integral. Since one of the interpretations of a definite integral is the area under graph, we can use quadrilaterals to approximate the area thus the definite integral. This method is known as quadrature.

Definition 10.3. (n-point quadrature rule) An n-point quadrature rule is of the form

$$Q_n(f) = \sum_{i=1}^{n} w_i f(x_i),$$

where w_i is the weight and x_i is the node.

10.2. **Interpolation and Integration.**

Since among all the functions, we know how to calculate the definite integrals of polynomials for sure, we can use the method of interpolation to fit a polynomial to the function we want to integrate first.

In fact, we can choose the Lagrange basis to interpolate the function. Once this is done, we have

$$\begin{aligned} p(x) = \sum_{i=1}^{n} f(x_i) l_i(x) \implies \int_a^b p(x)\ dx &= \int_a^b \sum_{i=1}^{n} f(x_i) l_i(x)\ dx \\ \implies \int_a^b p(x)\ dx &= \sum_{i=1}^{n} \left(\int_a^b l_i(x)\ dx \right) f(x_i) \\ \implies \int_a^b p(x)\ dx &= \sum_{i=1}^{n} w_i f(x_i), \end{aligned}$$

where w_i is just the weight in the n-point quadrature rule.

Now, we seek a quadrature rule that integrates polynomials of degree less than n exactly. By taking the monomial basis and doing the computations, we have

$$\begin{aligned} \int_a^b 1\ dx &= b - a = w_1 \cdot 1 = \ldots = w_n \cdot 1 \\ \int_a^b x\ dx &= \frac{1}{2}(b^2 - a^2) = w_1 \cdot x_1 + \ldots + w_n \cdot x_1 \\ &\vdots \\ \int_a^b x^{n-1}\ dx &= \frac{1}{n}(b^n - a^n) = w_1 \cdot x_1^{n-1} + \ldots + w_n \cdot x_n^{n-1}, \end{aligned}$$

which leads to a linear system of equations with variables in w_i.

10.3. **Newton-Coates quadrature.**

In this quadrature method, we choose points $x_1, \ldots, x_n$ to be equally spaced, and define $h = x_i - x_{i-1}$.

When $n = 1$, we sample only one point $x_1 = \dfrac{b-a}{2}$, which corresponds to the interpolation by a constant polynomial.

Definition 10.4. (Midpoint rule) The midpoint rule is a 1-point open Newton-Coates quadrature rule which approximates the integral by

$$M(f) = (b-a) f\left(\frac{a+b}{2} \right).$$

When $n = 2$, we sample at $x_1 = a$ and $x_2 = b$, which corresponds to the interpolation by a linear polynomial.

Definition 10.5. (Trapezoid rule) The trapezoid rule is a 2-point closed Newton-Coates quadrature rule which approximates the integral by

$$T(f) = \frac{b-a}{2}\left(f(a) + f(b)\right).$$

Here, the linear through the points $(a, f(a))$, $(b, f(b))$ is given by

$$p(x) = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a},$$

integrating which gives the trapezoid rule.

Remark 10.6. In the above definitions, we say that midpoint rule is an open quadrature rule whereas trapezoid rule is a closed quadrature rule. The word 'open' here refers to the fact that we do not use endpoints in the approximation, while the word 'closed' means we do use endpoints in the approximation.

When $n = 3$, we sample at $x_1 = a$, $x_2 = \dfrac{a+b}{2} = m$, and $x_3 = b$, which corresponds to the interpolation by a quadratic polynomial. The interpolation polynomial is thus

$$p(x) = f(a)\frac{(x-m)(x-b)}{(a-m)(a-b)} + f(m)\frac{(x-a)(x-b)}{(m-a)(m-b)} + f(b)\frac{(x-a)(x-m)}{(b-a)(b-m)},$$

integrating which gives the Simpson's rule.

Definition 10.7. (Simpson's rule) The Simpson's rule is a 3-point closed Newton-Coates quadrature rule which approximates the integral by

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f(m) + f(b)\right).$$

However, Newton-Coates methods are based on interpolation with equally spaced nodes, and such interpolation methods can suffer from oscillations and are not guaranteed to converge.

10.4. **Clenshaw-Curtis quadrature.**

From the previous discussion, we know that Chebyshev points tend to give better results. Even better, we can use the extrema rather than the zeros of the Chebyshev polynomials.

These are referred to as the Clenshaw-Curtis method and practical Clenshaw-Curtis method respectively.

In fact, Clenshaw-Curtis method is equivalent to the change of variables

$$\int_{-1}^{1} f(x)\ dx = \int_{0}^{\pi} f(\cos\theta)\sin\theta\ d(\theta).$$

10.5. **Composite Newton-Coates quadrature.**

With some calculations in the lecture notes, when $n \geq 11$, at least one of the weights must be negative. Therefore, we turn to piecewise interpolation, which leads to composite Newton-Coates rule.

Basically, for all composite rules, we just divide the interval $[a, b]$ into k sub-intervals or panels with length $h = \dfrac{b-a}{k}$ and apply Newton-Coates quadrature rule to each of the sub-interval. In these sub-intervals, x_i is given by $a + ih$ for $0 \leq i \leq k$. Technically, there is nothing interesting except for the error analysis.

Definition 10.8. (Composite midpoint rule) The formula for the composite midpoint rule is

$$M_k(f) = h\sum_{i=1}^{k} f\left(\frac{x_{i-1}+x_i}{2}\right).$$

Definition 10.9. (Composite trapezoid rule) The formula for the composite trapezoid rule is

$$\begin{aligned} T_k(f) &= \frac{h}{2}\left[(f(a)+f(x_1))+(f(x_1)+f(x_2))+\ldots+(f(x_{k-1})+f(b))\right] \\ &= \frac{h}{2}\left[f(a)+f(b)+2\sum_{i=1}^{k-1} f(x_i)\right]. \end{aligned}$$

For the case of composite Simpson's rule, if we want to use Simpson's rule k times, we need to divide the interval into $2k$ sub-intervals with length $h = \dfrac{b-a}{2k}$ and take $x_i = a + ih$ for $0 \leq i \leq 2k$. Then, we use Simpson's rule to every pair of consecutive sub-intervals (without double counting).

Definition 10.10. (Composite Simpson's rule) The formula for the composite Simpson's rule is

$$\begin{aligned} S_{2k}(f) &= \frac{h}{3}\left[(f(a)+4f(x_1)+f(x_2))+(f(x_2)+4f(x_3)+f(x_4))+\ldots+(f(x_{2k-2})+4f(x_{2k-1})+f(b))\right] \\ &= \frac{h}{3}\left[f(a)+f(b)+4\sum_{i=1}^{k} f(x_{2i-1})+2\sum_{i=1}^{k-1} f(x_{2i})\right] \end{aligned}$$

10.6. **Adaptive quadrature.**

Since the previous methods use equally spaced sample points, it could lead to some problems when the function vary drastically in some parts of the domain but not the other.

To address the issues emerged, we rely on the idea of extrapolation. Given two quadrature rules P_m and Q_n, we apply both on the initial interval $[a, b]$. If the difference between the two rules is too large, then we divide $[a, b]$ in half and repeat the process. Once the approximation on an interval is good enough, we stop dividing that interval. This method is known as the adaptive quadrature.

10.7. **Gaussian quadrature.**

Definition 10.11. (Degree of precision) The degree of precision of a numerical integration method is the greatest integer k for which all degree k or less polynomials are integrated exactly by the method.

Proposition 10.12. *Newton-Cotes Methods of degree n have degree of precision n for odd n and $n+1$ for even n. Here, the first degree n refers to the degree of interpolation polynomial used.*

We wonder there is more power quadrature methods. Indeed, the answer is yes, and an example of it is the Gaussian quadrature.

Proposition 10.13. *Gaussian quadrature has degree of precision $2n+1$ when $n+1$ points are used, double that of Newton-Cotes.*

Gaussian quadrature makes use of the orthogonal functions.

Definition 10.14. (Orthogonal polynomial) The set of nonzero integrable functions $S_{[a,b]}$ on the interval $[a, b]$ is orthogonal on $[a, b]$ if for all $p(x), g(x) \in S$, we have

$$p(x) \neq g(x) \implies \int_a^b p(x)g(x) = 0.$$

Theorem 10.15. *If $\{p_0, \ldots, p_n\}$ is an orthogonal set of polynomials on the interval $[a, b]$, where $\deg(p_i) = i$, then $\{p_0, \ldots, p_n\}$ is a basis for the vector space of degree at most n polynomials on $[a, b]$.*

Theorem 10.16. *If $\{p_0, \ldots, p_n\}$ is an orthogonal set of polynomials on $[a, b]$ and if $\deg(p_i) = i$, then p_i has i distinct roots in the interval (a, b).*

Definition 10.17. (Gaussian quadrature) The formula for Gaussian quadrature is

$$\int_{-1}^{1} f(x)\ dx \approx \sum_{i=1}^{n} c_i f(x_i),$$

where $c_i = \int_{-1}^{1} L_i(x)\ dx$ for $1 \leq i \leq n$. Here, L_i is the i-th Lagrange basis.