

APPLIED NUMERICAL ANALYSIS : 151

YOUNG-JU LEE

ABSTRACT. This note is prepared to give a lecture on Applied Numerical Analysis.

1. ROOTFINDING

We consider the solution of nonlinear equation in one unknowns :

$$(1.1) \quad \text{Given a function } f, \text{ find a value for } x \text{ such that } f(x) = 0.$$

Such an x is called a zero of the function f or a root of the equation $f(x) = 0$. This problem is known as the root-finding problem.

If f is linear in x , i.e, $f = Ax + B$ for some constants A and B , the problem is easy to solve. $x = -B/A$. The difficulties come from the nonlinearity of f with respect to x . For example, if f is quadratic, still doable. However, if f is a cubic polynomial in x , then how would you do?

For nonlinear systems, there is rarely a direct method of solution, i.e., an algorithm that terminates at the exact solution. Hence we must use iterative methods which produce a sequence of approximate solutions x_0, x_1, \dots for which, hopefully, $\lim x_i$ exists and equals a root x of f .

The first such technique that will be discussed is the Bisection method.

1.1. The Bisection Method. The most basic algorithm for the root-finding problem is perhaps the bisection method. The main idea behind this algorithm is the well-known Intermediate Value Theorem.

Theorem 1.1. *Let f be a continuous function over the closed interval $[a, b]$, and let k be any real number that lies between the values $f(a)$ and $f(b)$. There there exists a real number c with $a < c < b$ such that $f(c) = k$.*

Note that this theorem just says that a continuous function on a closed interval assumes every value between the values achieved at the endpoints of the interval.

To demonstrate how the aforementioned IVT can be used, we take a simple example. Let $f(x) = x^3 + 2x^2 - 3x - 1$.

- The first task is to find an interval containing a zero. It can be achieved by finding say $[a, b]$ for which the following holds:

$$f(a)f(b) < 0.$$

Namely, such an interval should contain a zero (maybe more than one). For example, the interval $[1, 2]$ satisfies the aforementioned property, i.e., $f(1) = -1$ and $f(2) = 9$. Hence a zero lies in between $(1, 2)$.

- The second task is to systematically shrink the size of that root enclosing interval until we hit a zero. Most natural approach should be reducing the interval size by halving it and repeat this process. The mid-point of the shrunked interval shall be chosen as iterates x_i 's.

We can write such a process as the following psuedocode:

- (1) Input two endpoints a and b for which $f(a)f(b) < 0$ and set $i = 0$, $a_i = a$ and $b_i = b$.
- (2) Set $x_i = (a_i + b_i)/2$
 - If $f(x_i) = 0$, then x_i is a zero
 - Else check the sign of $f(a_i)f(x_i)$ and $f(x_i)f(b_i)$
 - If $f(a_i)f(x_i) < 0$, then goto (1) with $b_{i+1} = x_i$; $a_{i+1} = a_i$; $i = i + 1$;
 - If $f(x_i)f(b_i) < 0$, then goto (1) with $a_{i+1} = x_i$; $b_{i+1} = b_i$; $i = i + 1$;

Several remarks on the Bisection method are in order.

- Bisection method is called the enclosure method.
- The guess root in each iterations is the average of end points of closed interval containing a zero.
- Bisection finds only one root even if there are more.
- The stopping criterion should be used with the following choices
 - (1) $(b_i - a_i)/2 < tolerance$.
 - (2) $|f(x_i)| < tolerance$.
 - (3) $|x_i - x_{i-1}| < tolerance$.
 - (4) $|x_i - x_{i-1}|/|x_i| < tolerance$.

Some stopping criterion listed in the above might have some drawback. The most desired stopping criterion will be (4), which I will call **relative error**.

Example 1.1. Let $f(x) = (x - 1)^{10}$, $x = 1$ and $x_n = 1 + 1/n$. Show that $f(x_n) < 10^{-3}$, whenever $n > 1$ but that $|x - x_n| < 10^{-3}$ requires that $n > 1000$. Now consider the iterates $\{x_i\}_i$ defined by $x_i = \sum_{k=1}^i (1/k)$. We can show that x_i diverges. But $\lim_{i \rightarrow \infty} |x_i - x_{i-1}| \rightarrow 0$.

The main question that needs to be addressed is whether or not the bisection procedure will provide an approximate solution that is close enough to the real solution. The answer is yes.

Theorem 1.2. *Suppose that $f \in C[a, b]$, and $f(a)f(b) < 0$. Then the bisection method generates a sequence $\{p_n\}$ approximating a root p of f with*

$$|p_n - p| \leq \frac{b - a}{2^n}.$$

Proof. For each $n \geq 1$,

$$(1.2) \quad b_n - a_n = \frac{1}{2^{n-1}}(b - a), \text{ and } p \in (a_n, b_n).$$

Since $p_n = \frac{1}{2}(a_n + b_n)$ for all $n \geq 1$, it follows that

$$|p_n - p| \leq \frac{1}{2}(b_n - a_n) = \frac{b - a}{2^n}.$$

□

We conclude that the Bisection method always converges to p . Furthermore, based upon the theorem discussed above. One can deduce how many iterations shall be needed for any given tolerance.

Example 1.2. *Figure out how many iterations are necessary to solve*

$$f(x) = x^3 + 4x^2 - 10 = 0$$

with accuracy 10^{-3} using $a = 1$ and $b = 2$. In other words, we wish $|p - p_n| < 10^{-3}$.

Note that we have a bound that

$$|p - p_n| \leq \frac{1}{2^n}.$$

We wish to find n for which $1/2^n < 10^{-3}$. Namely

$$2^n > 10^3 \Rightarrow n \log 2 > 3 \Rightarrow n > 3/\log 2 \approx 9.96$$

We then conclude that about 10 iterations are needed to achieve the tolerance.

1.1.1. *Some notes on the speed of convergence of sequences.* Let x_i be a sequence that converges to x . For $k > 1$, we say that the sequence converges to x with order k if there is a constant C and a number N such that

$$\|x_{i+1} - x\| \leq C\|x_i - x\|^k, \quad \forall i \geq N.$$

For $k = 1$, a sequence would then be said to be convergent linearly to x if there is $r < 1$ and N such that

$$\|x_{i+1} - x\| \leq r\|x_i - x\|, \quad \forall i \geq N.$$

This one is norm-dependent. In order to avoid the norm-dependency, we note that the above definition imply that there exists C such that $\|x_i - x\| \leq Cr^i$ for all i . Observe that it holds true that $\|x_i - x\| \leq r^{i-N}\|x_N - x\|$ for all $i > N$. Equivalently, $\|x_i - x\| \leq C_0r^i$ for $i > N$ where $C_0 = r^{-N}\|x_N - x\|$. Set $C = \max\{C_0, \max_{0 \leq i \leq N} \|x_i - x\|/r^i\}$. We shall take this as our definition for the linear convergence.

Namely, in order to find the order of convergence k , we need to show that $|x_{i+1} - x|$ is approximately equal to some constant c times the previous error $|x_i - x|^k$ when i is sufficiently large. If k is one, then one need to show that the constant c is less than one to validate the convergence is at least linear.

Question 1.1. *How to obtain a convergence order given errors?*

$$k = \frac{\ln(e_{n+1}/e_n)}{\ln(e_n/e_{n-1})}$$

How to obtain the convergence rate given errors?

$$r = \frac{e_{n+1}}{e_n}.$$

Example 1.3. *If $e_1 = 5.000 * 10^{-1}$, $e_2 = 2.500 * 10^{-1}$, $e_3 = 1.2500 * 10^{-1}$, $e_4 = 6.2500 * 10^{-2}$, $e_5 = 3.12500 * 10^{-2}$, $e_6 = 1.5625 * 10^{-2}$, $e_7 = 7.8125 * 10^{-3}$, then what is the convergence speed?*

Note that the calculation done in the example is based on the upper bound given in the theorem. It takes fewer iterations. We remark that in each iteration of bisection method, we cut the interval in half. For speediness, it is best to start with an interval $[a, b]$ as small as you can, but it is difficult part.

Example 1.4. *Consider to find a root of $f(x) = 2x^3 - x^2 + x - 1 = 0$. We see that $f(-4) < 0$ and $f(4) > 0$. However, it is the case that $f(0) < 0$ and $f(1) > 0$. Note that $[-4, 4]$ is 8 times bigger than $[0, 1]$ hence we save 3 iterations.*

1.2. Newton's Method. Newton's method for finding a root is one of the powerful tool. We will discuss two ways of introducing the idea behind it.

Suppose that $f \in C^2[a, b]$ and let $p_0 \in [a, b]$ be an approximation to the solution p to $f(x) = 0$ such that $f'(p_0) \neq 0$ and furthermore $|p - p_0|$ is small enough. We then take the Taylor expansion of f around p_0 and evaluate at p to obtain

$$f(p) = f(p_0) + f'(p_0)(p - p_0) + \frac{f''(\xi(p))}{2}(p - p_0)^2,$$

where $\xi(p)$ is in between p and p_0 . Since $f(p) = 0$, we have

$$0 = f(p_0) + f'(p_0)(p - p_0) + \frac{f''(\xi(p))}{2}(p - p_0)^2,$$

Now since $|p - p_0|^2$ much smaller than $|p - p_0|$, it can be ignored to obtain that

$$0 \approx f(p_0) + f'(p_0)(p - p_0)$$

Now if we set

$$f(p_0) + f'(p_0)(\hat{p} - p_0) = 0$$

Then we expect that \hat{p} should be closer to p than p_0 since p_0 is a wild guess while \hat{p} has been obtained by the aforementioned reasoning. Indeed, it is the case (will be shown later and also graphically soon) and \hat{p} shall serve as the next iterate of p_0 .

$$(1.3) \quad p_1 = \hat{p} = p_0 - \frac{f(p_0)}{f'(p_0)}.$$

Repeating the above procedure, we can have the following recursive relation starting from p_0 :

$$(1.4) \quad p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}, \quad \forall n \geq 1.$$

The functional iteration formula (1.4) indicates that there can be another interpretation of the Newton's method. Note that the p^n is nothing but a zero of the tangent line of curve $y = f(x)$ that passes through $(p_{n-1}, f(p_{n-1}))$, namely,

$$y - f(p_{n-1}) = f'(p_{n-1})(x - p_{n-1}).$$

- (1) Input initial approximation p_0 ; Set $n = 1$;
- (2) Compute the next iterate as follows:

$$(1.5) \quad p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}.$$

- (3) If $|p_n - p_{n-1}|/|p_n| < tol$ then we found an approximate solution p_n , else set $n = n + 1$; and goto (2).

Example 1.5. What is the Newton's iteration scheme for finding a root of $f(x) = x^3 - 2x^2 - 5$. Note that $f'(x) = 3x^2 - 4x$. Hence,

$$(1.6) \quad p_n = p_{n-1} - \frac{p_{n-1}^3 - 2p_{n-1}^2 - 5}{3p_{n-1}^2 - 4p_{n-1}}.$$

Example 1.6. What is the Newton's iteration scheme for finding a root of $\frac{\sin x}{x} = \frac{1}{2}$.

Find a Newton's iteration scheme to find an approximation to $^4\sqrt{3}$.

Example 1.7. Let $f(x) = \cos x - x$. Write a Newton's iteration scheme for finding a root for $f(x)$.

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

We now study the convergence of the Newton's method.

Theorem 1.3. *Let $f \in C^2[a, b]$. If $p \in [a, b]$ is such that $f(p) = 0$ and $f'(p) \neq 0$, then there exists a $\delta > 0$ such that Newton's method generates a sequence $\{p_n\}$ converging to p for any initial approximation $p_0 \in [p - \delta, p + \delta]$. Furthermore, the convergence rate is quadratic.*

Proof. One may view the Newton's method as the functional iteration scheme with

$$p_n = g(p_{n-1}) \quad n \geq 1,$$

where

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

Note that $g(p) = p$ and also

$$(1.7) \quad |p_n - p| = |p_n - g(p)| = |g(p_{n-1}) - g(p)| = |g'(\xi(p_{n-1}))||p_{n-1} - p|,$$

where $\xi(p_{n-1})$ lies in between p and p_{n-1} . Our goal is to find $\delta > 0$ such that for all $x \in [p - \delta, p + \delta]$, $|g'(x)| \leq k < 1$. Then we are done! (Why?). This will end the proof since then with the initial guess $p_0 \in (p - \delta, p + \delta)$, we see that

$$\begin{aligned} |p_n - p| &= |g(p_{n-1}) - p| = |g(p_{n-1}) - g(p)| \\ &= |g'(\xi(p_{n-1}))||p_{n-1} - p| \leq k|p_{n-1} - p| \\ &\leq \cdots \leq k^n |p_0 - p|. \end{aligned}$$

Under the assumption $f'(p) \neq 0$, there exists $\delta_1 > 0$ such that

$$(1.8) \quad f'(x) \neq 0, \quad \forall x \in [p - \delta_1, p + \delta_1].$$

Note that due to (1.8), $g'(x)$ is well-defined on $[p - \delta_1, p + \delta_1]$ and it is given as follows:

$$g'(x) = 1 - \frac{f'(x)f(x) - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2}.$$

Note that

$$g'(p) = 0.$$

Hence, there exists a constant $0 < \delta < \delta_1$, for which

$$(1.9) \quad |g'(x)| \leq k < 1, \quad \forall x \in [p - \delta, p + \delta].$$

Note that

$$(1.10) \quad \begin{aligned} e_{n+1} &= |p_{n+1} - p| \\ &= \left| p - \left(p_n - \frac{f(p_n)}{f'(p_n)} \right) \right| \end{aligned}$$

Note that since p_0 is sufficiently close to p and $\{p_n\}$ are all close to p , and hence $f'(p_n) \neq 0$ for all n . Now we Taylor expand $f(p)$ around p_n to obtain that

$$f(p) = f(p_n) + f'(p_n)(p - p_n) + \frac{f''(\xi_n)}{2}(p - p_n)^2.$$

Since $f(p) = 0$, we have

$$f(p_n) = -f'(p_n)(p - p_n) - \frac{f''(\xi_n)}{2}(p - p_n)^2.$$

Plugging this expression into the equation (1.10), we obtain that

$$\begin{aligned} e_{n+1} &= \left| p - \left(p_n - \frac{-f'(p_n)(p - p_n) - \frac{f''(\xi_n)}{2}(p - p_n)^2}{f'(p_n)} \right) \right| \\ &= \left| p - \left(p_n + (p - p_n) + \frac{f''(\xi_n)}{2f'(p_n)}(p - p_n)^2 \right) \right| \\ &= \left| \frac{f''(\xi_n)}{2f'(p_n)}(p - p_n)^2 \right| \end{aligned}$$

Namely, we have

$$\lim_{n \rightarrow \infty} \frac{e_{n+1}}{e_n^2} = \lim_{n \rightarrow \infty} \left| \frac{f''(\xi_n)}{2f'(p_n)} \right| = \left| \frac{f''(p)}{2f'(p)}(p - p_n)^2 \right|.$$

This completes the proof. \square

Remark 1.1. Note that if $f'(p) = 0$, then the problem immediately arises in the convergence. The function f is very smooth near by zero.

Remark 1.2. It is crucial how to choose the initial iterate, p_0 so that at least $|p_1 - p| < |p_0 - p|$. First of all, it may not be less than one and hence break down in the convergence if one chooses a crude initial iterate. Consider the problem to solve $xe^{-x} = 0$ with different initial guess. Secondly, the iterative process can be proceeded only if $f'(p_n)$ is not zero and also the evaluation is available in each iteration.

The theory does not help in choosing p_0 . In practice, we guess p_0 and see if it is convergent or divergent.

Example 1.8. Consider the root finding problem for $f(x) = x^3 - x - 3$ with initial condition -3 (Cycling!)

The major drawback is in whether or not the initial condition is nearby the root. Another drawback should be in that the evaluation of f and also f' should be expensive.

1.3. Secant Method. Recall the Newton's method is given by

$$(1.11) \quad p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})}.$$

The main drawback of Newton's method (1.11) lies in the computation of $f'(p_n)$ for each n . We use the following approximation:

$$(1.12) \quad f'(p_{n-1}) = \lim_{x \rightarrow p_{n-1}} \frac{f(x) - f(p_{n-1})}{x - p_{n-1}} \approx \frac{f(p_{n-2}) - f(p_{n-1})}{p_{n-2} - p_{n-1}}.$$

By replacing $f'(p_{n-1})$ by the approximation given in (1.12), we obtain that

$$(1.13) \quad p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-2} - p_{n-1})}{f(p_{n-2}) - f(p_{n-1})}, \quad \forall n \geq 2.$$

This functional iteration (1.13) is called the secant method. Note that we need two starting values, p_0 and p_1 .

Example 1.9. Write a functional iteration of the Secant method to find a solution to $x = \cos x$.

Example 1.10. Write a functional iteration of the Secant method to find a solution to $\sin x - e^{-x} = 0$.

Remark 1.3. The secant method is not enclosure method and hence it is similar to Newton's method. The drawback of the Newton method or the Secant method is that the convergence is guaranteed only if the initial iterate p_0 is close to the real solution. In this regard, the bisection method can be used effectively since it is always convergent once the initial closed interval that contains a zero is found. Moreover, it can be used to shrink the interval size. Having used the bisection method and sufficiently small interval that contains zero, we can now apply the Newton's method or the Secant method. Why does it help? Since Newton's method is faster than the bisection method.

In the preceding discussion, we made the restriction that $f'(p) \neq 0$, where p is the solution to $f(x) = 0$. Let us look at details arising if $f'(p) = 0$.

Definition 1.1. Assume that $f(x)$ and its derivatives $f'(x), \dots, f^{(m)}$ are defined and continuous on an interval about p . A solution p of $f(x) = 0$ is a zero of multiplicity m of f if and only if $f(p) = f'(p) = \dots = f^{(m-1)}(p) = 0$ but $f^{(m)}(p) \neq 0$.

Example 1.11. Find the multiplicities of roots for each of the following:

- (1) $(x - 2)^2$, root at $x = 2$.
- (2) $x^4 - 6x^3 + 13x^2 - 12x + 4$, root at $x = 2$
- (3) $\sin(x)$, root at $x = 0$

- (4) $x \cos(x)$, root at $x = 0$
 (5) $x^2 \sin(x)$, root at $x = 0$.
 (6) $e^x - x - 1$, root at $x = 0$.

Theorem 1.4. *If p is a root of multiplicity m of $f(x)$, then $f(x)$ can be written as $f(x) = (x - p)^m q(x)$ with $\lim_{x \rightarrow p} q(x) \neq 0$.*

Proof. If p is a root of multiplicities m , then by definition, $f(p) = \dots = f^{(m-1)}(p) = 0$ with $f^{(m)}(p) \neq 0$. Expand $f(x)$ around p to obtain that

$$\begin{aligned} f(x) &= f(p) + f'(p)(x - p) + \dots + \frac{f^{(m+1)}(p)}{(m+1)!}(x - p)^{m+1} + \dots \\ &= (x - p)^m \left(\frac{f^{(m)}(p)}{m!} + \frac{f^{(m+1)}(p)}{(m+1)!}(x - p) + \dots \right) = (x - p)^m q(x). \end{aligned}$$

Note that $\lim_{x \rightarrow p} q(x) \neq 0$. This completes the proof. \square

The quadratic convergence has been guaranteed if p is a simple zero for $f(x)$. One method of handling the problem of multiple roots is to define

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

If p is a zero of f of multiplicity m and $f(x) = (x - p)^m q(x)$, then

$$\mu(x) = (x - p) \frac{q(x)}{mq(x) + (x - p)q'(x)}.$$

Moreover, p is a simple zero of μ . (Why?)

Remark 1.4. *To show p is a simple zero for μ , we may set $f(x) = (x - p)^m q(x)$ and $f'(x) = (x - p)^{m-1} r(x)$, for which $q(p) \neq 0$ and $r(p) \neq 0$. We now observe that*

$$\mu(x) = f(x)/f'(x) = (x - p) \frac{q(x)}{r(x)},$$

where $q(p)/r(p) \neq 0$. This implies what we want.

Example 1.12. *If $f(x) = x^2$, then $x = 0$ is double root, but $\mu(x) = f(x)/f'(x) = x/2$ has a simple root $x = 0$.*

Newton's method can then be applied to μ to give

$$(1.14) \quad g(x) = x - \frac{\mu(x)}{\mu'(x)}$$

$$(1.15) \quad = x - \frac{f(x)/f'(x)}{[f'(x)]^2 - [f(x)][f''(x)]/[f'(x)]^2}$$

$$(1.16) \quad = x - \frac{f(x)f'(x)}{[f'(x)]^2 - f(x)f''(x)}.$$

In practice, multiple roots can cause serious round-off problems since the denominator of the equation (1.16) consists of the difference of two numbers that are both close to zero.

2. ROUND-OFF ERRORS AND COMPUTER ARITHMETIC

Some motivating examples from my computer arithmetic are

Example 2.1. *In computer arithmetics,*

$$1.00000000000000001 - 1 = 0$$

although the answer should be 0.00000000000000001. Furthermore,

$$(1.00000000000000001 - 1) * 10^{117} = 0$$

although the answer should be 10^{100} .

$$\sqrt{3} = 1.73205080756888$$

but $(1.73205080756888)^2 = 3.000000000000001$ with 13 zeros in it.

What went wrong? To understand why this is the case, we must explore the world of finite-digit arithmetic. Arithmetic performed by a machine is different from the arithmetic we do in our heads in many cases. For us, $2 + 2 = 4$, $4 \times 4 = 16$ and $(\sqrt{3})^2 = 3$. But for the computational world $(\sqrt{3})^2 \neq 3$. This is because each representable number has only a fixed and finite number of digits.

In most cases, it is fair to say that the machine arithmetic is satisfactory and passes without notice or concern, but at times, problems might arise.

Definition 2.1. *The error that is produced when a calculator or computer is used to perform real number calculations is called the round-off error.*

The two most common floating-point binary storage formats used by Intel processors and later standardized by IEEE organizations are Both formats use essentially the same method for storing floating-point binary numbers, so we will use the Long Real as an example in this discussion. PCs store a real number by so-called long real. It is 64 bit (binary digit) long, namely 64 1's and 0's.

IEEE Short Real : 32 bits	1 bit for the sign, 8 bits for the exponent, and 23 bits for the mantissa. Also called single precision.
IEEE Long Real: 64 bits	1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa. Also called double precision.

TABLE 2.1. Floating-Point Binary

2.1. Review on binary (base 2) notation. A binary number is a sequence of 1's and 0's, where the places represent powers of 2. For example, the binary number 1011001 actually means from right to left,

$$1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

which is

$$64 + 16 + 8 + 1 = 89.$$

In our usual base 10 notation, it is

$$8 \times 10^1 + 9 \times 10^0.$$

Another example: 111111, which corresponds to 63.

- **The Sign**, First bit : The sign of a binary floating-point number is represented by a single bit. A 1 bit indicates a negative number, and a 0 bit indicates a positive number. A sign indicator denoted by s .
- **The Exponent**, Next 11 bit : Exponent denoted by c and called the characteristic (See the remark below).
- **The Mantissa**, Next 52 bit : Binary Fraction, f called the mantissa.

It is useful to consider the way decimal floating-point numbers represent their mantissa. Using -3.154×10^5 as an example, the sign is negative, the mantissa is 3.154, and the exponent is 5. The fractional portion of the mantissa is the sum of each digit multiplied by a power of 10:

$$.154 = 1 \frac{1}{10} + 5 \left(\frac{1}{10} \right)^2 + 4 \left(\frac{1}{10} \right)^3.$$

A binary floating-point number is similar. For example, we consider the number $+11.1011 \times 2^3$. Before storing a floating-point binary number correctly, we need to normalize it. The process is basically the same as when normalizing a floating-point decimal number. For example, decimal 1234.567 is normalized as 1.234567×10^3 by moving the decimal point so that only one digit appears before the decimal.

The number $+11.1011 \times 2^3$ shall be normalized into $+1.11011 \times 2^4$, for which the sign is positive, the mantissa is 1.11011, and the exponent is 4. The fractional portion of the mantissa is the sum of successive powers of 2. In our example, it is

Binary Value	Normalized As	Exponent	Biased Exponent
1101.101	1.101101	3	1026
.00101	1.01	-3	1020
1.0001	1.0001	0	1023
10000011.0	1.0000011	7	1030

TABLE 2.2. Normalizing the Mantissa

expressed as:

$$.1011 = 1 \frac{1}{2} + 0 \left(\frac{1}{2}\right)^2 + 1 \left(\frac{1}{2}\right)^3 + 1 \left(\frac{1}{2}\right)^4 .$$

Note the following formula holds true :

$$\sum_{n=0}^N 2^n = 2^{N+1} - 1.$$

Namely, the exponent of 11 binary digits gives a range of 0 to $2^{11} - 1 = 2047$, since the largest exponent will be

$$11111111111 = 2^{10} + 2^9 + \dots + 2^0 = 2^{11} - 1 = 2048 - 1 = 2047.$$

The exponent is always positive. However, using only positive integers for the exponent would not permit an adequate representation of numbers with small magnitude and to ensure that numbers with small magnitude are equally representable, 1023 is subtracted from the characteristic, so the range of the exponent is actually from -1023 to 1024. In conclusion, using this system, a floating-point number is given in the following form:

$$(-1)^s 2^{c-1023} (1 + f).$$

Namely, we store the biased exponent.

Remark 2.1. *Note that the number of decimal points that can be stored is related to the fractional portion of the mantissa. With base, 2 and since the smallest decimal point number that can be stored is in base two will be 2^{-52} , which is about $2.e - 16$, we can assume that any normalized number represented in this system has at most 16 decimal digits of precision.*

Example 2.2. *Consider, for example, the machine number*

$$0 \ 10000000011 \ 10111001000100000 \dots .$$

The leftmost bit is zero, which indicates that the number is positive. The next 11 bits gives the characteristic:

$$c = 1 \times 2^{10} + \dots + 1 \times 2 + 1 \times 2^0 = 1027.$$

The exponent part of the number is therefore $2^{1027-1023} = 2^4$. The final 52 bits specify that the mantissa is

$$f = 1 \left(\frac{1}{2}\right)^1 + 1 \left(\frac{1}{2}\right)^3 + 1 \left(\frac{1}{2}\right)^4 + 1 \left(\frac{1}{2}\right)^5 + 1 \left(\frac{1}{2}\right)^8 + 1 \left(\frac{1}{2}\right)^{12}$$

Written as a decimal floating point number, it is

$$27.56640625.$$

Next smallest machine number is

$$0 \ 10000000011 \ 10111001000011111 \dots$$

and next largest machine number is

$$0 \ 10000000011 \ 10111001000100000 \dots 1.$$

In decimal, these are

$$27.5664062499999982 \dots \text{ and } 27.5664062500000017 \dots$$

What about the elements of the real number between these? Effectively, they get set to the closest machine number.

2.2. Overflow/Underflow. Smallest normalized positive machine number : Take $s = 0$, $c = 1$ and $f = 0$

$$(-1)^0 2^{1-1023} (1 + 0) \approx 10^{-308}.$$

The largest normalized positive machine number can be obtained by setting $s = 0$, $c = 2046$ and $f = 1 - 2^{-52}$:

$$(-1)^0 2^{1023} (2 - 2^{-52}) \approx 10^{308}.$$

Numbers occurring in calculations that have a magnitude less than 2^{-1022} get set to zero which is called the underflow. Numbers greater than $2^{1023}(2 - 2^{-52})$ result in overflow and typically cause the computations to stop. Note that two representations for number zero are given as follows:

- (positive zero) $s = 0$, $c = 0$ and $f = 0$ (smallest positive number)
- (negative zero) $s = 1$, $c = 0$ and $f = 0$ (largest negative number).

Namely, $\pm 2^{-1023}$.

2.3. Error in number storage. For simplicity, we assume that machine numbers are represented in the normalized decimal floating point form (k-digit decimal machine numbers) with some abuse of definition :

$$\pm 0.d_1 d_2 \dots d_k \times 10^n, \quad 1 \leq d_1 \leq 9 \text{ and } 0 \leq d_i \leq 9,$$

where $i = 2, \dots, k$.

Note that the fractional portion of the mantissa can only be accurate 2^{-52} , in decimal, which is about 10^{-16} , namely, when storing true decimal numbers y , it should be converted to machine decimal number in general $fl(y)$ called the floating-point form. In general, it is done by chopping or rounding.

Chopping : $y = 0.d_1d_2 \cdots d_k d_{k+1} d_{k+2} \cdots \times 10^n$ becomes $fl(y) = 0.d_1d_2 \cdots d_k \times 10^n$, i.e., keep only first k -digits.

Rounding : Add $5 \times 10^{n-(k+1)}$ to y and then chop the result to obtain a number of the form : $fl(y) = 0.\delta_1\delta_2 \cdots \delta_k \times 10^n$. Namely, when rounding, if $d_{k+1} \geq 5$, then we add 1 to d_k to obtain $fl(y)$; that is we round up. When $d_{k+1} < 5$, we merely chop off all but the first k digits; that is we round down. When rounding down, $\delta_i = d_i$ for $i = 1, \cdots, k$.

Example 2.3. *The decimal expression for π is*

$$\pi = 3.14159265 \cdots$$

The normalized decimal form is

$$\pi = 0.3141592 \cdots \times 10^1.$$

The float-point form of π using five digit chopping is

$$fl(\pi) = 0.31415 \times 10^1 = 3.1415.$$

Rounding however gives

$$fl(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416.$$

The error resulting from the chopping or rounding is called round-off error.

Definition 2.2. *If p^* is an approximation to p , the absolute error is $|p - p^*|$ and the relative error is $|p - p^*|/|p|$, provided that $p \neq 0$.*

Example 2.4.

	p	p^*	$ p - p^* $	$ p - p^* / p $
(2.1)	0.3000×10^1	0.3100×10^1	0.1	$0.333\bar{3} \times 10^{-1}$
	0.3000×10^{-3}	0.3100×10^{-3}	0.1×10^{-4}	$0.333\bar{3} \times 10^{-1}$
	0.3000×10^4	0.3100×10^4	0.1×10^3	$0.333\bar{3} \times 10^{-1}$

What we can conclude is that although relative error is the same, the absolute errors widely vary. Relative error may be more meaningful.

Example 2.5. *Find the largest interval in which p^* must lie to approximate π with relative error at most 10^{-4} .*

$$\frac{|\pi - p^*|}{\pi} \leq 10^{-4}.$$

Hence

$$|\pi - p^*| \leq \pi 10^{-4} \Leftrightarrow \pi - \pi \times 10^{-4} \leq p^* \leq \pi + \pi 10^{-4}$$

More examples

Example 2.6. Find the largest interval in which p^* must lie to approximate 100π with relative error at most 10^{-4} .

Example 2.7. Find the largest interval in which p^* must lie to approximate 100π with absolute error at most 10^{-4} .

We shall now consider how much error is introduced by k -digit chopping for

$$y = 0.d_1d_2 \cdots d_k d_{k+1} \cdots \times 10^n.$$

The relative error is given by

$$\begin{aligned} \text{rel error} &= \frac{|0.d_1d_2 \cdots d_k d_{k+1} \cdots \times 10^n - 0.d_1d_2 \cdots d_k \times 10^n|}{|0.d_1d_2 \cdots d_k d_{k+1} \cdots \times 10^n|} \\ &= \frac{|0.d_{k+1}d_{k+2} \cdots \times 10^{n-k}|}{|0.d_1d_2 \cdots d_k d_{k+1} \cdots \times 10^n|} \\ &= \frac{|0.d_{k+1}d_{k+2} \cdots|}{|0.d_1d_2 \cdots d_k d_{k+1} \cdots|} 10^{-k} \end{aligned}$$

Upper bound on nominator in the last quantity is one and the lower bound in the denominator is 0.1. Hence the relative error is bounded by the following :

$$\text{rel error} \leq \frac{1}{0.1} 10^{-k} = 10^{-k+1}.$$

For the rounding error, it is $0.5 \times 10^{-k+1}$, which will be assigned as Homework. (Matlab seems to take this storage).

Remark 2.2. This relative error is mainly dependent of the digits we keep in the normalized form. For instance, for 3 digit chopping on the real number, 0.3141592×10^1 , the maximum relative error will be 0.01 and also for the number 0.3141592×10^3 , we have same maximum relative error.

2.4. Error in Computer Arithmetics. We shall assume that the floating-point representations $fl(x)$ and $fl(y)$ are given for the real numbers x and y . The notation \oplus, \ominus, \otimes , and \oslash represent machine addition, subtraction, multiplication and division operations, respectively. A finite digit arithmetic is given by

$$\begin{aligned} x \oplus y &= fl(fl(x) + fl(y)), & x \otimes y &= fl(fl(x) \times fl(y)) \\ x \ominus y &= fl(fl(x) - fl(y)), & x \oslash y &= fl(fl(x)/fl(y)). \end{aligned}$$

The arithmetic described above corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its finite-digit floating-point representation.

Example 2.8. *Let us consider 5 digit chopping: Let $u = 0.714251$ and $x = 5/7 \approx 0.714285714\dots$. In such a case :*

$$\begin{aligned} fl(u) &= 0.71425 \times 10^0 \\ fl(x) &= 0.71428 \times 10^0 \end{aligned}$$

Now with

$$\begin{aligned} x \ominus u &= fl(fl(x) - fl(u)) = fl(0.00003 \times 10^0) = 0.30000 \times 10^{-4} \\ x - u &= 0.0000347142857\dots = 0.34714\dots \times 10^{-4}. \end{aligned}$$

Absolute error is $0.471\dots \times 10^{-5}$ but the relative error is approximately 0.1358, which is 13.5%

Now we consider a multiplication by a large number, for example $v = 98765.9$ for which $fl(v) = 0.98765 \times 10^5$.

$$(x \ominus u) \otimes v = 0.29629 \times 10^1$$

The actual value is 0.34285×10^1 , hence the absolute error is 0.465 but the relative error is 0.1358 still 13.5%.

Remark 2.3. *Several Observations are in order:*

- *Subtracting two nearly equal number results in large relative error.*

$$12345678912345678 - 12345678912345677 = 1 \text{ real answer}$$

$$12345678912345678 - 12345678912345677 = 2 \text{ scilab answer}$$

Here, the abs error is 1, but relative error is $|2 - 1|/1 = 1$

- *Subsequent multiplication by large number or division by small number magnifies absolute error.*

$$100(12345678912345678 - 12345678912345677) = 100 \text{ real answer}$$

$$100(12345678912345678 - 12345678912345677) = 200 \text{ scilab answer}$$

Abs error is now 100 but the relative error is still $|200 - 100|/|100| = 1$.

- *Addition of large and small number results in large absolute error (but small relative error).*

$$10000000000000000 + 100 = 10000000000000100.$$

But scilab shows

$$10000000000000000 + 100 = 10000000000000000.$$

Hence relative error is now $100/100000000000000100 = 1.10^{-15}$. but absolute error is 100.

Most common error-producing calculations involves the subtraction of nearly equal numbers. Hence, we conclude that such an operator should be avoided. Remember the Newton's method for the multiple root case. This operation is not very accurate since we need to subtract nearly equal numbers. Such a price should be often paid to overcome some known drawbacks of the algorithm.

Example 2.9. *Try the following example: $0.1207 \times 10^1 + 0.9654 \times 10^{-3}$ by 4 digit chopping.*

$$\begin{aligned} 0.1207 \times 10^1 + 0.9654 \times 10^{-3} &= 0.12079654 \times 10^1 \\ fl(0.1207 \times 10^1 + 0.9654 \times 10^{-3}) &= fl(0.12079654 \times 10^1) = 0.1207 \times 10^1. \end{aligned}$$

The absolute error will be $0.96539.. \times 10^{-5}$. The relative error will be $0.96539.. \times 10^{-5} / 0.12079654 \times 10^1 = 0.799195... \times 10^{-5}$. Therefore, the relative error is not much big.

The natural question is how to avoid such a numerical inaccuracy ? The answer is simple, avoid such a calculations as much as possible !.

The loss of accuracy due to round-off error can be avoided by a reformulation of the problem, as illustrated by the next example.

Recall that for $ax^2 + bx + c = 0$, we have the following two solution

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Let us assume that $a = 1, b = 62.10$ and $c = 1$, namely, we are considering to solve $x^2 + 62.10x + 1 = 0$ with 4 digit chopping. Note that the true solutions are approximately

$$x_1 = -0.01610723 \text{ and } x_2 = -62.08390.$$

Observe that b^2 is large compared to $4ac$, hence $\sqrt{b^2 - 4ac}$ will be very close to b . More precisely,

$$\sqrt{b^2 - 4ac} = \sqrt{(62.10)^2 - 4(1.000)(1.000)} = 62.0678 = 0.620678 \times 10^2.$$

Hence four digit chopping gives us 0.6206×10^2 . We then have

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.0200.$$

The relative error is then approximately

$$\begin{aligned} \frac{|-0.01611 + 0.0200|}{|-0.01611|} &\approx 2.4 \times 10^{-1}. \\ fl(x_2) &= \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.08. \end{aligned}$$

Year	1940	1950	1960	1970	1980	1990
Population (in 1000)	132,165	151,326	179,323	203,302	226,542	249,633

TABLE 3.1. Population in Years

Hence

$$\frac{|-62.08390 + 62.08|}{|-62.08390|} \approx 6.2 \times 10^{-5}.$$

The remedy is to reformulate the formula especially for x_1 as follows:

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

From this formula, we get

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.016100.$$

The relative error is now

$$\frac{|-0.01610723 + 0.0161|}{|-0.01610723|} = 4.4887 \times 10^{-4}.$$

3. INTERPOLATION AND LAGRANGE POLYNOMIAL

Let a census of the population of the U.S. is taken every 10 years. The following table lists the population, in thousands of people, from 1940 to 1990.

In reviewing these data, we might ask whether they could be used to provide a reasonable estimate of the population, say in 1965, or even in the year 2010. Predictions of this type can be obtained by using a function fitting the given data. This process is called the interpolation.

The most natural choice of interpolation tool is perhaps, the algebraic polynomials,

$$(3.1) \quad P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

Oftentimes, due to its simplicity, the polynomials are used to replace any given functions. We note that if f is nice, one can always find right polynomial that approximate it well.

Theorem 3.1 (Weierstrass Approximation Theorem). *Suppose that f is defined and continuous on $[a, b]$. For each $\varepsilon > 0$, there exists a polynomial $P(x)$, with the property that*

$$|f(x) - p(x)| < \varepsilon, \quad \forall x \in [a, b].$$

n	0	1	2	3	4	5
$P_n(3)$	1	-1	3	-5	11	-21

TABLE 3.2. Taylor polynomial of $1/x$ at $x_0 = 3$

Note that for any given data (x_i, y_i) for $i = 0, 1, \dots, N$, with $x_i < x_{i+1}$, our goal is basically to find a polynomial passing through the data, i.e.,

$$(3.2) \quad P(x_i) = y_i, \quad \forall i = 0, \dots, N.$$

We can also interpret such a polynomial as follows: For a given function, $f(x)$, we can construct a polynomial that has the same values with f at specific nodes, x_i 's with $i = 0, \dots, N$. In such a case, we say that the polynomial is an interpolation of the function f at nodes x_i 's.

The Weierstrass theorem does not say anything about how to find a nice and appropriate polynomial so that it is close to the given function. For instance, we consider the Taylor polynomial at $x_0 = 0$ to approximate $f(x) = e^x$. Namely, construct the interpolation polynomial of f at x_0 by the Taylor polynomial.

The Taylor polynomials are

$$\begin{aligned} p_0(x) &= 1 \\ p_1(x) &= 1 + x \\ p_2(x) &= 1 + x + \frac{x^2}{2} \\ p_3(x) &= 1 + x + \frac{x^2}{2} + \frac{x^3}{6}. \end{aligned}$$

Although better approximations are obtained for $f(x) = e^x$, if higher-degree Taylor polynomials are used, this is not the case for all functions.

Consider $f(x) = 1/x$ expanded about $x_0 = 1$ to approximate $f(3) = 1/3$. Note that $f^{(k)}(x) = (-1)^k k! x^{-k-1}$ and the Taylor polynomials are

$$P_n(x) = \sum_{k=0}^n \frac{f^{(k)}(1)}{k!} (x-1)^k = \sum_{k=0}^n (-1)^k (x-1)^k.$$

The values of $P_n(3)$ are listed in Table, 3.2. The reason why the Taylor polynomial introduce such a dramatic error is that it is designed to approximate a function at a specific point. We are then directed to spread out the nodes at which the polynomial interpolates the given function.

3.1. Lagrange Polynomials. The Lagrange interpolation polynomial is the polynomial that is determined simply by specifying certain points on the plane through which they much pass.

More precisely, we construct a polynomial of degree at most n that passes through the $n + 1$ points.

$$(3.3) \quad (x_0, f(x_0)), \dots, (x_n, f(x_n)), \text{ with } x_i \neq x_j \text{ for } i \neq j.$$

Note first that the polynomial of degree at most n that interpolates the given $n + 1$ data can be shown to exist uniquely. To see this, we set

$$(3.4) \quad P(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n.$$

Since we wish P to interpolate $(x_i, f(x_i))$ for $i = 0, \dots, N$, we arrive at the following systems of equation:

$$(3.5) \quad \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ \vdots \\ f(x_N) \end{pmatrix}$$

It is well known that the system matrix given in the equation (3.5) is called the Vandermonde matrix which is always invertible if $x_i \neq x_j$ for $i \neq j$. Hence, it shows the unique existence of the polynomial that interpolates the data $(x_i, f(x_i))$ with $i = 0, \dots, x_N$.

Such a polynomial P is called the *Lagrange interpolant* of f at the x_i . While the proof of unique existence of a Lagrange interpolant just given was indirect, it is straightforward to derive a formula for the solution. We begin by a simple example below.

Example 3.1. *We consider to construct a polynomial passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ by a first-degree polynomial.*

First, we define linear functions $L_{1,0}$ and $L_{1,1}$ so that $L_{1,0}(x_0) = 1, L_{1,0}(x_1) = 0$ and $L_{1,1}(x_0) = 0, L_{1,1}(x_1) = 1$ as follows:

$$L_{1,0}(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{and} \quad L_{1,1}(x) = \frac{x - x_0}{x_1 - x_0}$$

and then define

$$P_1(x) = f(x_0)L_{1,0}(x) + f(x_1)L_{1,1}(x).$$

It is then easy to check that P_1 is linear and passing through the given points.

In general, to construct a polynomial passing through points (3.3), we first construct $L_{n,k}(x)$ so that it is n -th degrees of polynomials and

$$L_{n,k}(x_k) = 1, \quad L_{n,k}(x_\ell) = 0, \quad \forall \ell = 1, \dots, k-1, k+1, \dots, n$$

and then define

$$\begin{aligned} P_n(x) &= f(x_0)L_{n,0}(x) + \cdots + f(x_n)L_{n,n}(x) \\ &= \sum_{k=0}^n f(x_k)L_{n,k}(x), \end{aligned}$$

where

$$L_{n,k}(x) = \prod_{0 \leq m \leq n, m \neq k} \frac{x - x_m}{x_k - x_m}.$$

We consider how to construct $L_{n,k}(x)$ with the aforementioned properties: First of all, $L_{n,k}(x)$ contains the following numerator:

$$(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n).$$

To satisfy $L_{n,k}(x_k) = 1$, the denominator should be given so that

$$(3.6) \quad L_{n,k}(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0)(x_k - x_1) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

We call $L_{n,k}$ the k -Lagrange basis functions.

Example 3.2. Construct the quadratic polynomial that interpolates $f(x) = 1/x$ at nodes $x_0 = 2, x_1 = 2.5, x_2 = 4$. We first construct three Lagrange basis functions as follows:

$$\begin{aligned} L_0(x) &= \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \\ L_1(x) &= \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \\ L_2(x) &= \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \end{aligned}$$

We then construct $P(x) = f(x_0)L_0 + f(x_1)L_1 + f(x_2)L_2$.

3.2. Error formula for the Lagrange interpolant. We shall now consider some error analysis here. Recall that Rolle's theorem says that between any two points where the graph of the differentiable function $f(x)$ cuts the x-axis there must be a point where $f'(x) = 0$.

Lemma 3.1. If f is continuous and differentiable with two roots $f(x_1) = f(x_2) = 0$, then there exists $x_1 < c < x_2$ such that $f'(c) = 0$. Furthermore, if $f(x) = 0$ at $n+1$ points in $[a, b]$ and $f \in C^n([a, b])$, then there exists $c \in [a, b]$ such that $f^n(c) = 0$.

Theorem 3.2. Let $x_i, i = 0, 1, \dots, n$ be distinct points and let $p \in \mathcal{P}_n$ be the Lagrange interpolant of some function f at the points x_i . Let $x \in \mathbb{R}$ and suppose

$f \in C^{n+1}(I)$ for some interval I containing x and x_i 's. Then there exists a point ξ in the interior of I such that

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - x_0) \cdots (x - x_n).$$

Proof. We may assume that x differs from all the x_i 's, since the theorem is obvious otherwise. Let $\omega(x) = (x - x_0) \cdots (x - x_n)$ and set

$$G(t) = [f(x) - p(x)]\omega(t) - [f(t) - p(t)]\omega(x).$$

Then G has $n + 2$ distinct zeros, the x_i 's and x . By repeated application of Rolle's theorem, there exists ξ strictly between the largest and the smallest of the zeros such that $d^{(n+1)}G/dt^{(n+1)}(\xi) = 0$. Since $p^{(n+1)} = 0$ and $\omega^{(n+1)} = (n+1)!$, this gives

$$[f(x) - p(x)](n+1)! - f^{(n+1)}(\xi)\omega(x).$$

This completes the proof. \square

Remark 3.1. *If all the x_i tend to the same point a , then p tends to the Taylor polynomial of degree n at a , and the estimate tends to the standard remainder formula for the Taylor polynomial*

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x - a)^{n+1},$$

where ξ is between a and x .

The error bound in particular indicates that if $a \leq \min x_i$ and $b \geq \max x_i$, then

$$\|f - p\|_{\infty, [a, b]} \leq \frac{1}{(n+1)!} \|f^{(n+1)}\|_{\infty, [a, b]} |b - a|^{n+1},$$

no matter what the configuration of the point $x_i \in [a, b]$. This gives a useful estimate if we hold n fixed and let the points x_i tend to the evaluation point x , from which a rate of convergence of order $n + 1$ can be attained.

Another interesting question, closer to the approximation problem we have considered asks about the error on a fixed interval as we increase the number of interpolation points and so the degree of the interpolating polynomial,

$$\begin{aligned} & x_0^0 \\ & x_0^1 < x_1^1 \\ & x_0^2 < x_1^2 < x_2^2 \\ & \vdots \end{aligned}$$

all belonging to $[a, b]$. Then we let $p_n \in \mathcal{P}_n$ interpolate f at x_i^n and inquire about the convergence of p_n to f_n as $n \rightarrow \infty$.

Whether such convergence occurs, and how fast, depends on the properties of the function f and on the particular arrangement of points.

A famous example using equally spaced interpolation points was given by Runge: $a = -5$ and $b = 5$ with $x_i^n = -5 + 10i/n$, $f(x) = 1/(1+x^2)$. He proved the existence of a number $\kappa \approx 3.63338$ such that $\lim_{n \rightarrow \infty} p_n(x) = f(x)$ if and only if $|x| < \kappa$.

If the function is not smooth, the results may be even worse: in 1918, S. Bernstein proved that equidistant interpolation to $f(x) = |x|$ on $[-1, 1]$ does not converge at any point except $x = -1, 0, 1$. Fortunately, as we shall see in the next section, there exist much better choices of interpolation points than equally spaced points. But in 1914, Faber showed that no choice of points works for all continuous functions.

Will there be no hope? If points are well-chosen and also f is required to have a little smoothness, then convergence can be obtained.

Let us focus on one specific points and do well for this points.

Example 3.3. *Let us for now assume that f is very smooth. Increase the accuracy by adding more points. How much should we add the degree of polynomials?*

To approximate f at a specific points, we can generate polynomials successively in order to achieve a desired numerical value. However, we do not know the actual error. It is hand-waiving method. However, when doing so, we wish to use the previous information. This can be done.

4. NEWTON'S DIVIDED DIFFERENCE

Oftentimes, the iterated interpolation can be used to generate successively higher degree polynomial approximations at a specific point. The divided-difference methods can be effectively used for such a purpose.

For $x_0, x_1, x_2, \dots, x_n$ given in an interval $[a, b]$, suppose that m_0, m_2, \dots, m_k are k distinct integers, with $0 \leq m_i \leq n$ for each i . We shall denote the Lagrange polynomial that agrees with $f(x)$ at the k points x_{m_0}, \dots, x_{m_k} is denoted by $P_{m_0, m_2, \dots, m_k}(x)$.

Let f be a real-valued function on an interval $[a, b]$ and let $P_{m_0, \dots, m_k} \in \mathcal{P}_k$ be the Lagrange interpolating polynomial for f at the nodes x_{m_0}, \dots, x_{m_k} .

Our goal is to generate higher order Lagrange interpolating polynomial that P_{m_0, \dots, m_k} that interpolate $f(x)$ at an additional point m_{k+1} , say $P_{m_0, \dots, m_{k+1}}(x)$. This can be done by finding the correction function $C(x)$ that relates functions P_{m_0, \dots, m_k} and $P_{m_0, \dots, m_{k+1}}$, namely,

$$(4.1) \quad P_{m_0, \dots, m_{k+1}} = P_{m_0, \dots, m_k} + C(x).$$

It is then easy to see that $C(x)$ is given as follows:

$$C(x) = a_{m_{k+1}}(x - x_{m_0})(x - x_{m_1}) \cdots (x - x_{m_k}).$$

To find $a_{m_{k+1}}$, we use the fact that $P_{m_0, \dots, m_{k+1}}(x_{m_{k+1}}) = f(x_{m_{k+1}})$ and hence,

$$a_{m_{k+1}} = \frac{f(x_{m_{k+1}}) - P_{m_0, \dots, m_k}(x_{m_{k+1}})}{(x_{m_{k+1}} - x_{m_0})(x_{m_{k+1}} - x_{m_1}) \cdots (x_{m_{k+1}} - x_{m_k})}.$$

We denote $a_{m_{k+1}} = f[x_{m_0}, \dots, x_{m_{k+1}}]$ and call $k + 2$ -th divided difference of f relative to $x_{m_0}, \dots, x_{m_{k+1}}$.

We shall now start from the $P_{m_0}(x)$, which is a constant function, for which there is no lower degree polynomial from which it is obtained, hence, we have

$$P_{m_0} = a_{m_0} = f(x_{m_0}) = f[x_{m_0}].$$

Now to construct $P_{m_0, m_1}(x)$, we use the fact that

$$\begin{aligned} P_{m_0, m_1}(x) &= f[x_{m_0}] + f[x_{m_0}, x_{m_1}](x - x_{m_0}) \\ P_{m_0, m_1, m_2}(x) &= f[x_{m_0}] + f[x_{m_0}, x_{m_1}](x - x_{m_0}) \\ &\quad + f[x_{m_0}, x_{m_1}, x_{m_2}](x - x_{m_0})(x - x_{m_1}) \\ P_{m_0, m_1, m_2, m_3}(x) &= f[x_{m_0}] + f[x_{m_0}, x_{m_1}](x - x_{m_0}) \\ &\quad + f[x_{m_0}, x_{m_1}, x_{m_2}](x - x_{m_0})(x - x_{m_1}) \\ &\quad + f[x_{m_0}, x_{m_1}, x_{m_2}, x_{m_3}](x - x_{m_0})(x - x_{m_1})(x - x_{m_2}) \\ &\quad \vdots \end{aligned}$$

We then obtain the following for instance when $m_0 = 0, m_1 = 1, \dots, m_k = k, \dots, m_n = n$,

$$P_{0, \dots, n}(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, x_1, \dots, x_n](x - x_0) \cdots (x - x_{n-1}).$$

The natural question is now reduced to how to find the divided difference coefficients. The idea comes from the following recursive relation :

$$(4.2) \quad f[x_{m_0}, \dots, x_{m_k}] = \frac{f[x_{m_1}, \dots, x_{m_k}] - f[x_{m_0}, \dots, x_{m_{k-1}}]}{x_{m_k} - x_{m_0}}$$

Proof. Prove it. □

The recursive relation basically says that the coefficient for the Lagrange polynomial can again be computed by the data $f[x_0], \dots, f[x_n]$.

Let us assume that we are looking at the Lagrange interpolating polynomial at nodes $x_{-2}, x_{-1}, x_0, x_1, x_2$ of f . Then each divided difference can be obtained by the following:

x	0.0	0.2	0.4	0.6
$f(x)$	15.0	21.0	30.0	51.0

TABLE 4.1. Values of f

Example 4.1. Let the following data for f is given at nodes $x_0 = 1, x_1 = 2, x_2 = 3$ with $f(x_0) = 3, f(x_1) = 2$ and $f(x_2) = 1$, compute the Lagrange interpolating polynomial of f at x_0, x_1, x_2 .

Example 4.2. Use the Lagrange interpolating polynomial to approximate $f(0.3)$ based on the following data (4.1):

Example 4.3. Do the exercise number 17.

5. NUMERICAL DIFFERENTIATION AND INTEGRATION

5.1. Numerical Differentiation based on Lagrange Interpolating polynomials. The objective of this section is to approximate $f'(x)$ by using the value of f . More precisely, we consider the following problem that given nodes, x_0, x_1, \dots, x_n , we evaluate $f'(x_i)$ for some i , using $f(x_\ell)$'s with $\ell \in \{0, 1, \dots, n\}$.

In the following discussion, we shall simply assume that $x_i = x_{i-1} + h$ with $h > 0$.

We consider to approximate $f'(x_0)$ using two data $f(x_0)$ and $f(x_1)$. First construct the Lagrange polynomial $P_{0,1}(x)$ for f , with error term:

$$\begin{aligned} f(x) &= P_{0,1}(x) + \frac{(x-x_0)(x-x_1)}{2!} f''(\xi(x)) \\ &= \frac{f(x_0)(x-(x_0+h))}{(x_0-(x_0+h))} + \frac{f(x_0+h)(x-x_0)}{(x_0+h-x_0)} \\ &\quad + \frac{(x-x_0)(x-(x_0+h))}{2!} f''(\xi(x)) \end{aligned}$$

By taking derivative, we have that

$$\begin{aligned} f'(x) &= \frac{f(x_0+h) - f(x_0)}{h} + D_x \left(\frac{(x-x_0)(x-x_0-h)}{2} f''(\xi(x)) \right) \\ &= \frac{f(x_0+h) - f(x_0)}{h} + \left(\frac{2(x-x_0) - h}{2} f''(\xi(x)) \right) \\ &\quad + \frac{(x-x_0)(x-x_0-h)}{2} D_x(f''(\xi(x))). \end{aligned}$$

By plugging x_0 in x , we obtain that

$$\begin{aligned} f'(x) &= \frac{f(x_0+h) - f(x_0)}{h} + D_x \left(\frac{(x-x_0)(x-x_0-h)}{2} f''(\xi(x)) \right) \\ &= \frac{f(x_0+h) - f(x_0)}{h} + \left(\frac{-h}{2} f''(\xi(x_0)) \right). \end{aligned}$$

If h is small and $|f''(x)| \leq M$ then the difference quotient $\frac{f(x_0+h)-f(x_0)}{h}$ can be used to approximate $f'(x_0)$ and the error is given as below:

$$(5.1) \quad \left| f'(x_0) - \frac{f(x_0+h) - f(x_0)}{h} \right| \leq \frac{h}{2} M = O(h).$$

This is called the forward difference. Note that if we choose $h \rightarrow -h$, then the same argument can be made to show that

$$(5.2) \quad \left| f'(x_0) - \frac{f(x_0-h) - f(x_0)}{-h} \right| = \left| f'(x_0) - \frac{f(x_0) - f(x_0-h)}{h} \right| = O(h).$$

This is called the backward difference.

Example 5.1. Let $f(x) = \ln(x)$ and $x_0 = 1.8$. The forward-difference formula is

$$D_h^+ f(1.8) = \frac{f(1.8+h) - f(1.8)}{h} \approx f'(1.8).$$

$$\begin{aligned} h &= D_h^+ f(1.8) & f'(1.8) \\ 0.1 &= 0.5406722 & 0.555555 \dots \\ 0.01 &= 0.5540180 & 0.555555 \dots \\ 0.001 &= 0.5554013 & 0.555555 \dots \end{aligned}$$

Example 5.2. Use the forward difference or backward difference to determine the missing entries in the following:

x	$f(x)$	$f'(x)$
0.5	0.4794	
0.6	0.5646	
0.7	0.6442	

Now, if we wish to increase the accuracy, we use more higher order Lagrange interpolating polynomials (The choice of nodes in taking LIP is oftentimes crucial in reducing the error.)

Example 5.3. Suppose we approximate $f'(x_0)$ using the value $f(x_0), f(x_1)$ and $f(x_2)$, namely we shall derive three point formulas for $f'(x_0)$.

To do so, we construct the Lagrange interpolating polynomial $P_{0,1,2}$ of f at x_0, x_1 and x_2 .

$$\begin{aligned} f(x) &= P_{0,1,2}(x) + \frac{f'''(\xi(x))}{3!}(x-x_0)(x-x_1)(x-x_2) \\ &= f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \\ &+ f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} + \frac{f'''(\xi(x))}{3!}(x-x_0)(x-x_1)(x-x_2). \end{aligned}$$

We then take the derivative of f at x_0 to obtain that

$$\begin{aligned} f(x_0)' &= f(x_0)\frac{(x-x_1)+(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)+(x-x_2)}{(x_1-x_0)(x_1-x_2)} \\ &+ f(x_2)\frac{(x-x_0)+(x-x_1)}{(x_2-x_0)(x_2-x_1)} + \frac{f'''(\xi(x))}{3!}(x_0-x_1)(x_0-x_2)|_{x_0}. \end{aligned}$$

Suppose $x_1 = x_0 + h$ and $x_2 = x_0 + 2h$. Then we have

$$\begin{aligned} f(x_0)' &= f(x_0)\frac{-3h}{(-h)(-2h)} + f(x_1)\frac{-2h}{(h)(-h)} \\ &+ f(x_2)\frac{-h}{2hh} + \frac{f'''(\xi(x))}{3!}(-h)(-2h). \end{aligned}$$

Namely,

$$f(x_0)' = \frac{1}{h} \left(\frac{-3}{2}f(x_0) + 2f(x_1) - \frac{1}{2}f(x_2) \right) + \frac{h^2}{3}f'''(\xi)$$

Let us assume now that $x_1 = x_0 - h$ and $x_2 = x_0 + h$.

$$\begin{aligned} f(x_0)' &= f(x_1)\frac{-h}{(-h)(-2h)} + f(x_2)\frac{h}{(h)(2h)} + \frac{f'''(\xi)}{3!}(h)(-h) \\ &= \frac{1}{h} \left(\frac{1}{2}f(x+h) - \frac{1}{2}f(x-h) \right) - \frac{f'''(\xi)}{6}h^2. \end{aligned}$$

To obtain general derivative approximation formulas, given $n+1$ distinct points $\{x_0, \dots, x_n\}$, then we have

$$f(x) = \sum_{k=0}^n f(x_k)L_k(x) + \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}f^{(n+1)}(\xi(x)).$$

Taking derivative

$$\begin{aligned}
f(x) &= \sum_{k=0}^n f(x_k)L_k(x) + \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi(x)). \\
&= \sum_{k=0}^n f(x_k)L_k(x)' + D_x\left[\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}\right] f^{(n+1)}(\xi(x)) \\
&\quad + \frac{(x-x_0)\cdots(x-x_n)}{(n+1)!} D_x(f^{(n+1)}(\xi(x)))
\end{aligned}$$

For x_j , we evaluate f , then

$$f(x_j) = \sum_{k=0}^n f(x_k)L_k(x_j)' + D_x\left[\frac{(x-x_0)\cdots(x-x_n)}{(n+1)!}\right](x_j) f^{(n+1)}(\xi(x_j)).$$

This is $n+1$ -point formula to approximate $f(x_j)$.

5.1.1. *Numerical Differentiation is not robust.* Suppose we take numerical differentiation on a finite precision computer and so there is round-off error in evaluating f .

Then

$$\begin{aligned}
f(x_0+h) &= fl(f(x_0+h)) + E(x_0+h) \\
f(x_0-h) &= fl(f(x_0-h)) + E(x_0-h)
\end{aligned}$$

So total error will be

$$\begin{aligned}
\left| f'(x_0) - \frac{fl(f(x_0+h)) - fl(f(x_0-h))}{2h} \right| &= \frac{E(x_0+h) - E(x_0-h)}{2h} \\
&\quad - \frac{h^2}{6} f'''(\xi).
\end{aligned}$$

We then have with $E(x_0+h), E(x_0-h) \leq \varepsilon$ and $f''' < M$,

$$(5.3) \quad \text{Error} \leq \frac{\varepsilon}{h} + \frac{h^2}{6} M.$$

Reducing h decreases the approximation error $h^2/6$ however increases the round off error.

Remark 5.1. *Numerical differentiation is known as an unstable method because of this phenomenon, however, it is essential ingredient to solve the differential equation.*

5.2. Extrapolation. Extrapolation can be used whenever it is known that an approximation technique has an error term with a predictable form to enhance the accuracy.

Suppose we expand the function f in a fourth order Taylor polynomial about x_0 . Then

$$\begin{aligned} f(x) &= f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \frac{1}{6}f'''(x_0)(x - x_0)^3 \\ &+ \frac{1}{24}f^{(4)}(x_0)(x - x_0)^4 + \frac{1}{120}f^{(5)}(\xi)(x - x_0)^5, \end{aligned}$$

where ξ in between x and x_0 .

Evaluating f at $x_0 + h$ and $x_0 - h$ gives

$$\begin{aligned} f(x_0 + h) &= f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 \\ &+ \frac{1}{24}f^{(4)}(x_0)h^4 + \frac{1}{120}f^{(5)}(\xi)h^5, \end{aligned}$$

and

$$\begin{aligned} f(x_0 - h) &= f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 \\ &+ \frac{1}{24}f^{(4)}(x_0)h^4 - \frac{1}{120}f^{(5)}(\xi)h^5, \end{aligned}$$

where $x_0 - h < \xi_2 < x_0 < \xi_1 < x_0 + h$.

We can obtain that

$$\begin{aligned} f(x_0 + h) - f(x_0 - h) &= 2hf'(x_0) + \frac{h^3}{3}f'''(x_0) \\ &+ \frac{h^5}{120}[f^{(5)}(\xi_1) + f^{(5)}(\xi_2)]. \end{aligned}$$

By the Intermediate Value theorem, we have

$$\frac{1}{2}[f^{(5)}(\xi_1) + f^{(5)}(\xi_2)] = f^{(5)}(\xi).$$

Hence, we have

$$(5.4) \quad f'(x_0) = \frac{1}{2h}[f(x_0 + h) - f(x_0 - h)] - \frac{h^2}{6}f'''(x_0) - \frac{h^4}{120}f^{(5)}(\xi).$$

Let us use nodes $x_0 + 2h$ and $x_0 - 2h$ to obtain that

$$(5.5) \quad f'(x_0) = \frac{1}{4h}[f(x_0 + 2h) - f(x_0 - 2h)] - \frac{4h^2}{6}f'''(x_0) - \frac{16h^4}{120}f^{(5)}(\xi).$$

Multiplying 4 by the first $f'(x_0)$ and subtracting the second $f'(x_0)$, we obtain that

$$\begin{aligned} f'(x_0) &= \frac{1}{12h}[f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h)] \\ &+ \frac{h^4}{30}f^{(5)}(\xi). \end{aligned}$$

6. NUMERICAL INTEGRATION

In this section, we shall approximate $\int_a^b f dx$ by the following sum:

$$\sum_{i=0}^n a_i f(x_i).$$

The aforementioned sum is called the numerical quadrature. It depends on the choice of nodes since the weights a_i shall be constructed based on the interpolation polynomials.

6.1. The quadrature rule based on the interpolation polynomials. Given a set of distinct nodes $\{x_0, \dots, x_n\}$ from the interval $[a, b]$. We recall the following identity:

$$(6.1) \quad f(x) = \sum_{i=0}^n f(x_i)L_i(x) + \prod_{i=0}^n (x - x_i) \frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

We now take integration both sides to obtain:

$$(6.2) \quad \int_a^b f(x) dx = \sum_{i=0}^n f(x_i) \int_a^b L_i(x) dx + \int_a^b \prod_{i=0}^n (x - x_i) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} dx$$

The quadrature formula shall then be given as follows:

$$(6.3) \quad \int_a^b f(x) dx \approx \sum_{i=0}^n f(x_i) \int_a^b L_i(x) dx$$

Note that $\int_a^b L_i(x) dx = a_i$ is the natural weight and the error term is given by

$$(6.4) \quad E(f) = \int_a^b \prod_{i=0}^n (x - x_i) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} dx$$

For error analysis, we shall use the following *integral mean value theorem* that if $u \in C([a, b])$ and w is an integrable function on $[a, b]$ which does not change sign, then the following holds true: there exists $\eta \in (a, b)$ such that

$$(6.5) \quad \int_a^b u(x)w(x) dx = u(\eta) \int_a^b w(x) dx.$$

Example 6.1. *The Trapezoidal rule is given by choosing $x_0 = a$ and $x_1 = b$. It is two point rule. To derive the formula, we consider the linear interpolating polynomial for f given as follows:*

$$(6.6) \quad P_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1).$$

We then have

$$\begin{aligned} \int_a^b f(x) dx &= \int_{x_0}^{x_1} \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) dx \\ &+ \frac{1}{2} \int_{x_0}^{x_1} (x - x_0)(x - x_1) f^{(2)}(\xi(x)) dx \\ &= \frac{(x_1 - x_0)}{2} [f(x_0) + f(x_1)] + \frac{1}{2} \int_{x_0}^{x_1} (x - x_0)(x - x_1) f^{(2)}(\xi(x)) dx \end{aligned}$$

To investigate the error term, since $(x - x_0)(x - x_1)$ does not change its sign, we apply the integral mean value theorem to obtain that

$$E = \frac{1}{2} \int_{x_0}^{x_1} (x - x_0)(x - x_1) f^{(2)}(\xi(x)) dx = \frac{1}{2} f^{(2)}(\xi) \int_{x_0}^{x_1} (x - x_0)(x - x_1) dx.$$

By change of variables $(x - x_0)(x_1 - x_0) = t$, we have that

$$E = -(h^3/12) f''(\xi).$$

Remark 6.1. *Since the error term for the Trapezoidal rule involves f'' , the rule gives the exact result when applied to any function whose second derivative is zero, that is any polynomial of degree one or less.*

We now look at the Simpson's rule with nodes $x_0 = a, x_1 = a + (b - a)/2$ and $x_2 = b$.

To obtain Simpson's rule, we form the quadratic interpolating polynomials of f at the aforementioned nodes and then take integrations

$$(6.7) \quad \int_a^b f(x) dx = \int_a^b P_2(x) dx + \int_a^b E(x, f) dx.$$

The quadrature rule will be then as follows:

$$(6.8) \quad \int_a^b f(x) dx \approx \frac{b - a}{6} (f(a) + 4f((a + b)/2) + f(b)).$$

By construction, an interpolatory quadrature with $n + 1$ points has *degree of precision* at least n . Note that by degree of precision, we mean more precisely that

Definition 6.1. *The degree of precision or accuracy of a quadrature formula is the largest positive integer n such that the formula is exact for any polynomial of degree less than or equal to n .*

Note that the quadrature rule is of the degree of precision n , if and only if the quadrature rule is exact for x^k for $k = 0, 1, \dots, n$ (Why?). Note that the degree of precision is **independent** of the interval in which the integration is taken. Therefore, in order to find out the degree of precision for any given quadrature rule $\text{NI}_{[a,b]}$, simply take any interval $[a, b]$ and check what is the maximum k for which the following holds true :

$$(6.9) \quad \int_a^b x^k dx = \text{NI}_{[a,b]}(x^k).$$

Example 6.2. *Find the degree of precision of the Simpson's rule. To do so, we check the exactness for each polynomial, $1, x, x^2, \dots$*

$$\begin{aligned} \int_0^1 1 dx &= 1 = \frac{1}{6}(1 + 4 + 1) \\ \int_0^1 x dx &= \frac{1}{2} = \frac{1}{6}(0 + 4(1/2) + 1) \\ \int_0^1 x^2 dx &= \frac{1}{3} = \frac{1}{6}(0 + 4(1/2)^2 + 1) \\ \int_0^1 x^3 dx &= \frac{1}{4} = \frac{1}{6}(0 + 4(1/2)^3 + 1) \\ \int_0^1 x^4 dx &= \frac{1}{5} \neq \frac{1}{6}(0 + 4(1/2)^4 + 1) \end{aligned}$$

Namely, for some choices of nodes, a higher degree of precision can be achieved. The Trapezoidal and Simpson's rule are based on using the equidistant nodes. In general, a class of interpolatory quadrature rules based on using $n + 1$ equi-spaced points including end points is called the *closed Newton-Cotes rule*. For $n = 1$, the Trapezoidal rule, for $n = 2$, it is Simpson's rule. By construction, the Newton-Cotes rule has degree of precision n . But, if n is even, it has the degree of precision $n + 1$, which is due to symmetry, namely, one more higher degree of precision is like free lunch obtained by the symmetry. To be more precise, we shall show that if n is even, then $n + 1$ interpolatory rule is exact for x^{n+1} . First note that

$$\begin{aligned} \int_a^b x^{n+1} dx &= \int_a^b \left(x - \frac{a+b}{2}\right)^{n+1} + g(x) dx \\ &= \int_a^b g(x) dx, \end{aligned}$$

where $g(x) \in \mathcal{P}_n$. On the other hand, for $n + 1$ equidistant nodes, the Lagrange interpolating polynomial for x^{n+1} will read:

$$\begin{aligned} \sum_{i=0}^n \omega_i x_i^{n+1} &= \sum_{i=0}^n \omega_i \left(\left(x_i - \frac{a+b}{2} \right)^{n+1} + g(x_i) \right) \\ &= \sum_{i=0}^n \omega_i \left(x_i - \frac{a+b}{2} \right)^{n+1} + \sum_{i=0}^n \omega_i g(x_i) \end{aligned}$$

Note that

$$\sum_{i=0}^n \left(x_i - \frac{a+b}{2} \right)^{n+1} \int_a^b L_i(x) dx = 0,$$

which can be proven by observing that

$$\begin{aligned} w_0 &= w_n, & (x_0 - (a+b)/2) &= -(x_n - (a+b)/2) \\ w_1 &= w_{n-1}, & (x_1 - (a+b)/2) &= -(x_{n-1} - (a+b)/2) \\ &\vdots \\ w_{n/2} &= w_{n/2}, & (x_{n/2} - (a+b)/2) &= 0. \end{aligned}$$

and $n + 1$ is odd (more detailed proof is left for an exercise). We have that

$$(6.10) \quad \int_a^b x^{n+1} dx = \int_a^b g(x) dx = \sum_{i=0}^n g(x_i) w_i = \sum_{i=0}^n x_i^{n+1} w_i.$$

We now consider another simple quadrature rule, called the *mid-point rule*, which is defined to be

$$(6.11) \quad \int_a^b f(x) dx \approx (b-a)f((a+b)/2).$$

For the mid-point rule on $[0, 1]$, the interpolant is simply the constant value $f(1/2)$ and so we get:

$$(6.12) \quad E = \int_0^1 f(x) - f(1/2) = \int_0^1 f'(\xi(x))(x - 1/2) dx.$$

For this, we may not use the integral mean value theorem since $(x - 1/2)$ changes its sign.

A simple approach for the mid-point rule is to use Taylor's theorem. Assuming that $f \in C^2([0, 1])$, we have

$$(6.13) \quad f(x) = f(1/2) + f'(1/2)(x - 1/2) + \frac{1}{2} f''(\xi(x))(x - 1/2)^2, \quad x \in (0, 1)$$

By taking integration, we obtain that

$$\begin{aligned}\int_0^1 f(x) - f(1/2) &= \int_0^1 \frac{1}{2} f''(\xi(x))(x - 1/2)^2 dx \\ &= \frac{1}{2} f''(\eta) \int_0^1 (x - 1/2)^2 dx = \frac{1}{24} f''(\eta).\end{aligned}$$

If we apply the mid-point rule in the interval $[0, h]$, then we have the following error:

$$\begin{aligned}\int_0^h f(x) - f(h/2) &= \int_0^h \frac{1}{2} f''(\xi(x))(x - h/2)^2 dx \\ &= \frac{1}{2} f''(\eta) \int_0^h (x - h/2)^2 dx = \frac{h^3}{24} f''(\eta).\end{aligned}$$

The point is that the most of quadrature rule is not accurate if we use it for evaluating $\int_a^b f(x) dx$ with $b-a$ being large. The key idea to overcome this drawback is to divide and conquer. It is worth remarking that to enhance the accuracy, it is better to divide the domain as many as possible and then apply the quadrature rule that requires small nodes, but maximum degree of precision (Gaussian Quadrature rule).

6.2. Composite Numerical Integration. We consider to approximate $\int_a^b f(x) dx$. As indicated, we first divide the interval into n equal subintervals of size h .

Let us first check what is h in terms of a, b and n . Since $nh = b-a$, $h = (b-a)/n$. More precisely, we have

$$\begin{aligned}\int_a^b f(x) dx &= \int_a^{a+h} f(x) dx + \int_{a+h}^{a+2h} f(x) dx + \cdots + \int_{a+(n-1)h}^{a+nh} f(x) dx \\ &= \sum_{i=1}^n \int_{a+(i-1)h}^{a+ih} f(x) dx.\end{aligned}$$

The composite Trapezoidal rule is then given by applying the Trapezoidal rule for each integral defined in each subinterval.

$$\begin{aligned}\int_a^b f(x) dx &= \sum_{i=1}^n \int_{a+(i-1)h}^{a+ih} f(x) dx \\ &\approx \sum_{i=1}^n \frac{h}{2} (f(x_{i-1}) + f(x_i)),\end{aligned}$$

where $x_i = a + ih$ with $i = 0, \dots, n$.

Now, we shall take a look at the error given by the following:

$$\begin{aligned} \left| \int_a^b f(x) dx - \sum_{i=1}^n \frac{h}{2} (f(x_{i-1}) + f(x_i)) \right| &= \left| \sum_{i=1}^n -\frac{1}{12} h^3 f''(\xi_i) \right| \\ &\leq \frac{1}{12} \sum_{i=1}^n |h^3 f''(\xi_i)| \leq \frac{1}{12} M h^2 \sum_{i=1}^n h \\ &= \frac{1}{12} M h^2 (b-a). \end{aligned}$$

We shall now list several quadrature rules and their composite rules.

- left endpoint rule : $\int_a^b f \approx (b-a)f(a)$.
- right endpoint rule : $\int_a^b f \approx (b-a)f(b)$.
- midpoint rule : $\int_a^b f \approx (b-a)f((a+b)/2)$.
- trapezoidal rule : $\int_a^b f \approx \frac{b-a}{2}[f(a) + f(b)]$.
- Simpson's rule : $\int_a^b f \approx \frac{b-a}{6}[f(a) + 4f((a+b)/2) + f(b)]$

The composite version of each rule is given as follows:

- left endpoint rule : $\int_a^b f \approx \sum_{i=1}^n (x_i - x_{i-1})f(x_{i-1})$.
- right endpoint rule : $\int_a^b f \approx \sum_{i=1}^n (x_i - x_{i-1})f(x_i)$.
- midpoint rule : $\int_a^b f \approx \sum_{i=1}^n (x_i - x_{i-1})f((x_i + x_{i-1})/2)$.
- trapezoidal rule : $\int_a^b f \approx \sum_{i=1}^n \frac{x_i - x_{i-1}}{2}[f(x_{i-1}) + f(x_i)]$.
- Simpson's rule : $\int_a^b f \approx \sum_{i=1}^n \frac{x_i - x_{i-1}}{6}[f(x_{i-1}) + 4f((x_{i-1} + x_i)/2) + f(x_i)]$,

where $x_i = a + ih$ for $i = 0, \dots, n$.

Example 6.3. Use the Composite Trapezoidal rule with four equal subintervals to approximate $\int_1^2 x \sin x dx$. First of all, $4h = 2 - 1$, so $h = 1/4$. Set $x_i = 1 + i/4$ with $i = 0, 1, 2, 3, 4$.

$$\begin{aligned} \int_1^2 x \sin x dx &\approx \sum_{i=1}^4 \frac{x_i - x_{i-1}}{2} (x_{i-1} \sin(x_{i-1}) + x_i \sin(x_i)) \\ &= \sum_{i=1}^4 \frac{1 + i/4 - (1 + (i-1)/4)}{2} (x_{i-1} \sin(x_{i-1}) + x_i \sin(x_i)) \\ &= \frac{1}{8} \sum_{i=1}^4 (x_{i-1} \sin(x_{i-1}) + x_i \sin(x_i)). \end{aligned}$$

6.3. Gaussian Quadrature. In this section, we shall ask a question on the maximal possible order of precision that can be obtained by applying n point quadrature rule. We have seen that the choice of nodes is important to achieve higher degree of precision. Since the degree of precision is independent of the interval in which you take the integration and once the quadrature rule is made at a specific interval, it can be transformed to other intervals, we shall restrict our concern on the interval $[-1, 1]$.

6.3.1. *Bottom-up Approach.* We shall ask how to choose w_i 's and x_i 's so that

$$\int_{-1}^1 f dx = \sum_{i=1}^n w_i f(x_i), \quad \forall f \in \mathcal{P}_{MAX}.$$

To do so, by linearity of the integral operator and summation operator, we may need to make sure

$$(6.14) \quad \int_{-1}^1 x^k dx = \sum_{i=1}^n w_i x_i^k, \quad \forall k = 0, \dots, MAX.$$

First of all, we note that the maximum degree of precision that can be obtained with such w_i 's and x_i 's are $2n - 1$ since we have $2n$ degrees of freedoms. Namely, one can possibly solve $2n$ equations for $k = 0, 1, \dots, 2n - 1$. For more equations to be satisfied, the problem is over-determined.

Example 6.4. Suppose that we want to determine w_1, w_2, x_1 and x_2 so that the integration formula

$$\int_{-1}^1 f(x) dx \approx w_1 f(x_1) + w_2 f(x_2)$$

gives the exact result for $f \in \mathcal{P}_3$. Note that $2 \times 2 - 1 = 3$. We then form the following set of equations:

$$\begin{aligned} \int_{-1}^1 1 dx &= 2 = w_1 + w_2 \\ \int_{-1}^1 x dx &= 0 = w_1 x_1 + w_2 x_2 \\ \int_{-1}^1 x^2 dx &= \frac{2}{3} = w_1 x_1^2 + w_2 x_2^2 \\ \int_{-1}^1 x^3 dx &= 0 = w_1 x_1^3 + w_2 x_2^3 \end{aligned}$$

A little algebra shows that this system has the unique solution

$$w_1 = w_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3}, x_2 = \frac{\sqrt{3}}{3}.$$

This is called the 2-point Gauss rule.

For the arbitrary interval $[a, b]$, the 2-point Gauss rule reads and in fact, it is the interpolatory quadrature rules at x_1 and x_2 . (Note that $L_1 = (x - x_1)(x_2 - x_1)$ and $\int_{-1}^1 L_1 = 1$. Same is true for L_2 .)

$$(6.15) \quad \int_a^b f \approx \frac{(b-a)}{2} \left(f \left(\frac{(a+b)}{2} - \frac{(b-a)}{2\sqrt{3}} \right) + f \left(\frac{(a+b)}{2} + \frac{(b-a)}{2\sqrt{3}} \right) \right).$$

It is highly non-linear equations and very difficult to find w_i and x_i for which the maximum degree of precision can be achieved. We would now ask a question if there is a simpler way to find the quadrature rule of the maximum degree of precision? In fact, there is a systematic way to find n -points in the interval $[-1, 1]$ from which we can obtain the interpolatory quadrature rule which has the maximum degree of precision.

To motivate the readers, the nodes x_i can be given by n -points of roots of some well-known polynomials called the Legendre's polynomials. The weights w_i are given by integrating the Lagrange basis that corresponds to node x_i 's.

6.3.2. *Top-down approach.* Our observation from the above simple example is the quadrature rule of the maximum possible degrees of freedom can be obtained by an interpolatory quadrature rule with special nodes, which is the roots of Legendre polynomials. Namely, n -point rule with $2n - 1$ degree of precision can be achieved by choosing n -points as roots of n -degree Legendre polynomial and constructing the interpolatory quadrature rule based on such n -points.

Our top-down approach is based on this observation. In order to construct n -point quadrature rule of the maximum degree of precision. We simply need to find the roots of the Legendre polynomial of degree n and to construct the interpolatory quadrature rule, which is called the n -point Gauss rule. It will then be crucial to know what is the Legendre polynomials, whose studies shall show our observation is in fact true and no other n -point quadrature rule can beat the n -point Gauss rule. We shall first recall the well-known class of polynomials called the Legendre Polynomials. We consider the following function space

$$(6.16) \quad \mathcal{V} = \left\{ f : \int_{-1}^1 f^2 dx < \infty. \right\}.$$

We now introduce an inner product for the space \mathcal{V} . The inner product $(\cdot, \cdot) : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$ is defined to be as follows:

$$(6.17) \quad (f, g) = \int_{-1}^1 fg \, dx, \quad \forall f, g \in \mathcal{V}.$$

We now observe that $\mathcal{P}_n([-1, 1]) \subset \mathcal{V}$ and that $\{1, x, x^2, \dots, x^n\}$ is the basis for $\mathcal{P}_n([-1, 1])$. We wish now to construct $\{q_0, q_1, \dots, q_n\} \subset \mathcal{P}_n([-1, 1])$ such that

$$(6.18) \quad (q_i, q_j) = 0, \quad i \neq j.$$

In that case, we say that q_i is orthogonal to q_j and denote $q_i \perp q_j$. In fact, we shall construct q_j in a way that $q_j = x^j + \text{lower order term} \in \mathcal{P}_j$. In such a case, the relation (6.18) tells that $q_j \perp \mathcal{P}_{j-1}$.

These can be achieved by the well-known Gram-Schmidt process. More precisely, we start from $q_0 = 1$ and

$$\begin{aligned} q_0(x) &= 1, \\ q_1(x) &= x, \\ q_2(x) &= xq_{n-1}(x) - \frac{(xq_{n-1}, q_{n-2})}{\|q_{n-2}\|^2} q_{n-2}(x), \quad n \geq 2. \end{aligned}$$

Note that it is easy to check that these polynomials are monic and mutually orthogonal. These polynomials are called the Legendre polynomials.

$$\begin{aligned} q_0(x) &= 1, \\ q_1(x) &= x, \\ q_2(x) &= x^2 - \frac{1}{3}, \\ q_3(x) &= x^3 - \frac{3x}{5}. \end{aligned}$$

By linear scalings, we can obtain the Legendre polynomials on an arbitrary interval. We normalized the Legendre polynomials by taking their leading coefficient as 1. More commonly, the Legendre polynomials are normalized to have value 1 at 1, which shall be denoted by p_k with $k = 0, \dots, n$. Then the following recursion can be shown to hold true:

$$(n+1)p_{n+1}(x) = (2n+1)xp_n(x) - np_{n-1}(x),$$

starting with $p_0 = 1, p_1(x) = x$.

We list some properties of the Legendre's polynomials.

- (1) $p_n(1) = 1, p_n(-1) = (-1)^n$.
- (2) more...

(3) Let $x_1 < x_2 < \dots < x_n$ denote the roots of P_n , then these are distinct and belong to $[-1, 1]$.

We shall now show that the maximum possible degree of precision $2n - 1$ can be achieved by a unique n -point rule. We may restrict to the interval $[a, b] = [-1, 1]$. Our chief tool will be the Legendre polynomials $p_n(x)$. Let $x_1 < x_2 < \dots < x_n$ denote the roots of p_n , then these are distinct and belong to $[-1, 1]$. These are called the n Gauss points on $[-1, 1]$.

First, we define a quadrature rule (the n -point Gauss rule) by

$$(6.19) \quad \int_{-1}^1 f \approx \int_{-1}^1 I_n f,$$

where $I_n f \in \mathcal{P}_{n-1}$ is the Lagrange interpolating polynomial of f at x_1, \dots, x_n .

Theorem 6.1. *The n -point Gauss rule has degree of precision $2n - 1$.*

Proof. Given $f \in \mathcal{P}_{2n-1}$, we can write

$$f(x) = h(x)p_n(x) + r(x), \quad h \in \mathcal{P}_{n-1}, r \in \mathcal{P}_{n-1}.$$

Then

$$\int_{-1}^1 f dx = \int_{-1}^1 q(x)p_n(x) + r(x) dx = \int_{-1}^1 r(x) dx.$$

We now note that any n -point rule has degree of precision $n - 1$, hence,

$$\int_{-1}^1 r(x) dx = \sum_{i=1}^n w_i r(x_i).$$

However, since x_i 's are root for p_n , we have that

$$\int_{-1}^1 r(x) dx = \sum_{i=1}^n w_i f(x_i) = \int_{-1}^1 f(x) dx.$$

□

Theorem 6.1. *The n -point Gauss rule is the unique n -point rule with the degree of precision $2n - 1$.*

Proof. We suppose that there are n -points for which

$$\sum_{i=1}^n w_i f(x_i) = \int_{-1}^1 f dx, \quad \forall f \in \mathcal{P}_{2n-1}$$

Then if we choose $f(x) = q(x)\prod_{i=1}^n (x - x_i)$ with $q \in \mathcal{P}_{n-1}$, the following holds true that

$$\int_{-1}^1 q(x)\prod_{i=1}^n (x - x_i) dx = 0, \quad \forall q \in \mathcal{P}_{n-1}.$$

This implies that $\prod_{i=1}^n (x - x_i)$ is a multiple of p_n . Hence x_i should be n - Gauss points. \square

Note that there is no quadrature rule of the form $\sum_{i=1}^n w_i f(x_i)$ with degree of precision $2n$. We assume that there exists a n -point quadrature rule with $2n$ degree of precision. Then

$$(6.20) \quad \int_{-1}^1 f dx = \sum_{i=1}^n f(x_i)w_i, \quad \forall f \in \mathcal{P}_{2n}.$$

However, it can not be true since for $f = \prod_{i=1}^n (x - x_i)^2 \in \mathcal{P}_{2n}$, we are led to a contradiction since

$$0 \neq \int_{-1}^1 \prod_{i=1}^n (x - x_i)^2 dx \neq \sum_{i=1}^n w_i (x_i - x_i)^2 = 0.$$

Example 6.5. Construct two points Gauss quadrature rule on $[-1, 1]$. Since $p_2(x) = (3x^2 - 1)/2$, by solving $p_2(x) = 0$, we obtain that $x_1 = -\frac{3}{\sqrt{3}}$ and $x_2 = \frac{3}{\sqrt{3}}$.

Now with these two points, we construct the Lagrange interpolating polynomial

$$I_2 f = f(x_1)L_1 + f(x_2)L_2$$

and by taking an integration, we obtain that

$$\int_{-1}^1 f(x) dx \approx f(x_1) + f(x_2).$$

Example 6.6. Construct 3-point Gauss point rule on $[-1, 1]$. Since $q_3 = x^3 - 3x/5$, the roots are given by $x_1 = -\frac{\sqrt{3}}{\sqrt{5}}$, $x_2 = 0$ and $x_3 = \frac{\sqrt{3}}{\sqrt{5}}$. Now with these two points, we construct the Lagrange interpolating polynomial

$$I_2 f = f(x_1)L_1 + f(x_2)L_2 + f(x_3)L_3$$

and by taking an integration, we obtain the three point Gauss-rule, which is left as an exercise.

$$\int_{-1}^1 f(x) dx \approx w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3).$$

Note that with $x_1 = -\sqrt{3}/\sqrt{5}$, $x_2 = 0$ and $x_3 = \sqrt{3}/\sqrt{5}$, we have

$$\int_{-1}^1 L_1 dx = \int_{-1}^1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} dx = \int_{-1}^1 \frac{6}{5} x \left(x - \frac{\sqrt{3}}{\sqrt{5}} \right) dx = \frac{5}{9}$$

$$\int_{-1}^1 L_2 dx = \int_{-1}^1 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} dx = \int_{-1}^1 -\frac{5}{3} \left(x^2 - \frac{3}{5} \right) dx = \frac{8}{9}$$

$$\int_{-1}^1 L_3 dx = \int_{-1}^1 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} dx = \int_{-1}^1 \frac{6}{5} x \left(x + \frac{\sqrt{3}}{\sqrt{5}} \right) dx = \frac{5}{9}$$

7. DIRECT METHODS ON NUMERICAL LINEAR ALGEBRA

We consider to solve the following linear system of equation:

$$(7.1) \quad Ax = b,$$

where $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$.

For an example, we can consider the following problem:

$$\begin{aligned} x_1 - x_2 + 2x_3 - x_4 &= -8 \\ 2x_1 - 2x_2 + 3x_3 - 3x_4 &= -20 \\ x_1 + x_2 + x_3 &= -2 \\ x_1 - x_2 + 4x_3 + 3x_4 &= 4 \end{aligned}$$

This can be rewritten as follows:

$$\begin{pmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 3 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -8 \\ -20 \\ -2 \\ 4 \end{pmatrix}.$$

Definition 7.1. An $n \times m$ (n by m) matrix is a rectangular array of elements with n rows and m columns.

We oftentimes use the notation $A = (a_{ij})_{1 \leq i \leq n, 1 \leq j \leq m}$, which denotes the matrix A of the following:

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \cdots & \cdots \\ a_{n1} & \cdots & \cdots & a_{nm} \end{pmatrix}$$

Example 7.1.

$$A = \begin{pmatrix} 2 & -1 & 7 \\ 3 & 1 & 0 \end{pmatrix}$$

is 2×3 matrix with $a_{11} = 2$, $a_{12} = -1$, $a_{13} = 7$, $a_{21} = 3$, $a_{22} = 1$ and $a_{23} = 0$.

We are interested in solving the following system of equations:

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & \cdots & \cdots & a_{nm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

This corresponds to the set of equations

$$E_i : a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{im}x_m = b_m.$$

In solving the aforementioned linear system of equations, the following three operations produce equivalent systems.

- Equation E_i can be replaced by λE_i for $\lambda \neq 0$. $(\lambda E_i) \rightarrow E_i$
- Equation E_j can be multiplied by any constant λ and added to equation E_i with the resulting equation used in place of E_i . $(\lambda E_j + E_i) \rightarrow E_i$.
- Equations E_i and E_j can be transposed in order. $(E_i \leftrightarrow E_j)$.

Example 7.2. *Give an example here.*

This process can be understood as a matrix form by constructing the augmented matrix:

$$\begin{pmatrix} 1 & 1 & 0 & 3 & : & 4 \\ 2 & 1 & -1 & 1 & : & 1 \\ 3 & -1 & -1 & 2 & : & -3 \\ -1 & 2 & 3 & -1 & : & 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 & 3 & : & 4 \\ 0 & -1 & -1 & -5 & : & -7 \\ 0 & -4 & -1 & -7 & : & -15 \\ 0 & 3 & 3 & 2 & : & 8 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 1 & 0 & 3 & : & 4 \\ 0 & -1 & -1 & -5 & : & -7 \\ 0 & 0 & 3 & 13 & : & 13 \\ 0 & 0 & 0 & -13 & : & -13 \end{pmatrix}$$

Observe that we use a_{11} to eliminate $a_{21}, a_{31}, \dots, a_{n1}$ and from the new matrix, using a_{22} , we eliminate $a_{32}, a_{42}, \dots, a_{n2}$, and so on. This process is called the Gaussian elimination with backward substitution.

We now consider the following linear system:

Example 7.3.

$$\begin{pmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 3 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -8 \\ -20 \\ -2 \\ 4 \end{pmatrix}.$$

We form the augmented matrix of the following form:

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 2 & -2 & 3 & -3 & -20 \\ 1 & 1 & 1 & 0 & -2 \\ 1 & -1 & 4 & 3 & 4 \end{array} \right)$$

By $(E_2 - 2E_1 \rightarrow E_2, E_3 - E_1 \rightarrow E_3, E_4 - E_1 \rightarrow E_4)$, we obtain

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & 2 & 4 & 12 \end{array} \right)$$

We are in trouble since we have $a_{22} = 0$, which is called the pivot element. But, it is o.k. since we can apply $E_2 \leftrightarrow E_3$ to obtain that

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 2 & 4 & 12 \end{array} \right)$$

Apply $E_4 + 2E_3 \rightarrow E_4$ to obtain

$$\left(\begin{array}{cccc|c} 1 & -1 & 2 & -1 & -8 \\ 0 & 2 & -1 & 1 & 6 \\ 0 & 0 & -1 & -1 & -4 \\ 0 & 0 & 0 & 2 & 4 \end{array} \right)$$

From this, we have $x_4 = 2, x_3 = 2, x_2 = 3$ and $x_1 = -7$.

7.1. LU decomposition. Consider the following augmented system of equation:

$$\begin{aligned} [A : b] &\Leftrightarrow \left(\begin{array}{ccc|c} a_{11}^{(0)} & a_{12}^{(0)} & a_{13}^{(0)} & b_1^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & a_{23}^{(0)} & b_2^{(0)} \\ a_{31}^{(0)} & a_{32}^{(0)} & a_{33}^{(0)} & b_3^{(0)} \end{array} \right) \\ &\Leftrightarrow \left(\begin{array}{ccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ 0 & a_{32}^{(1)} & a_{33}^{(1)} & b_3^{(1)} \end{array} \right) \text{ by } E_2 - (a_{21}^{(0)}/a_{11}^{(0)})E_1 \rightarrow E_2, E_3 - (a_{31}^{(0)}/a_{11}^{(0)})E_1 \rightarrow E_3 \\ &\Leftrightarrow \left(\begin{array}{ccc|c} a_{11}^{(2)} & a_{12}^{(2)} & a_{13}^{(2)} & b_1^{(2)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(2)} & b_3^{(2)} \end{array} \right) \text{ by } E_3 - (a_{32}^{(1)}/a_{22}^{(1)})E_2 \rightarrow E_3 \end{aligned}$$

We call $m_{21} = a_{21}^{(0)}/a_{11}^{(0)}, m_{31} = a_{31}^{(0)}/a_{11}^{(0)}$ and $m_{32} = a_{32}^{(1)}/a_{22}^{(1)}$ multipliers.

In case the Gaussian elimination has been performed (namely, the pivot elements are nonzero) on the linear system $Ax = b$ without row interchanges, the matrix A

can be factored into the product of a lower-triangular matrix L and an upper-triangular matrix U , so that

$$A = LU,$$

where U is the triangular matrix generated by the Gaussian elimination and $L = (L_{ij})$ with $L_{ij} = m_{ij}$ for $i \neq j$ and $L_{ii} = 1$.

$$\begin{aligned} [A : b] &\Leftrightarrow \left(\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 2 & 2 & 3 & 3 \\ -1 & -3 & 0 & 2 \end{array} \right) \\ &\Leftrightarrow \left(\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & -1 & 1 & 2 \end{array} \right) \quad \text{by } E_2 - 2E_1 \rightarrow E_2, E_3 - (-1)E_1 \rightarrow E_3 \\ &\Leftrightarrow \left(\begin{array}{ccc|c} 1 & 2 & 1 & 0 \\ 0 & -2 & 1 & 3 \\ 0 & 0 & 1/2 & 1/2 \end{array} \right) \quad \text{by } E_3 - (1/2)E_2 \rightarrow E_3 \end{aligned}$$

We can check that

$$\begin{aligned} A &= \begin{pmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & -3 & 0 \end{pmatrix} \\ &= LU \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & 1/2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 1 \\ 0 & -2 & 1 \\ 0 & 0 & 1/2 \end{pmatrix}. \end{aligned}$$

Let us assume that we have factored A into LU , then how can we use this factoring to solve the system of equation:

To solve $Ax = b$ or $LUx = b$, we first introduce the substitution $y = Ux$. Then, we have $Ly = b$, having solved the equation $Ly = b$ for y , we can solve x by solving $y = Ux$.

8. INITIAL VALUE PROBLEMS

The general form of an IVP that we shall attack is

$$\begin{aligned} y' &= f(t, y), \quad a \leq t \leq b \\ y(a) &= y_0. \end{aligned}$$

Theorem 8.1 (Well-Posedness). *Let $f(t, y)$ be continuous for all (t, y) in $D = I \times \mathbb{R}$ with $I = [a, b]$ and furthermore, there exists a constant L such that for all (t, y_1) and (t, y_2) in $I \times \mathbb{R}$,*

$$|f(t, y_2) - f(t, y_1)| \leq L|y_2 - y_1|.$$

Then the following holds true

- (1) For any $y_0 \in \mathbb{R}$, there exists a unique solution $y(t)$ throughout the interval $I = [a, b]$ for the IVP. Moreover, the solution is differentiable.
- (2) The solution y depends continuously on the initial data, namely, if \hat{y} is the solution to

$$\begin{aligned}\hat{y}' &= f(t, \hat{y}), & a \leq t \leq b \\ \hat{y}(a) &= \hat{y}_0.\end{aligned}$$

Then

$$|y(t) - \hat{y}(t)| \leq e^{Lt}|y(0) - \hat{y}(0)|.$$

The solution is not so sensitive with respect to the initial data.

- (3) Let \hat{y} satisfies, more generally, a perturbed ODE, namely

$$\begin{aligned}\hat{y}' &= f(t, \hat{y}) + r(t, \hat{y}), & a \leq t \leq b \\ \hat{y}(a) &= \hat{y}_0.\end{aligned}$$

Moreover, r is bounded on $I \times \mathbb{R}$, more specifically, $\|r\| \leq M$, then the following holds true :

$$|y(t) - \hat{y}(t)| \leq e^{Lt}|y(0) - \hat{y}(0)| + \frac{M}{L}(e^{Lt} - 1).$$

Proof. We shall omit the proof here. But notice that the most crucial factor is the Lipschitz constant of the function f in the second variables. \square

We shall now discuss the Lipschitz condition in more details.

Definition 8.1. A function $f(t, y)$ is said to satisfy a Lipschitz condition in the variable y on a set $D \subset \mathbb{R}^2$ if a constant $L > 0$ exists with

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|,$$

whenever $(t, y_1), (t, y_2) \in D$. The constant L is called a Lipschitz constant of f .

Example 8.1. Let $D = \{(t, y) : 1 \leq t \leq 2, -3 \leq y \leq 4\}$ and $f(t, y) = t|y|$, then we have

$$|f(t, y_1) - f(t, y_2)| = |t|y_1| - t|y_2|| = |t|||y_1| - |y_2|| \leq 2|y_1 - y_2|.$$

Note that if $f \in C^1(I \times \mathbb{R})$ and $\partial f / \partial y$ is bounded, then f satisfies a Lipschitz condition on $I \times \mathbb{R}$ in the Theorem 8.1. (Why?)

Example 8.2. Consider the initial value problem

$$y' = 1 + t \sin(ty), \quad 0 \leq t \leq 2, \quad y(0) = 0.$$

Is it well-posed? To answer the question, we check the Lipschitz condition for f .

$$\begin{aligned} |f(t, y_1) - f(t, y_2)| &= |(1 + t \sin(ty_1)) - (1 + t \sin(ty_2))| \\ &= |t(\sin(ty_1) - \sin(ty_2))| = |t| |\sin(ty_1) - \sin(ty_2)| \\ &= t^2 \cos(t\xi) |y_1 - y_2| \leq 4|y_1 - y_2|. \end{aligned}$$

Example 8.3. Consider the initial value problem

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \quad y(0) = 0.5$$

Is it well-posed? To answer the question, we check the Lipschitz condition for f .

$$\frac{\partial f}{\partial y} = 1.$$

Notice that many simple, smooth functions f on $I \times \mathbb{R}$ such as $f(t, y) = ty^2$ or $f(t, y) = y^2$ fail to satisfy a uniform Lipschitz condition with respect to its second variable since $\partial f / \partial y$ is unbounded. For such functions, we can not assert the global existence and uniqueness.

8.1. Some Preliminary. Given an interval $I = [a, b]$ in which the IVP is well-posed. We first decide the times at which approximations to $y(t)$ are to be made, namely,

$$a = t_0, \dots, t_N = b.$$

These sets of times $\{t_0, t_1, \dots, t_N\}$ are called mesh points. Note that once the approximate solution is obtained at the points, the approximation at other points are obtained by interpolation.

Assume that the mesh points are equidistant with distance h , then

$$t_i = a + ih, \quad \text{with } i = 0, 1, \dots, N.$$

Note that since $t_N = b = a + Nh$, h is $(b - a)/N$.

We shall use y_n by the numerical approximation of $y(t)$ at the mesh point t_n .

8.2. Derivation of Euler's Method. In this subsection, we shall discuss several derivation of Euler's method.

8.2.1. Geometric Derivation : A solution to

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = y_0.$$

can be considered to a function whose graph passes through (t_0, y_0) and at each point, is tangent to the line segment at that point determined by the slope field by f .

Start its graph at the initial point, extending it in the direction of, say, increasing t , along the line through that point with slope $f(t_0, y_0)$. This determines an approximate solution on a short time interval $[t_0, t_0 + h]$ as

$$y^h(t) = y_0 + hf(t_0, y_0), \quad t_0 \leq t \leq t_0 + h.$$

If h is sufficiently small, this should not differ much from the true solution $y(t)$ since a curve does not differ much from its tangent in a small interval.

We may then repeat the process starting from $t_1 := t_0 + h$ and using the slope at (t_1, y_1) where $y_1 = y^h(t_1) = y_0 + hf(t_0, y_0)$.

Note that it can not be generalized to give other numerical methods.

8.2.2. *Taylor's Expansion.* Note that the exact solution satisfies

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + O(h^2).$$

Neglecting the $O(h^2)$ term, we get

$$y(t_{n+1}) \approx y(t_n) + hy'(t_n) = y(t_n) + hf(t, y(t_n)).$$

This suggests the method

$$y^h(t_{n+1}) = y(t_n) + hy'(t_n) = y(t_n) + hf(t, y(t_n)).$$

or

$$y_{n+1} = y_n + hf(t_n, y_n).$$

8.2.3. *Numerical Differentiation.* Approximating the derivative $y'(t_n)$ by the forward difference $[y(t_{n+1}) - y(t_n)]/h$ leads to

$$\frac{y(t_{n+1}) - y(t_n)}{h} \approx f(t_n, y(t_n)),$$

or

$$y(t_{n+1}) \approx y(t_n) + hy'(t_n) = y(t_n) + hf(t, y(t_n)).$$

Interchange $y(t_n)$ and y_n , we obtain that

$$y_{n+1} = y_n + hf(t_n, y_n).$$

8.2.4. *Numerical Integration.* Note that the exact solution satisfies the integral condition

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t)) dt.$$

Approximating the integral by the left endpoint rule, we get again

$$y(t_{n+1}) \approx y(t_n) + hf(t_n, y(t_n)).$$

Interchange $y(t_n)$ and y_n , we obtain that

$$y_{n+1} = y_n + hf(t_n, y_n).$$

8.2.5. *Implementation of Euler's method.* We consider to approximate the solution to the initial-value problem

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \quad y(0) = 0.5$$

with $N = 10$. Note that $h = 2/10 = 0.2$ and $t_i = 0.2i$.

Algorithm 8.1. Set $a = 0, b = 2, N = 10$ and $h = (b - a)/N$;

- Set mesh points $t(n + 1) = a + nh$ for $n = 0, 1, \dots, N - 1$
- Set initial condition $y_1 = y(1) = 0.5$
- Compute the approximate values y_{n+1} for $n = 1, \dots, N - 1$:

$$\begin{aligned} y_{n+1} &= y(n + 1) = y(n) + hf(t(n), y(n)) \\ &= y(n) + h(y(n) - t(n)^2 + 1) \end{aligned}$$

for $n = 0, 1, \dots, N - 1$.

8.2.6. *Numerical Error Analysis.* In this section, we discuss how much the approximate solution obtained by Euler method differs from the exact solution.

We assume that $f \in C(I \times \mathbb{R})$, $I = [a, b]$, and satisfies a uniform Lipschitz condition with respect to its second variable. In order to indicate the numerical approximate solution is dependent on the mesh size h , we use the following notation that for any $h > 0$, define $N = N^h = (b - a)/h$ so that t_N is the largest break point in I and define $y_n = y^h(t_n)$ for $0 \leq n \leq N$ by Euler's method. The error is

$$e^h = y^h - y.$$

We shall measure the error in the discrete max norm

$$\|e^h\|_{\infty, h} = \max_{0 \leq n \leq N^h} |e^h(t_n)|$$

Definition 8.2 (Local Truncation Error of Euler's method). *Euler's method is given as follows :*

$$\begin{aligned} y_n &= \alpha \\ y_{n+1} &= y_n + hf(t_n, y_n), \quad \forall n = 0, \dots, N - 1. \end{aligned}$$

The local truncation error is defined by the following :

$$\tau_{n+1}(h) = \frac{y(t_{n+1}) - (y(t_n) + hf(t_n, y(t_n)))}{h}$$

Note that

$$\begin{aligned}\tau_{n+1}(h) &= \frac{y(t_{n+1}) - (y(t_n) + hf(t_n, y(t_n)))}{h} \\ &= \frac{y(t_{n+1}) - y(t_n)}{h} - y'(t_n) \\ &= \frac{h}{2}y''(\xi_n) = O(h) \quad \xi_n \in (t_n, t_{n+1}).\end{aligned}$$

It is local relative error of the Euler's method. Namely, we compare the exact solution and approximate solution of the Euler's method starting with the exact solution at the beginning step.

Lemma 8.1. Let $A, B, \eta_0, \dots, \eta_N$ be non-negative numbers satisfying

$$\eta_{n+1} \leq A\eta_n + B, \quad n = 0, 1, \dots, N-1.$$

Then

$$\eta_n \leq A^n \eta_0 + \left(\sum_{i=0}^{n-1} A^i \right) B, \quad n = 0, \dots, N.$$

For $A \neq 1$, then the quantity in parenthesis is equal to $(A^n - 1)/(A - 1)$.

Theorem 8.2 (Convergence of Euler's method). Assume that $|y''| \leq M$, then

$$\lim_{h \rightarrow 0} \|y^h - y\|_{\infty, h} = 0.$$

and moreover,

$$\|y^h(t_n) - y(t_n)\| \leq \frac{hM}{2L} |e^{L(t_n - a)} - 1|$$

Proof. Define the local error at $(n+1)$ st step by the equation

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + T_n.$$

Since Euler's methods read

$$y_{n+1} = y_n + hf(t_n, y_n)$$

we have the following error equation

$$e_{n+1} = e_n + h[f(t_n, y_n) - f(t_n, y(t_n))] - T_n.$$

Setting $T = \max_{0 \leq n \leq N-1} |T_n|$ and using the Lipschitz condition, we have

$$|e_{n+1}| \leq (1 + hL)|e_n| + T, \quad 0 \leq n \leq N-1,$$

Since we start with the exact initial value, $e_0 = 0$.

Applying the lemma, we get

$$|e_n| \leq \frac{(1 + hL)^n - 1}{hL} T, \quad n = 0, 1, \dots, N.$$

Now since $1 + x \leq e^x$, we have

$$|e_n| \leq \frac{e^{L|t_n - a|} - 1}{L} \frac{T}{h},$$

and now, it is clear that

$$\frac{T}{h} \leq \frac{h}{2} M.$$

This completes the proof. \square

Remark 8.1. *The error bound is often very pessimistic compared to the outcome of actual computation, but the first order convergence is not. This is the principal importance of the theorem given above. Consequently, diminishing the step size should give correspondingly greater accuracy to the approximations.*

Example 8.4. *Consider the IVP given by the following :*

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \quad y(0) = 0.5$$

Note that

$$y''(t) = f' = \frac{\partial f}{\partial t}(t, y(t)) + \frac{\partial f}{\partial y}(t, y(t)) \cdot f(t, y(t)).$$

For this problem, the exact solution is $y(t) = (t+1)^2 - \frac{1}{2}e^t$. so $y''(t) = 2 - 0.5e^t$. and

$$|y''(t)| \leq 0.5e^2 - 2.$$

Let us choose $h = 0.2$ and $L = 1$ and $M = 0.5e^2 - 2$. We then have

$$|e_n| \leq 0.1 * (0.5e^2 - 2)(e^{t_n} - 1).$$

8.3. Higher Order Taylor Methods. All single step methods may be written in the form:

$$(8.1) \quad y_{n+1} = y_n + h\Phi(f; t_n, y_n, h)$$

and the local truncation error is defined to be

$$(8.2) \quad \tau_{n+1} = \frac{y(t_{n+1}) - [y(t_n) + h\Phi(f; t_n, y(t_n), h)]}{h}.$$

If $\tau_{n+1} = O(h^p)$, then we say that the method is of order p . Recall that the Euler's method is of order one.

Suppose the solution $y(t)$ to the initial-value problem

$$y' = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha,$$

has $n + 1$ continuous derivatives. The Taylor series expansion about $t = t_n$ leads to

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + hy'(t_n) \\ &+ \frac{h^2}{2}y''(t_n) + \cdots + \frac{h^n}{n!}y^{(n)}(t_n) + \frac{h^{n+1}}{(n+1)!}y^{(n+1)}(\xi_n), \end{aligned}$$

where $\xi_n \in (t_n, t_{n+1})$.

Since $y^{(n)}(t) = f^{(n-1)}(t, y(t))$, we have the following relation

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}f'(t_n, y(t_n)) \\ &+ \cdots + \frac{h^n}{n!}f^{(n-1)}(t_n, y(t_n)) + \frac{h^{n+1}}{(n+1)!}f^{(n)}(\xi_n, y(\xi_n)), \end{aligned}$$

From this, we derive the Taylor method of order n as follows:

$$\begin{aligned} y_0 &= \alpha \\ y_{n+1} &= y_n + h\phi(t_n, y_n), \quad \forall n = 0, 1, \dots, \end{aligned}$$

where

$$\phi(t_n, y_n) = f(t_n, y_n) + \frac{h}{2}f'(t_n, y_n) + \cdots + \frac{h^{n-1}}{n!}f^{(n-1)}(t_n, y_n)$$

Theorem 8.3. Assume that the solution $y(t)$ to

$$y'(t) = f(t, y(t)), \quad a \leq t \leq b, \quad y(a) = \alpha,$$

belongs to $C^{n+1}[a, b]$ and the Taylor's method of order n is used to approximate the solution with step size h , then the local truncation error is $O(h^n)$.

Example 8.5. Apply Taylor's method of orders two and four to

$$y' = y - t^2 + 1, \quad 0 \leq t \leq 2, \quad y(0) = 0.5.$$

Note that

$$\begin{aligned} f(t, y(t)) &= y - t^2 + 1 \\ f'(t, y(t)) &= y' - 2t = y - t^2 + 1 - 2t \\ f''(t, y(t)) &= y' - 2t - 2 = y - t^2 - 2t - 1 \\ f'''(t, y(t)) &= y - t^2 - 2t - 1. \end{aligned}$$

Hence

$$\begin{aligned}
 T^{(2)}(t_n, y_n^h) &= f(t_n, y_n^h) + \frac{h}{2} f'(t_n, y_n^h) \\
 &= \left(1 + \frac{h}{2}\right) (y_n^h - t_n^2 + 1) - ht_n \\
 T^{(4)}(t_n, y_n^h) &= \left(1 + \frac{h}{2} + \frac{h^2}{6} + \frac{h^3}{24}\right) (y_n^h - t_n^2) \\
 &\quad - \left(1 + \frac{h}{3} + \frac{h^2}{12}\right) ht_n + 1 + \frac{h}{2} - \frac{h^2}{6} - \frac{h^3}{24}.
 \end{aligned}$$

Remark 8.2. Note that Taylor method has been derived by taking Taylor expansion $y(t)$ around t_n and evaluate it at t_{n+1} . The explicit implementation could be done by using the relation $y'(t) = f(t, y(t))$. This leads to the relation $y^{(n)}(t) = f^{(n-1)}(t, y(t))$.

Note that $f'(t, y(t)) = f_t(t, y) + f_y y' = D^1 f$ and

$$\begin{aligned}
 f''(t, y(t)) &= f_{tt}(t, y) + f_{ty}(t, y)y' + (f_{yt} + f_{yy}y')y' + f_y y'' \\
 &= f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_y f_t + f_y^2 f \\
 &= D^2 f
 \end{aligned}$$

etc.

More precisely, the Taylor method of order n reads:

$$(8.3) \quad y_{n+1} = y_n + hf_n + \frac{h^2}{2} Df_n + \cdots + \frac{h^n}{n!} D^{(n-1)} f_n.$$

This method can be implemented in some cases, but requires evaluation of the partial derivatives of f , and is not commonly used.

9. RUNGE-KUTTA METHODS

9.1. Heun's method. Let us consider the Heun's method, which is given by the following:

$$(9.1) \quad y_{n+1} = y_n + h(f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)))/2.$$

To find out the order of the Heun's method, we recall the following :

Lemma 9.1. Assume that $f(t, y)$ and all its partial derivatives of order less than or equal to $n + 1$ are continuous on $D = \{(t, y) : a \leq t \leq b, c \leq y \leq d\}$ and

$(t_0, y_0) \in D$, then the following holds true :

$$\begin{aligned}
 f(t, y) &= f(t_0, y_0) \\
 &+ \sum_{k=1}^n \frac{1}{k!} \sum_{j=0}^k \binom{k}{j} (t - t_0)^{k-j} (y - y_0)^j \frac{\partial^k f}{\partial t^{k-j} \partial y^j} (t_0, y_0) \\
 &+ \frac{1}{(n+1)!} \sum_{j=0}^{n+1} \binom{n+1}{j} (t - t_0)^{n+1-j} (y - y_0)^j \frac{\partial^{n+1} f}{\partial t^{n+1-j} \partial y^j} (\xi_0, \xi_1) \\
 &= P_n(t, y) + R_n(t, y),
 \end{aligned}$$

where ξ_0 is in between t and t_0 and ξ_1 is in between y and y_0 .

The first two terms are called the n th Taylor polynomial in two variables for the function f about (t_0, y_0) and the last term is the remainder term.

Let $\Phi(t_n, y_n) = (f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)))/2$ and by expanding the second term around (t_n, y_n) , we obtain that

$$\begin{aligned}
 \Phi &= f(t_n, y_n) + (t_{n+1} - t_n) f_t(t_n, y_n)/2 + hf(t_n, y_n) f_y(t_n, y_n)/2 + O(h^2) \\
 &= f(t_n, y_n) + hf_t(t_n, y_n)/2 + hf(t_n, y_n) f_y(t_n, y_n)/2 + O(h^2) \\
 &= f_n + \frac{h}{2} D^1 f_n + O(h^2).
 \end{aligned}$$

By comparison with Taylor method of order two, we immediately, notice that the Heun's method is of order two. Note that if we expand Φ out to terms of order h^2 , we get

$$\Phi = f + \frac{h}{2} D^1 f + \frac{h^2}{4} D^2 f + O(h^3).$$

The coefficient of h^2 does not agree with $D^2 f/3!$, so Heun's method is definitely not of higher than second order.

9.2. The second order explicit Runge-Kutta's method (Mid-point method).

In this section, we study the derivation of the second order Runge-Kutta method. Note that the following is the second order Taylor's method reads

$$y_{n+1} = y_n + h\Phi,$$

where

$$\Phi = f(t, y) + \frac{h}{2} (f_t(t, y) + f_y(t, y)y') = f(t, y) + \frac{h}{2} D^1 f.$$

9.2.1. *The mid-point method.* To derive the second order Runge-Kutta method, we shall determine values a_1, α_1, β_1 so that

$$a_1 f(t + \alpha_1, y + \beta_1) = f(t, y) + \frac{h}{2}(f_t(t, y) + f_y(t, y)y') + O(h^2).$$

Such determinations shall lead to more efficient method without spoiling the order of accuracy. Expanding $f(t + \alpha_1, y + \beta_1)$ in its Taylor polynomial of degree one about (t, y) gives

$$\begin{aligned} a_1 f(t + \alpha_1, y + \beta_1) &= a_1 f(t, y) \\ &+ a_1 \alpha_1 f_t(t, y) + a_1 \beta_1 f_y(t, y) + a_1 R_1(t + \alpha_1, y + \beta_1), \end{aligned}$$

where

$$R_1(t + \alpha_1, y + \beta_1) = \frac{\alpha_1^2}{2} f_{tt}(\xi, \mu) + \alpha_1 \beta_1 f_{ty}(\xi, \mu) + \frac{\beta_1^2}{2} f_{yy}(\xi, \mu).$$

Matching coefficients, we are led to have

$$a_1 = 1, \alpha_1 = \frac{h}{2}, \quad \beta_1 = \frac{h}{2} f(t, y).$$

Moreover, $R_1 = O(h^2)$ if the second order derivative of f is bounded.

The mid-point rule is given as follows:

$$\begin{aligned} y_0^h &= \alpha \\ y_{n+1}^h &= y_n^h + h f\left(t + \frac{h}{2}, y_n^h + \frac{h}{2} f(t_n, y_n^h)\right). \end{aligned}$$

One can derive it using the numerical integration using the mid-point rule. It is oftentimes called the mid-point method.

Example 9.1. Use the mid-point method to solve $y' = y - t^2 + 1$ with $0 \leq t \leq 2$ and $y(0) = 0.5$.

The mid-point method reads

$$y_{n+1} = y_n + h \left(y_n + \frac{h}{2} [y_n - t_n^2 + 1] - (t_n + \frac{h}{2})^2 \right).$$

9.2.2. *Modified-Euler Method.* If we used $a_1 f(t, y) + a_2 f(t + \alpha_2, y + \delta_2 f(t, y))$, then we obtain the Modified Euler method, which corresponds to $a_1 = a_2 = \frac{1}{2}$ and $\alpha_2 = \delta_2 = h$. The modified Euler's method reads:

$$y_{n+1} = y_n + \frac{h}{2} [f(t_n, y_n) + h f(t_{n+1}, y_n + h f(t_n, y_n))].$$

9.2.3. *The general explicit RK method.* The derivation of the general form of an explicit RK method can be obtained by setting

$$y_{h+1} = y_n + h(b_1\phi_1 + \cdots + b_q\phi_q),$$

where

$$\phi_i = f(t_n + c_i h, \eta_i)$$

and

$$\begin{aligned} \eta_1 &= y_n, \\ \eta_2 &= y_n + ha_{2,1}\phi_1, \\ \eta_3 &= y_n + h(a_{3,1}\phi_1 + a_{3,2}\phi_2), \\ &\dots \\ \eta_q &= y_n + h(a_{q,1}\phi_1 + \cdots + a_{q,q-1}\phi_{q-1}). \end{aligned}$$

To specify a particular method of this form, we must give the number of stages $q \geq 1$, the coefficients b_i, c_i with $1 \leq i \leq q$, and $a_{ij}, 1 \leq i \leq q, 1 \leq j \leq i$. The b_i is called the weights, c_i or the points $t_n + c_i h$, the nodes and the η_i or ϕ_i are called the stages of *RK*.

One can try to match (with $c_1 = 0$) $b_1\phi_1 + b_2\phi_2$ with the second order Taylor's method for which

$$\Phi = f(t, y) + \frac{h}{2}f'(t, y) + O(h^2),$$

upto $O(h^2)$ where

$$\begin{aligned} b_1\phi_1 + b_2\phi_2 &= b_1f(t + c_1h, y) + b_2f(t + c_2h, y + haf(t + c_1h, y)) \\ &= b_1f(t, y) + b_2f(t + c_2h, y + haf(t, y)) \end{aligned}$$

Note that if we succeed to find constants b_1, b_2, \dots , we obtain method of accuracy of order 2.

Recall that

$$\begin{aligned} b_1f(t, y) + b_2f(t + c_2h, y + haf(t, y)) &= b_1f(t, y) \\ &+ b_2(f(t, y) + c_2hf_t + (ha)ff_y) + O(h^2) \\ &= f(t, y) + \frac{h}{2}(f_t + f_y y') + O(h^2). \end{aligned}$$

From this, we obtain that

$$b_1 + b_2 = 1, b_2c_2h = \frac{h}{2}, b_2haf = \frac{h}{2}y'.$$

Hence, we have

$$b_1 + b_2 = 1, b_2c_2 = \frac{1}{2}, b_2a = \frac{1}{2}.$$

We then obtain $b_1 = b_2 = \frac{1}{2}$ and $c_2 = a = 1$ or $b_1 = \frac{1}{4}$, $b_2 = \frac{3}{4}$ and $c_2 = a = \frac{2}{3}$.
The first choice shall lead to the modified Euler's method

$$y_{n+1} = y_n + \frac{h}{2} (f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n))).$$

and the second choice shall lead to the Heun's method

$$y_{n+1} = y_n + \frac{h}{4} \left(f(t_n, y_n) + 3f\left(t_n + \frac{2}{3}h, y_n + \frac{2}{3}hf(t_n, y_n)\right) \right).$$

Example 9.2. Solve $y' = y - t^2 + 1$ with $0 \leq t \leq 2$ and $y(0) = 0.5$ using Heun's method and Modified Euler method.

The Modified Euler Method reads

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2} (y_n - t_n^2 + 1 + y_n + hf(t_n, y_n) - t_{n+1}^2 + 1) \\ &= y_n + \frac{h}{2} (y_n - t_n^2 + 1 + y_n + h(y_n - t_n^2 + 1) - t_{n+1}^2 + 1) \end{aligned}$$

and the Heun's method reads

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{4} \left(y_n - t_n^2 + 1 + 3\left(y_n + \frac{h}{3}f(t_n, y_n) - (t_n + \frac{2}{3}h)^2 + 1\right) \right) \\ &= y_n + \frac{h}{4} \left(y_n - t_n^2 + 1 + 3\left(y_n + \frac{h}{3}(y_n - t_n^2 + 1) - (t_n + \frac{2}{3}h)^2 + 1\right) \right) \end{aligned}$$

The most popular method is the Runge-Kutta method of order Four :

Definition 9.1 (RK of order four).

$$\begin{aligned} \phi_1 &= f(t_n, y_n) \\ \phi_2 &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}\phi_1\right) \\ \phi_3 &= f\left(t_n + \frac{h}{2}, y_n + \frac{1}{2}\phi_2\right) \\ \phi_4 &= f(t_{n+1}, y_n + \phi_3) \\ y_{n+1} &= y_n + \frac{h}{6} (\phi_1 + 2\phi_2 + 2\phi_3 + \phi_4). \end{aligned}$$

Note that since it is of order four, if we let Φ be the Taylor's method of order four, then

$$\frac{1}{6} (\phi_1 + 2\phi_2 + 2\phi_3 + \phi_4) = \Phi + O(h^4).$$

10. STIFF EQUATIONS AND ABSOLUTE STABILITY

Recall the error analysis for the Euler's method, for which the convergence result has been given as $|e_n| \leq Ch$ for some constant C . The problem is that it may not give a quantitative indication of what happens when we actually compute with a time step size h , which is not very small.

For large step sizes, the difference equation should mimic the behavior of the differential equation in the sense that their stability properties should be similar.

Example 10.1. *Solve $y' = \lambda y$ using Euler Method with $h = 2.1$ when $\lambda = -1$. What happens if you increase the magnitude of λ keeping its sign negative ?*

What we are seeing is the stiffness problem.

10.1. Stiffness. The stiffness problem can be characterized as the following: The solution contains rapidly varying transients which decay quickly, but for some reason require us to take small step sizes even after they disappeared from the solution. Namely, the problem is not from the accuracy, but from something else. Such a stiff problems are important, because they arise in a number of applications including chemical reaction modeling, numerical solution of parabolic and hyperbolic PDEs, control theory, and electric circuit modeling, etc.

The goal is to find a method which can handle such stiff problems. Since the stiffness problem is as mentioned identified to be related to the stiff decays of the solution, we have taken the model problem

$$y' = \lambda y, \quad y(0) = 1.$$

where $\mathcal{R}e(\lambda) < 0$. This example shows clearly why the realization of numerical solution that respect the correct behaviour of the given IVP is so important and the scheme should be made in a way that the scheme produces the approximations that have the property of the exact solution as much as possible.

More concrete reason for the breakdown of the Euler's method for Example 10.1 is given as follows : We notice that e^{-t_n} is being approximated by $(1 - h)^n$ with $t_n = nh$. Now if $h \ll 1$, this is a reasonable approximation. Then $e^{-h} \approx 1 - h$ and $e^{-hn} \approx (1 - h)^n$. But if h is not so small, then $(1 - h)^n$ does not behave at all like e^{-hn} . In fact, if $h > 2$, then $1 - h$ is a negative number of magnitude greater than one, and so $(1 - h)^n$ is exponentially growing, and alternating sign.

This can be avoided for example if we choose $|1 - h| < 1$. Note that this requires small time step used.

Consider the following model problem.

$$(10.1) \quad y' = \lambda y, \quad y(0) = 1.$$

The exact solution is

$$y(t_n) = e^{\lambda t_n}.$$

The euler's method with a uniform step size h gives

$$y_n = y_{n-1} + h\lambda y_{n-1} = (1 + h\lambda)y_{n-1} = \cdots = (1 + h\lambda)^n.$$

In case $\mathcal{R}e(\lambda) < 0$, $|y(t)|$ decays exponentially. The numerical solution y_n should decay, as n increases. Generally this shall imply some restriction on the step size and λ is negative with larger magnitude, the restriction of the step size are more severe.

This yields an additional absolute stability requirement,

$$|y_n| < |y_{n-1}|, \quad n = 1, 2, \cdots$$

Now, for virtually any method, the values of y_n depends only on the product $h\lambda$.

Definition 10.1 (Region of absolute stability). *Consider the model problem*

$$y' = \lambda y$$

with $\mathcal{R}e(\lambda) < 0$, the absolute stability region is defined to be

$$(10.2) \quad S = \{h\lambda \in \mathbb{C} : \lim_{n \rightarrow \infty} y_n = 0, \}$$

where y_n is the numerical solution to (10.1) with step size h .

For the forward Euler method, we obtain the condition that

$$(10.3) \quad |1 + h\lambda| < 1.$$

Now, consider the improved Euler method,

$$(10.4) \quad y_{n+1} = y_n + h[f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n))]/2.$$

Applied to the equation $y' = \lambda y$, this becomes

$$(10.5) \quad y_{n+1} = y_n + h[\lambda y_n + \lambda(y_n + h\lambda y_n)]/2 = y_n(1 + \bar{h} + \bar{h}^2)/2,$$

where $\bar{h} = h\lambda$. The region for stability for the improved Euler method is $S = \{\bar{h} \in \mathbb{C} : |1 + \bar{h} + \bar{h}^2| < 1\}$. Let us now consider the backward Euler method given as follows:

$$(10.6) \quad y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

Applied to the equation $y' = \lambda y$, this becomes

$$(10.7) \quad y_{n+1} = y_n + h\lambda y_{n+1} = y_n + \bar{h}y_{n+1},$$

where $\bar{h} = h\lambda$. Note that $y_{n+1} = (1/(1 - \bar{h}))^n$ and the stability region is $S = \{\bar{h} \in \mathbb{C} : |1/(1 - \bar{h})| < 1\}$, the left half plane. This means that for any $h > 0$, the stability requirement holds true.

Finally, we consider the implicit Trapezoidal method:

$$(10.8) \quad y_{n+1} = y_n + \frac{h}{2} (f(t_n, y_n) + f(t_{n+1}, y_{n+1}))$$

Applied to the equation $y' = \lambda y$, this becomes

$$(10.9) \quad y_{n+1} = y_n + \frac{h}{2} (\lambda y_{n+1} + \lambda y_n) = y_n + \frac{\bar{h}}{2} y_{n+1} + \frac{\bar{h}}{2} y_n,$$

for which the solution $y_{n+1} = (1 + \bar{h}/2)/(1 - \bar{h}/2)y_n$. It is easy to see that the method is A-stable.

Several remarks are in order:

- A method is called A-stable if its region of absolute stability contains the entire left half plane (so whenever the exact solution decays, so does the numerical solution).
- All the explicit methods can be shown to have the bounded region of stability. Namely, only implicit methods can be A-stable.
- The unreasonable behavior of numerical solutions obtained with certain mesh size for which the AS violates has nothing to do with the local error of the numerical method. The step size should be depressed not by accuracy but by instability.

Example 10.2. *Try to solve the following equation using explicit Euler.*

$$(10.10) \quad y' = -100(y - \sin(t)), \quad t \geq 0, \quad y(0) = 1.$$

11. MULTISTEP METHODS

Euler's method is an example of a one-step method : the numerical solution y_{n+1} at t_{n+1} , the current time is determined from the numerical approximation at the single preceding point t_n . More generally, we can consider methods with a constant step size h and determine y_{n+1}^h using the values from several preceding steps :

$$(11.1) \quad y_{n+1} = \Phi(f, t_n, y_{n+1}, y_n, y_{n-1}, \dots, y_{n-k}, h).$$

Here y_{n+1} depends on $k+1$ previous numerical approximations $y_n, y_{n-1}, \dots, y_{n-k}$, so this is called a $k+1$ step method. Notice that we have allowed y_{n+1} to appear in the right hand side of the equation as well as the left. When this happens, the method is called an implicit method, and need to solve, in general, a nonlinear equation to determine y_{n+1} . In any case, we need to determine the first $k+1$ values y_0, \dots, y_k by some other method, such as a single step (higher order) method.

If Φ does not depend on y_{n+1} , the method is called an explicit method. A simple example of explicit method is the explicit Euler's method:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}),$$

and an example of implicit methods can be the trapezoidal method:

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})).$$

These are all one-step method.

11.1. linear multistep methods.

$$(11.2) \quad y_{n+1}^h = -a_0 y_n - a_1 y_{n-1} + \cdots - a_k y_{n-k} + h[b_{-1} f_{n+1} + b_0 f_n + \cdots + b_k f_{n-k}].$$

is called the linear multistep methods with constant step size. For an explicit linear multistep method $b_{-1} = 0$.

It is also convenient to define

$$\sum_{j=-1}^k a_j y_{n-j} = h \sum_{j=-1}^k b_j f_{n-j},$$

where $a_{-1} = 1$. One obvious question concerning implicit linear multistep methods is whether or not the method determines y_{n+1} . The answer is yes if h is sufficiently small. (Why? : Define

$$y_{n+1} = F(y_{n+1}),$$

where

$$F(y_{n+1}) = - \sum_{j=0}^j a_j y_{n-j} + h \sum_{j=-1}^k b_j f(t_{n-j}, y_{n-j}).$$

If F is contractive, then by the fixed point theorem, there is a unique fixed point, which is y_{n+1} . This is however, true when h is sufficiently small. The contraction mapping theorem also implies that the solution can be computed by fixed point iteration and this is often done in practice. Of course, only few iterations are made.)

Example 11.1. Show that if h is sufficiently small, the implicit Euler method

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$$

determines y_{n+1} uniquely.

11.2. Adams' Method. An important example of the multistep method can be Adams method.

Example 11.2. Given y_0, y_1, y_2 and y_3 ,

$$y_{n+1} = y_n + \frac{h}{24} (55f(t_n, y_n) - 59f(t_{n-1}, y_{n-1}) + 37f(t_{n-2}, y_{n-2}) - 9f(t_{n-3}, y_{n-3}))$$

This is called the fourth order Adams-Bashforth method.

Example 11.3. Given y_0, y_1 and y_2 ,

$$y_{n+1} = y_n + \frac{h}{24} (9f(t_{n+1}, y_{n+1}) + 19f(t_n, y_n) - 5f(t_{n-1}, y_{n-1}) + f(t_{n-2}, y_{n-2}))$$

This is called the fourth order Adams-Moulton method. In this section, we shall study how to derive such Adams' method in general.

11.2.1. *Derivation of Adams method.* Note first that the initial value problem

$$y'(t) = f(t, y), \quad a \leq t \leq b, \quad y(a) = y_0$$

is equivalent to

$$y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} y'(t) dt = \int_{t_n}^{t_{n+1}} f(t, y(t)) dt$$

The main idea in the derivation of Adams Method is to approximate the integrand by an appropriate approximating polynomial, which will be chosen to be the Lagrange interpolant for $f(t, y(t))$

We recall the formula of the Lagrange interpolant: Consider we are given $n+1$ distinct points $t_0 < t_1 < t_2 < \dots < t_n$ and $n+1$ values g_i with $i = 0, \dots, n$. Then there exists a unique polynomial $p(t)$ such that $g(t_i) = g_i$ for $0 \leq i \leq n$, which can be given as follows:

$$(11.3) \quad p(t) = \sum_{k=0}^n g_k \prod_{0 \leq m \leq n, m \neq k} \frac{t - t_m}{t_k - t_m}.$$

Example 11.4. For $n = 1$, namely, two distinct points are interpolated, then the Lagrange interpolant is given by

$$p(t) = g_0 \frac{t - t_1}{t_0 - t_1} + g_1 \frac{t - t_0}{t_1 - t_0}.$$

and $n = 2$, then

$$p(t) = g_0 \frac{t - t_1}{t_0 - t_1} \frac{t - t_2}{t_0 - t_2} + g_1 \frac{t - t_0}{t_1 - t_0} \frac{t - t_2}{t_1 - t_2} + g_2 \frac{t - t_0}{t_2 - t_0} \frac{t - t_1}{t_2 - t_1}.$$

We now assume that y_j is known for $0 \leq j \leq n$, let $p(t) \in \mathcal{P}_k$ denote the Lagrange interpolating polynomial satisfying

$$p(t_j) = f_j := f(t_j, y_j), \quad j = n, n-1, \dots, n-k.$$

We then define

$$(11.4) \quad y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} p(t) dt,$$

where

$$(11.5) \quad p(t) = \sum_{j=0}^k f_{n-j} \prod_{0 \leq m \leq k, m \neq j} \frac{t - t_{n-m}}{t_{n-j} - t_{n-m}}.$$

The better form will be

$$y_{n+1} = y_n + h \sum_{j=0}^k b_j f_{n-j},$$

where

$$\begin{aligned} b_j &= \frac{1}{h} \int_{t_n}^{t_{n+1}} \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{t - t_{n-m}}{t_{n-j} - t_{n-m}} \\ &= \int_0^1 \prod_{0 \leq m \leq k, m \neq j} \frac{m+t}{m-j} dt. \end{aligned}$$

The more compact form is

$$b_j = (-1)^{j-1} \sum_{i=j-1}^{k-1} \binom{i}{j-1} (-1)^i \int_0^1 \binom{-s}{i} ds.$$

Recall that

$$\binom{s}{i} = \frac{s(s-1)\cdots(s-i+1)}{i}, \quad \binom{s}{0} = 1.$$

This shall lead to the Adams-Bashford method. For $k = 0, 1$, we have

$$\begin{aligned} y_{n+1} &= y_n + h f_n \\ y_{n+1} &= y_n + h \left(\frac{3}{2} f_n - \frac{1}{2} f_{n-1} \right) \end{aligned}$$

The Adams-Moulton methods are constructed similarly, except that $p \in \mathcal{P}$ interpolates f_{n-j} at t_{n-j} for $j = -1, \dots, k$. The first few formulas are

$$\begin{aligned} y_{n+1} &= y_n + \frac{h}{2}(f_{n+1} + f_n) \\ y_{n+1} &= y_n + h \left(\frac{5}{12} f_{n+1} + \frac{2}{3} f_n - \frac{1}{12} f_{n-1} \right) \end{aligned}$$

Example 11.5. *Derive a single step Adams-Moulton methods. The Lagrange interpolating polynomial of f at t_n and t_{n+1} are given as follows:*

$$p(t) = f_{n+1} \frac{t - t_n}{t_{n+1} - t_n} + f_n \frac{t - t_{n+1}}{t_n - t_{n+1}}.$$

The numerical method will then be obtained from the following relation that

$$(11.6) \quad y(t_{n+1}) \approx y(t_n) + \int_{t_n}^{t_{n+1}} p(t) dt.$$

Simple calculation leads to

$$\begin{aligned} \int_{t_n}^{t_{n+1}} p(t) dt &= \int_{t_n}^{t_{n+1}} f_{n+1} \frac{t - t_n}{t_{n+1} - t_n} + f_n \frac{t - t_{n+1}}{t_n - t_{n+1}} dt \\ &= \frac{h}{2} (f_n + f_{n+1}). \end{aligned}$$

11.2.2. *The order of Adams method.* It is easy to check the order of the Adams methods. For example, for the $k + 1$ step Adams-Bashford method, let

$$L(y, h, t_n) = \int_{t_n}^{t_{n+1}} [f(t, y(t)) - p(t)] dt,$$

where $p \in \mathcal{P}_k$ interpolates f at t_n, \dots, t_{n-k} . By the Newton error formula for Lagrange interpolation,

$$f(t, y(t)) - p(t) = \frac{1}{(k+1)!} D^{k+1} f(\eta) (t - t_n) \cdots (t - t_{n-k}),$$

so using the integral mean value theorem,

$$L(y, h, t_n) = \frac{1}{(k+1)!} D^{k+1} f(\eta) \int_{t_n}^{t_{n+1}} (t - t_n) \cdots (t - t_{n-k}) dt = \gamma_{k+1} h^{k+2}.$$

For example, $\gamma_1 = \frac{1}{2}$, $\gamma_2 = \frac{5}{12}$, \dots . Therefore, we conclude that

$$(11.7) \quad y(t_{n+1}) = y(t_n) + \int_{t_n}^{t_{n+1}} p(t) dt$$

has the following local truncation error:

$$\begin{aligned} \tau &= \frac{y(t_{n+1}) - \left(y(t_n) + \int_{t_n}^{t_{n+1}} p(t) dt \right)}{h} \\ &= \frac{y(t_n) + \int_{t_n}^{t_{n+1}} f(t, y(t)) dt - \left(y(t_n) + \int_{t_n}^{t_{n+1}} p(t) dt \right)}{h} \\ &= \frac{\int_{t_n}^{t_{n+1}} f(t, y(t)) dt - \int_{t_n}^{t_{n+1}} p(t) dt}{h} \\ &= \frac{L(y, h, t_n)}{h} = \gamma_{k+1} h^{k+1} = O(h^{k+1}). \end{aligned}$$

The computations of local truncation error and the conclusion is that the Adams method has order of accuracy equal to $O(h^{k+1})$. For a k step Adams-Moulton method,

$$L(y, h, t_n) = \int_{t_n}^{t_{n+1}} [f(t, y(t)) - p(t)] dt,$$

where $p \in \mathcal{P}_k$ interpolates f at $t_{n+1}, t_n, \dots, t_{n-(k-1)}$. Similarly, using the Newton's error formula:

$$f(t, y(t)) - p(t) = \frac{1}{(k+1)!} D^{k+1} f(\eta)(t - t_{n+1}) \cdots (t - t_{n-(k-1)}),$$

and

$$L(y, h, t_n) = \gamma_{k+1}^* h^{k+2}$$

and we achieve the local truncation error like

$$\tau = \frac{L(y, h, t_n)}{h} = \gamma^* h^{k+1}.$$

Hence, the order of accuracy is still $k+1$. However the coefficient in the local truncation error is significantly smaller for the Adams-Moulton method. (Homework IV).

11.3. BDF method. Adams methods and various other linear multistep methods have small regions of absolute stability. Other kinds of linear multistep method is preferred. The most popular are the backward differentiation formula method or BDF methods of the following form :

$$\sum_{j=-1}^k a_j y_{n-j} = h f_{n+1}.$$

The coefficients a_j are determined by interpolating y_{n-j} at t_{n-j} with $j = -1, 0, \dots, k$ by a polynomial $p(t)$ of degree k and evaluating $p'(t_{n+1})$ and set

$$(11.8) \quad p'(t_{n+1}) = f_{n+1}.$$

The first two BDF method is

$$\begin{aligned} y_{n+1} - y_n &= h f_{n+1} \\ (3y_{n+1} - 4y_n + y_{n-1})/2 &= h f_{n+1}. \end{aligned}$$

k step BDF method is of order k .

11.3.1. *The Predictor-Corrector Scheme.* The implicit methods have the inherent weakness of first having to convert the method algebraically to an explicit representation for y_{n+1} . Let us consider the following example :

$$y' = e^y, \quad 0 \leq t \leq 0.25, \quad y(0) = 1.$$

Since $f(t, y) = e^y$, the three-step Adams-Moulton method is given as follows:

$$y_{n+1} = y_n + \frac{h}{24} (9e^{y_{n+1}} + 19e^{y_n} - 5e^{y_{n-1}} + e^{y_{n-2}}).$$

In practice, the implicit methods are seldom used. The most common method is to solve the equation approximately using a small number of fixed point iterations starting from an initial approximation obtained by an explicit method can be given as follows:

1. predict $p_{n+1} = E(y_n, y_{n-1}, \dots, f_n, f_{n-1} \dots ,)$
 2. evaluate $f_{n+1}^p = f(t_{n+1}, p_{n+1})$
 3. correct $y_{n+1}^{(1)} = I(y_n, y_{n-1}, \dots, f_{n+1}^p, f_n, f_{n-1} \dots ,)$
 4. evaluate $f_{n+1}^{(1)} = f(t_{n+1}, y_{n+1}^{(1)})$
 5. correct $y_{n+1}^{(2)} = I(y_n, y_{n-1}, \dots, f_{n+1}^{(1)}, f_n, f_{n-1} \dots ,)$
 6. evaluate $f_{n+1}^{(2)} = f(t_{n+1}, y_{n+1}^{(2)})$
- .
- .
- .

As a simple example, consider the PEC method with a 2 step Adams-Bashford predictor and a 2 step Adams-Moulton corrector.

This gives :

$$p_{n+1} = y_n + h[3f(t_n, y_n) - f(t_{n-1}, y_{n-1})]/2$$

$$y_{n+1} = y_n + h[5f(t_{n+1}, p_{n+1}) + 8f(t_n, y_n) - f(t_{n-1}, y_{n-1})]/12.$$

One can then obtain the nonlinear 2-step method as below.

Exercice 1. *Compute the local truncation error of this scheme?*

Remark 11.1.

12. SYSTEMS OF DIFFERENTIAL EQUATIONS

An m -th order system of first order initial value problems has the form

$$\begin{aligned}\frac{du^1}{dt}(t) &= f^1(t, u^1, \dots, u^m), \\ \frac{du^2}{dt}(t) &= f^2(t, u^1, \dots, u^m), \\ &\vdots \\ \frac{du^m}{dt}(t) &= f_m(t, u^1, \dots, u^m),\end{aligned}$$

for $a \leq t \leq b$, with the initial conditions

$$u^1(a) = \alpha_1, \quad u^2(a) = \alpha_2, \dots, u^m(a) = \alpha_m.$$

Let $U(a) = (u^1(a), \dots, u^m(a))^T$, $U(t) = (u^1(t), \dots, u^m(t))^T$, and $F(t, U) = (f^1(t, U), \dots, f^m(t, U))^T$. Then we may write the system of equation as the following compact form:

$$(12.1) \quad \frac{dU}{dt} = F(t, U),$$

which looks like the initial value problem that discussed in the previous section.

Theorem 12.1. *Suppose*

$$D = \{(t, u^1, u^2, \dots, u^m) : a \leq t \leq b, -\infty < u^i < \infty, \text{ for each } i = 1, \dots, m\},$$

and

$$(12.2) \quad \|F(t, U) - F(t, Z)\| \leq L\|U - Z\|,$$

where L is some constant and $\|\cdot\|$ is norm defined by the following, for $X = (X^1, \dots, X^m)^T$,

$$\|X\| = \sum_{i=1}^m |X^i|.$$

Then system has a unique solution subject to the initial conditions.

We now introduce some numerical scheme to solve the system given above. Such problems can be solved by some generalizations of the methods that have been discussed in the previous section.

For example, the Euler method is given as follows:

$$(12.3) \quad U_{n+1} = U_n + hF(t_n, U_n).$$

Note that

$$\begin{aligned} U_n &= (u_n^1, u_n^2, \dots, u_n^m) \\ F(t_n, U_n) &= (f^1(t_n, u_n^1, \dots, u_n^m), \dots, f^m(t_n, u_n^1, \dots, u_n^m))^T \end{aligned}$$

This can be written explicitly as follows:

$$\begin{aligned} u_{n+1}^1 &= u_n^1 + hf^1(t_n, u_n^1, u_n^2, \dots, u_n^m) \\ u_{n+1}^2 &= u_n^2 + hf^2(t_n, u_n^1, u_n^2, \dots, u_n^m) \\ &\vdots \\ u_{n+1}^m &= u_n^m + hf^m(t_n, u_n^1, u_n^2, \dots, u_n^m) \end{aligned}$$

Now, we shall write the RK method of order four for the system of ordinary differential equations.

$$\begin{aligned} U_0 &= U(a) \\ \Phi_1 &= F(t_n, U_n) \\ \Phi_2 &= F\left(t_n + \frac{h}{2}, U_n + \frac{1}{2}h\Phi_1\right) \\ \Phi_3 &= F\left(t_n + \frac{h}{2}, U_n + \frac{1}{2}h\Phi_2\right) \\ \Phi_4 &= F(t_{n+1}, U_n + h\Phi_3) \\ U_{n+1} &= U_n + \frac{h}{6}(\Phi_1 + 2\Phi_2 + 2\Phi_3 + \Phi_4). \end{aligned}$$

We suppose now that the approximants are known as U_n at time step n and set $\Phi_k = (\phi_k^1, \dots, \phi_k^m)^T$ for $k = 1, 2, 3, 4$.

Then one can compute U_{n+1} with $1 \leq i \leq m$ as follows explicitly : For each $i = 1, \dots, m$,

$$\begin{aligned} \phi_1^i &= f_i(t_n, u_n^1, \dots, u_n^m) \\ \phi_2^i &= f_i\left(t_n + \frac{h}{2}, u_n^1 + \frac{1}{2}h\phi_1^1, \dots, u_n^m + \frac{1}{2}h\phi_1^m\right) \\ \phi_3^i &= f_i\left(t_n + \frac{h}{2}, u_n^1 + \frac{1}{2}h\phi_2^1, \dots, u_n^m + \frac{1}{2}h\phi_2^m\right) \\ \phi_4^i &= f_i(t_{n+1}, u_n^1 + h\phi_3^1, \dots, u_n^m + h\phi_3^m) \\ u_{n+1}^i &= u_n^i + \frac{h}{6}(\phi_1^i + 2\phi_2^i + 2\phi_3^i + \phi_4^i). \end{aligned}$$

Example 12.1. Solve the following second order initial value problem using the Explicit Euler method.

$$(12.4) \quad y'' - 2y' + 2y = e^{2t} \sin(t) \quad \text{for } 0 \leq t \leq 1,$$

subject to the boundary conditions $y(0) = -0.4$ and $y'(0) = -0.6$. It is well-known that the second order initial value problem can be reformulated into the first order system of initial value problem. Introduce $u^1(t) = y(t)$ and $u^2(t) = y'(t)$. Then the equation (12.4) becomes :

$$\begin{aligned} \frac{du^1(t)}{dt} &= u^2(t) \\ \frac{du^2(t)}{dt} &= e^{2t} \sin(t) + 2u^2(t) - 2u^1(t), \end{aligned}$$

subject to the initial conditions $u^1(0) = -0.4$ and $u^2(0) = -0.6$. simple Euler method reads :

$$U_{n+1} = U_n + hF(t_n, U_n),$$

where with $f_1(t, U) = u^2(t)$ and $f_2(t, U) = e^{2t} \sin(t) + 2u^2(t) - 2u^1(t)$, we have

$$\begin{aligned} U_n &= (u_n^1, u_n^2) \\ F(t_n, U_n) &= \begin{pmatrix} f_1(t_n, u_n^1, u_n^2) \\ f_2(t_n, u_n^1, u_n^2) \end{pmatrix} = \begin{pmatrix} u_n^2 \\ \exp^{2t_n} \sin(t_n) + 2u_n^2 - 2u_n^1 \end{pmatrix}, \end{aligned}$$

Exercise 2. Try to write an explicit form of RK method of order four for the above example.

13. FINITE-DIFFERENCE METHODS FOR LINEAR PROBLEMS

In this section, we study the discretization of the linear second-order boundary value problem

$$(13.1) \quad y'' = p(x)y' + q(x)y + r(x), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

We first look at the simple example

$$(13.2) \quad y'' = r(x), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta.$$

To solve such a problem numerically, the first step is to choose nodes (N will be number of interior nodes). $x_i = a + ih$, for $i = 0, \dots, N+1$ with $h = (b-a)/(N+1)$ at which we shall find the solution, say y_i , only finite number of solutions. In between solution can be obtained by e.g., the linear interpolation.

Secondly, we consider the equations at each node:

$$y''(x_i) = r(x_i), \quad \forall i = 0, \dots, N+1$$

Thirdly, we approximate $y''(x_i)$ by certain finite difference using $y(x_i), y(x_{i-1}), y(x_{i+1})$ etc.

13.1. Centered Difference Formula. In general, $y''(x_i)$ is often approximated by the following centered difference formula:

$$(13.3) \quad y''(x_i) \approx \frac{1}{h^2} (y(x_{i+1}) - 2y(x_i) + y(x_{i-1})).$$

Let us set $N = 3$ and construct the linear system of equation for $y'' = r$. Note $h = (b-a)/4$ and $x_0 = a, x_1 = a+h, x_2 = a+2h, x_3 = a+3h$ and $x_4 = a+4h = b$.

At the boundary node x_0 and x_4 , the equations are given as $y(a) = \alpha$ and $y(b) = \beta$. At the interior points, we have

$$y''(x_1) = r(x_1) \approx \frac{1}{h^2} (y(x_0) - 2y(x_1) + y(x_2))$$

$$y''(x_2) = r(x_2) \approx \frac{1}{h^2} (y(x_1) - 2y(x_2) + y(x_3))$$

$$y''(x_3) = r(x_3) \approx \frac{1}{h^2} (y(x_2) - 2y(x_3) + y(x_4))$$

We then form the following linear system of equation:

$$(13.4) \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} \alpha \\ h^2 r(x_1) \\ h^2 r(x_2) \\ h^2 r(x_3) \\ \beta \end{pmatrix}.$$

Or equivalently,

$$(13.5) \quad \begin{pmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} h^2 r(x_1) - \alpha \\ h^2 r(x_2) \\ h^2 r(x_3) - \beta \end{pmatrix}.$$

We also approximate $y'(x_i)$ by a centered-difference formula in a similar manner

$$(13.6) \quad y'(x_i) \approx \frac{1}{2h} (y(x_{i+1}) - y(x_{i-1})).$$

We now see how accurate the centered difference formula as an approximation.

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + hy'(x_i) + \frac{h^2}{2}y''(x_i) + \frac{h^3}{3!}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi^+),$$

$$y(x_{i-1}) = y(x_i - h) = y(x_i) - hy'(x_i) + \frac{h^2}{2}y''(x_i) - \frac{h^3}{3!}y'''(x_i) + \frac{h^4}{24}y^{(4)}(\xi^-),$$

Therefore,

$$y(x_{i+1}) + y(x_{i-1}) = 2y(x_i) + h^2y''(x_i) + \frac{h^4}{24}(y^{(4)}(\xi^+) + y^{(4)}(\xi^-)),$$

This implies that (by intermediate value theorem)

$$\begin{aligned} \frac{1}{h^2}(y(x_{i+1}) - 2y(x_i) + y(x_{i-1})) &= y''(x_i) + \frac{h^2}{24}(y^{(4)}(\xi^+) + y^{(4)}(\xi^-)) \\ &= y''(x_i) + \frac{h^2}{12}y^{(4)}(\eta), \end{aligned}$$

Similarly,

$$y'(x_i) = \frac{1}{2h}(y(x_{i+1}) - y(x_{i-1})) - \frac{h^2}{6}y'''(\eta).$$

Example 13.1. Apply the centered difference scheme to discretize the differential equation

$$y'' + y' = 2 + 2x, \quad 0 \leq x \leq 1, \quad y(0) = 0, y(1) = 1.$$

$$\begin{aligned} \frac{y(x_2) - 2y(x_1) + y(x_0)}{h^2} + \frac{y(x_2) - y(x_0)}{2h} &= 2 + 2x_1 \\ \frac{y(x_3) - 2y(x_2) + y(x_1)}{h^2} + \frac{y(x_3) - y(x_1)}{2h} &= 2 + 2x_2 \\ \frac{y(x_4) - 2y(x_3) + y(x_2)}{h^2} + \frac{y(x_4) - y(x_2)}{2h} &= 2 + 2x_3 \\ &\vdots \\ \frac{y(x_N) - 2y(x_{N-1}) + y(x_{N-2})}{h^2} + \frac{y(x_N) - y(x_{N-2})}{2h} &= 2 + 2x_N \\ \frac{y(x_{N+1}) - 2y(x_N) + y(x_{N-1}))}{h^2} + \frac{y(x_{N+1}) - y(x_{N-1}))}{2h} &= 2 + 2x_N \end{aligned}$$

As the linear system of equation, we have the following: $Ax = b$, with

$$A = \begin{pmatrix} \frac{-2}{h^2} & \frac{1}{h^2} + \frac{1}{2h} & 0 & 0 & 0 & \cdots & 0 \\ \frac{1}{h^2} - \frac{1}{2h} & \frac{-2}{h^2} & \frac{1}{h^2} + \frac{1}{2h} & 0 & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{h^2} - \frac{1}{2h} & \frac{-2}{h^2} & \frac{1}{h^2} + \frac{1}{2h} \\ 0 & \cdots & \cdots & 0 & 0 & \frac{1}{h^2} - \frac{1}{2h} & \frac{-2}{h^2} \end{pmatrix}$$

$$x = (y_1, y_2, \dots, y_N)^T \quad \text{and}$$

$$b = (2 + 2x_1 + (-\frac{1}{h^2} + \frac{1}{2h})y_0, 2 + 2x_2,$$

$$\dots, 2 + 2x_{N-1}, 2 + 2x_N + (-\frac{1}{h^2} - \frac{1}{2h})y_{N+1})$$

14. NORMS OF VECTORS AND MATRICES

A vector \mathbf{x} in \mathbb{R}^n is given by

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \quad \text{with } x_i \in \mathbb{R}.$$

A vector norm on \mathbb{R}^n is a function, $\|\cdot\|$ from \mathbb{R}^n to \mathbb{R} with the following properties

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

Example 14.1. The ℓ_2 and ℓ_∞ norms for the vector $\mathbf{x} = (x_1, \dots, x_n)^T$ are given by

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^n x_i^2 \right)^{1/2} \quad \text{and} \quad \|\mathbf{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

Excercise 3. Show that ℓ_2 and ℓ_∞ are norms.

Example 14.2. The vector $\mathbf{x} = (-1, 1, -2)^T \in \mathbb{R}^3$ has norms

$$\|\mathbf{x}\|_2 = \sqrt{(-1)^2 + (1)^2 + (-2)^2} = \sqrt{6}$$

and

$$\|\mathbf{x}\|_\infty = \max\{|-1|, |1|, |-2|\} = 2.$$

Theorem 14.1. The following holds true :

$$\mathbf{x}^T \mathbf{y} = \mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\|_2 \|\mathbf{y}\|_2 \cos \theta,$$

where θ is an angle between two vector \mathbf{x} and \mathbf{y} .

Definition 14.1 (Distance between two vectors). If $\mathbf{x} = (x_1, \dots, x_n)^T$ and $\mathbf{y} = (y_1, \dots, y_n)^T$ are vectors in \mathbb{R}^n , then ℓ_2 and ℓ_∞ distances between \mathbf{x} and \mathbf{y} are defined by

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

and

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

This concept is useful to measure how much two vectors are different. For example, consider the following linear system of equation :

$$Ax = b$$

Now assume that the actual solution is $x = (1, 1, 1)^T$ but, by some numerical scheme, you obtained $\tilde{x} = (1.2, 0.99, 0.92)^T$. Then the error between the real solution and computed one can be quantitatively given in terms of ℓ_2 or ℓ_∞ norms as follows :

$$\|x - \tilde{x}\|_2 = ?$$

$$\|x - \tilde{x}\|_\infty = ?.$$

Definition 14.2. A sequence $\{\mathbf{x}^k\}$ of vectors in \mathbb{R}^n is said to converge to \mathbf{x} with respect to norm $\|\cdot\|$ if, given any $\varepsilon > 0$ there exists an integer $N(\varepsilon)$ such that

$$\|\mathbf{x}^k - \mathbf{x}\| < \varepsilon, \quad \text{for all } k \geq N(\varepsilon).$$

Theorem 14.2. $\mathbf{x}^k \rightarrow \mathbf{x}$ in $\|\cdot\|_\infty$ or $\|\cdot\|_2$ if and only if $x_i \rightarrow x$ for all i .

Proof. The case $\|\cdot\|_\infty$ is clear. For $\|\cdot\|_2$, we need the following result :

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

□

Example 14.3. Let $\mathbf{x}^k \in \mathbb{R}^4$ be defined by

$$\mathbf{x}^k = \left(1, 2 + \frac{1}{k}, \frac{3}{k^2}, e^{-k} \sin k \right)^T.$$

As $k \rightarrow \infty$, where does \mathbf{x}^k go with respect to $\|\cdot\|_\infty$?

Definition 14.3. A matrix norm on the set of all $n \times n$ matrices is a real-valued function, $\|\cdot\|$ defined on this set, satisfying the following :

- $\|A\| \geq 0$;
- $\|A\| = 0$, if and only if $A = 0$, the matrix with all zero entries;
- $\|\alpha A\| = |\alpha|\|A\|$;
- $\|A + B\| \leq \|A\| + \|B\|$;
- $\|AB\| \leq \|A\|\|B\|$.

Theorem 14.3. If $\|\cdot\|$ is a vector norm on \mathbb{R}^n , then

$$\|A\| = \max_{\|x\|=1} \|Ax\| = \max_{z \neq 0} \frac{\|Az\|}{\|z\|}$$

is a matrix norm.

We remark that the matrix norm induced from the vector norm is called the natural matrix norm.

Example 14.4. Two natural norms of the matrix induced from ℓ_∞ and ℓ_2 are respectively given by

$$\|A\|_\infty = \max_{\|x\|_\infty=1} \|Ax\|_\infty$$

and

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$$

Note that in case A is symmetric, then $\|A\|_2 = \rho(A) = \max\{|\lambda_1|, \dots, |\lambda_n|\}$, where λ_i 's are eigenvalues of A .

Theorem 14.4. If $A = (a_{ij})$ is an $n \times n$ matrix, then $\|A\|_\infty$ is the maximum row sum, namely

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

Example 14.5. If

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 0 & 3 & -1 \\ 5 & -1 & 1 \end{bmatrix},$$

then compute $\|A\|_\infty$. Note that each row sum is 4, 4 and 7. Hence $\|A\|_\infty = 7$.

15. ITERATIVE TECHNIQUES FOR SOLVING LINEAR SYSTEMS

In this section, we consider some classical iterative methods to solve

$$Au = f,$$

where $A \in \mathbb{R}^{n \times n}$ and $f \in \mathbb{R}^n$. The main assumption that we shall take for A is that it is **symmetric** and **positive definite**. This is because it is in practice an important case and also the theory is far simpler and better developed in that case.

The main example we shall take is the following :

$$(15.1) \quad A = \begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}.$$

Consider that we are asked to solve

$$Au = f.$$

The solution $u = (1, 2, -1, 1)^T$.

15.1. **Some Notation.** In this subsection, we shall discuss the matrix splitting :

Assume that $A \in \mathbb{R}^{4 \times 4}$ for simplicity.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

One can decompose A into several parts as follows :

$$\begin{aligned} A &= \begin{pmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & a_{33} & 0 \\ 0 & 0 & 0 & a_{44} \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 \\ -a_{21} & 0 & 0 & 0 \\ -a_{31} & -a_{32} & 0 & 0 \\ -a_{41} & -a_{42} & -a_{43} & 0 \end{pmatrix} \\ &- \begin{pmatrix} 0 & -a_{12} & -a_{13} & -a_{14} \\ 0 & 0 & -a_{23} & -a_{24} \\ 0 & 0 & 0 & -a_{34} \\ 0 & 0 & 0 & 0 \end{pmatrix} = D - L - L^T. \end{aligned}$$

15.2. **Classical Iterative Process.** A single step linear iterative method which uses an old approximation, u^{old} of the solution u , to produce a new approximation u^{new} , is given as follows:

$$u^{new} = u^{old} + B(f - Au^{old}).$$

Here B is a matrix which can be thought of as an approximate inverse of A . If $B = A^{-1}$, then one iteration will produce a solution.

15.2.1. *Residual correction procedure.* Let us assume that u^{old} is given as an approximate solution. The iterative process consists of the following three steps :

Given an old approximation, u^{old} , of the solution u , to produce a new approximation, u^{new} , we perform the following three steps :

- (1) Form $r^{old} = b - Au^{old}$
- (2) Solve $Ae = r^{old}$ approximately : $\hat{e} = Br^{old}$ with $B \approx A^{-1}$.
- (3) Update $u^{new} = u^{old} + \hat{e}$.

This process can be written as a single line :

$$u^{new} = u^{old} + B(f - Au^{old}).$$

Algorithm 15.1. Given $u^0 \in \mathbb{R}^n$,

$$u^{k+1} = u^k + B(f - Au^k), \quad k = 0, 1, 2, \dots,$$

We say that an iterative scheme converges if $\lim_{k \rightarrow \infty} u^k = u$ for any $u^0 \in \mathbb{R}^n$.

We have two contradicting conditions for B .

- B should be as close as A^{-1} , or B^{-1} should be as close as A
- B should be easy to compute, or B^{-1} should be easy to invert.

Example 15.1. Let $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ is the usual SPD matrix. We write $A = D - L - L^t$. The easiest approximate inverse of A are perhaps $B = D^{-1}$ or $B = (D - L)^{-1}$. The first choice is for Jacobi and the second choice leads to the Gauss-Seidel method.

15.2.2. *Matrix splitting.* Another understanding of the classical iterative process is based on the matrix splitting, Keller (1965). Namely, we split the matrix A as follows :

$$A = B^{-1} - C$$

Now consider the equation

$$Au = (B^{-1} - C)u = f$$

It can be written as follows :

$$B^{-1}u = Cu + f$$

Now we put some index :

$$B^{-1}u^{new} = Cu^{old} + f.$$

We then have

$$u^{new} = BCu^{old} + Bf = Tu^{old} + c,$$

where $T = BC$. This is the approach that the text book take.

We now note that it reduces to our first derivation.

$$\begin{aligned} u^{new} &= BCu^{old} + Bf = u^{old} - u^{old} + BCu^{old} + Bf \\ &= u^{old} + B(Cu^{old} + f - B^{-1}u^{old}) \\ &= u^{old} + B(f - (B^{-1} - C)u^{old}) \\ &= u^{old} + B(f - Au^{old}). \end{aligned}$$

15.3. Implementation. In this section, we discuss the implementation of various iterative method for the model problem

$$-u'' = -2,$$

subject to the boundary condition $u(0) = 0$ and $u(1) = 1$. As is easy to verify, the analytic solution is $u(x) = x^2$.

The discrete system corresponds to the aforementioned differential equation is given as follows:

$$\begin{pmatrix} \frac{2}{h^2} & -\frac{1}{h^2} & 0 & \cdots & \cdots & 0 \\ -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & \cdots & \cdots & 0 \\ 0 & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} & \cdots & 0 \\ \vdots & \ddots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & -\frac{1}{h^2} & \frac{2}{h^2} & -\frac{1}{h^2} \\ 0 & \cdots & \cdots & 0 & -\frac{1}{h^2} & \frac{2}{h^2} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} -2 + \frac{1}{h^2}u_a \\ -2 \\ \vdots \\ -2 \\ -2 + \frac{1}{h^2}u_b \end{pmatrix},$$

where u_a and u_b are left and right boundary conditions. For the description of algorithm, we introduce the notation of the decomposition of A as follows:

$$A = D - L - L^T.$$

Algorithm 15.2 (Richardson). For $i = 1 : N$,

$$u_i^{\ell+1} = u_i^\ell + \frac{\omega}{\rho(A)} \left(f_i - \sum_{j=1}^N a_{ij} u_j^\ell \right)$$

This corresponds to

$$u^{\ell+1} = u^\ell + \frac{\omega}{\rho(A)} (f - Au^\ell).$$

Algorithm 15.3 (Jacobi). For $i = 1 : N$,

$$u_i^{\ell+1} = u_i^\ell + a_{ii}^{-1} \left(f_i - \sum_{j=1}^N a_{ij} u_j^\ell \right)$$

This corresponds to

$$u^{\ell+1} = u^\ell + D^{-1}(f - Au^\ell)$$

Algorithm 15.4 (Gauss-Seidel (forward)). For $i = 1 : N$,

$$u_i^{\ell+1} = u_i^\ell + a_{ii}^{-1} \left(f_i - \sum_{j=1}^{i-1} a_{ij} u_j^{\ell+1} - \sum_{j=i}^N a_{ij} u_j^\ell \right)$$

This corresponds to

$$u^{\ell+1} = u^\ell + (D - L)^{-1}(f - Au^\ell).$$

(Why?)

Example 15.2. Consider the following equation:

$$(15.2) \quad Au = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Note that $u = (1, 1)^T$ is the solution. Starting an initial guess $u^0 = (0, 0)^T$, compute u^2 by applying the Jacobi and Gauss-Seidel methods.

Note that the aforementioned G-S method is equivalent to

$$\begin{aligned}
 (D - L)u^{\ell+1} &= (D - L)u^\ell + f - Au^\ell \\
 \implies Du^{\ell+1} &= Lu^{\ell+1} + (D - L)u^\ell + f - (D - L - L^T)u^\ell \\
 \implies Du^{\ell+1} &= Du^\ell - Du^\ell + Lu^{\ell+1} + f - (-L^T)u^\ell \\
 \implies Du^{\ell+1} &= Du^\ell + f - (Du^\ell - Lu^{\ell+1} - L^T u^\ell) \\
 \implies u^{\ell+1} &= u^\ell + D^{-1}(f - (Du^\ell - Lu^{\ell+1} - L^T u^\ell))
 \end{aligned}$$

Note that

$$(D - L)u^\ell = \sum_{j=i}^N a_{ij}u_j^\ell$$

and

$$-L^T u^{\ell+1} = \sum_{j=1}^{i-1} a_{ij}u_j^{\ell+1}.$$

Remark 15.1. As is always, we need to construct the stopping criterion. We propose the following as the good stopping criterion:

$$\frac{\|f - Au^k\|}{\|f - Au^0\|} < tol$$

which is called the relative residual error.

15.4. Convergence of the iterative method. We first recall that

Theorem 15.1. *The following statements are equivalent :*

- (1) A is a convergent matrix.
- (2) $\rho(A) < 1$
- (3) $\lim_{n \rightarrow \infty} A^n x = 0$.

We note that the error satisfies the following equation that

$$u - u^k = (I - BA)^k(u - u^0),$$

from which we obtain the following convergence result :

Lemma 15.1. *The iterative scheme converges if and only if*

$$\rho(I - BA) < 1.$$

In fact, we have the following relation that for any natural norm $\|\cdot\|$,

$$\begin{aligned}\|u - u^k\| &= \|(I - BA)^k(u - u^0)\| \leq \|(I - BA)^k\| \|u - u^0\| \\ &\leq \|I - BA\|^k \|u - u^0\|\end{aligned}$$

Therefore, the convergence rate of the method with respect to $\|\cdot\|$ is $\|I - BA\|$ and the convergence is guaranteed if $\|I - BA\| < 1$.

We consider a simple example called the Richardson iteration, which is perhaps the simplest possible iterative method:

$$u^{k+1} = u^k + \frac{\omega}{\rho(A)}(f - Au^k), \quad k = 1, 2, \dots,$$

To guarantee the convergence of the method, we need to choose ω appropriately. Note that if $\{\lambda_1, \dots, \lambda_n\}$ are the eigenvalues of A , then

$$\sigma(I - BA) = \left\{1 - \omega \frac{\lambda_i}{\rho(A)} : i = 1, \dots, n\right\}.$$

For the convergence, we should have that

$$|1 - \omega \lambda_i / \rho(A)| < 1$$

for all $i = 1, \dots, n$. We then conclude that $0 < \omega < 2/\rho(A)$. Furthermore, the convergence rate is given by

$$(15.3) \quad \delta = \max\{|1 - \omega \lambda_{\min}/\rho(A)|, |1 - \omega \lambda_{\max}/\rho(A)|\}$$

Therefore, the optimal choice of ω can be made if

$$|1 - \omega \lambda_{\min}/\rho(A)| = |1 - \omega \lambda_{\max}/\rho(A)|$$

Namely, we have

$$\omega = \frac{2}{\lambda_{\min} + \lambda_{\max}}.$$

However, it is difficult to check the spectral radius is less than one since finding eigenvalues are difficult in general. We rather ask if $(B^{-1})^T + B^{-1} - A$ is symmetric and positive definite. If so, we have convergence.

Theorem 15.2. *If the matrix $(B^{-1})^T + B^{-1} - A$ is symmetric and positive definite, then the method is convergent with respect to A norm, i.e., $\|\cdot\|_A = (A\cdot, \cdot)^{1/2}$.*

Proof. We consider the A -norm given by

$$(\cdot, \cdot)_A = (A\cdot, \cdot).$$

Now we have the following relation :

$$\|x\|_A^2 - \|(I - BA)x\|_A^2 = (\bar{B}Ax, x)_A = (\bar{B}Ax, Ax),$$

where $\bar{B} = B^T + B - B^T AB = B^T(B^{-1})^T + B^{-1} - A$ or

$$I - \bar{B} = (I - B^T A)(I - BA).$$

Note that if \bar{B} is symmetric and positive definite, then, we have that for some $\mu > 0$, a constant,

$$(\bar{B}Ax, Ax) \geq \mu(Ax, Ax)$$

The difficult part is to show that for all x , there exists a constant $\nu > 0$ such that $(Ax, Ax) \geq \nu(Ax, x)$. For now we assume it is true, then we have

$$\|x\|_A^2 - \|(I - BA)x\|_A^2 = (\bar{B}Ax, x)_A = (\bar{B}Ax, Ax) \geq \delta(Ax, x) = \delta\|x\|_A^2,$$

where $\delta = \mu \cdot \nu$. Then we have the convergence since, due to the aforementioned inequality,

$$(1 - \delta)\|x\|_A^2 \geq \|(I - BA)x\|_A^2, \quad \forall x.$$

This means that

$$\|(I - BA)\|_A = \max_{x \neq 0} \frac{\|(I - BA)x\|_A}{\|x\|_A} \leq \sqrt{1 - \delta} < 1$$

and therefore, we have the convergence since

$$\|u - u^\ell\|_A \leq \|I - BA\|_A^\ell \|u - u^0\|_A \leq \sqrt{1 - \delta}^\ell \|u - u^0\|_A \rightarrow 0 \text{ as } \ell \rightarrow \infty.$$

The conclusion can then be made, i.e., if $(B^{-1})^T + B^{-1} - A$ is symmetric and positive definite, we have the convergence.

We finally, show that $(Ax, Ax) \geq \nu(Ax, x)$. To show, let $A = Q\Lambda Q^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ with $0 < \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ and Q is the orthogonal matrix and introduce $A^{1/2} = Q\Lambda^{1/2}Q^T$, where $\Lambda^{1/2} = \text{diag}(\lambda_1^{1/2}, \dots, \lambda_n^{1/2})$. Then it is easy to see that $A = A^{1/2}A^{1/2}$ and $A^{1/2}$ is symmetric and positive definite.

Now, observe that since $(Au, u) \geq \lambda_1(u, u)$ for all u ,

$$\begin{aligned} (Ax, Ax) &= (A^{1/2}A^{1/2}x, A^{1/2}A^{1/2}x) \\ &= (A^{1/2}A^{1/2}A^{1/2}x, A^{1/2}x) \\ &= (AA^{1/2}x, A^{1/2}x) \\ &\geq \lambda_1(A^{1/2}x, A^{1/2}x) \\ &= \lambda_1(A^{1/2}A^{1/2}x, x) = \lambda_1(Ax, x) \end{aligned}$$

□

15.5. Error Bounds and Iterative Refinement. As in all other iterative process for the solution of $Au = f$, we need to construct the stopping criterion for the algorithm to complete and also the initial guess.

In general, for the initial guess, we use zero vector and the relative residual discussed in the previous section is used for the stopping criterion:

$$\frac{\|f - Au^\ell\|}{\|f - Au^0\|} < \text{tol.}$$

Note that if u^k be the k -th iterate. The quantity $r^k = f - Au^k$ is called the k -th residual. It is our intuition that if

$$\frac{\|r^k\|}{\|r^0\|} = \frac{\|f - Au^k\|}{\|f - Au^0\|}$$

is small, then u^k is close to u . In fact, in most of the practical problem, it can be a good stopping criterion, but for some special problem, it may not be.

The following theorem says that if $\|A\|\|A^{-1}\|$ is small, the relative residual can be a good stopping criterion.

Theorem 15.3.

$$\frac{\|x - x^k\|}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\|r\|}{\|f\|}.$$

Proof. Since $Au = f$, we have $\|f\| = \|Au\| \leq \|A\|\|u\|$. Therefore,

$$\frac{1}{\|x\|} \leq \frac{\|A\|}{\|f\|}.$$

and

$$\|x - x^k\| = \|A^{-1}A(x - x^k)\| \leq \|A^{-1}\|\|r\|.$$

□

Definition 15.1. The condition number of the nonsingular matrix A relative to a norm $\|\cdot\|$ is

$$K(A) = \|A\|\|A^{-1}\|.$$

The following example shows that the quantity $\|A\|\|A^{-1}\|$ can be very large for some special matrix.

Example 15.3. Consider the solution of the following equation:

$$Au = \begin{pmatrix} 1 & 2 \\ 1.0001 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} 3 \\ 3.0001 \end{pmatrix}$$

Note that $\|A\|_\infty = 3.0001$ and since

$$A^{-1} = \begin{pmatrix} -10000 & 10000 \\ 5000.5 & -5000 \end{pmatrix}$$

$\|A^{-1}\|_\infty = 20000$. Therefore, $K(A) = 60002$.

The condition number also measures the sensitivity of the solution to perturbations in A and also f . We first consider the perturbation of f .

If we perturb f to $\tilde{f} = f + \delta f$, then this will lead to a change in u to \tilde{u} defined by $A\tilde{u} = \tilde{f}$. If the relative error

$$\frac{\|\tilde{f} - f\|}{\|f\|} = \frac{\|\delta f\|}{\|f\|}$$

is small (measured in some vector norm), what about the relative error in u :

$$\frac{\|\tilde{u} - u\|}{\|u\|} = \frac{\|A^{-1}(\tilde{f} - f)\| \|Au\|}{\|f\| \|x\|} \leq K(A) \frac{\|\tilde{f} - f\|}{\|f\|}$$

Theorem 15.4. Suppose A is nonsingular and $\|A^{-1}\delta A\| < 1$ and \tilde{u} is the solution to $(A + \delta A)u = f$. Then $A + \delta A$ is nonsingular and

$$\|(A + \delta A)^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\delta A\|}.$$

Moreover,

$$\frac{\|u - \tilde{u}\|}{\|\tilde{u}\|} \leq K(A) \frac{\|\delta A\|}{\|A\|}.$$

Proof. Note that $I - E$ is invertible if $\|E\| < 1$ and

$$(I - E)^{-1} = I + E + E^2 + E^3 + E^4 \dots$$

Moreover,

$$\|(I - E)^{-1}\| \leq \sum_{j=0}^{\infty} \|E\|^j = \frac{1}{1 - \|E\|}.$$

Note then that

$$\|(A + \delta A)^{-1}\| = \|(I + A^{-1}\delta A)^{-1}A^{-1}\| \leq \|A^{-1}\| \frac{1}{1 - \|A^{-1}\delta A\|}.$$

Note that

$$u - \tilde{u} = A^{-1}\delta A\tilde{u}.$$

Therefore, we have that

$$(15.4) \quad \frac{\|u - \tilde{u}\|}{\|\tilde{u}\|} \leq \|A^{-1}\| \|\delta A\| \leq K(A) \frac{\|\delta A\|}{\|A\|}.$$

□

There is no way to accurately solve the system of equation that is ill-conditioned.

16. DISCRETE LEAST SQUARE METHODS

Assume that a set of data is given as follows:

$$(16.1) \quad \{(x_i, y_i) : i = 1, 2, \dots, m\}.$$

Now we consider data fitting problem by an algebraic polynomial

$$P_n(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n.$$

By data fitting, we mean that we choose coefficients so that the given data y_i at x_i is "close" to the values of $P_n(x_i)$ for all i . It is easy to see that if the degree n is greater than $m-1$, then since we can exactly fit the data. However, in general, there are so many data for which it should be expensive to use higher order polynomials and it is preferable to use less degree of polynomials.

We need the notion of "closeness" and it can be given in many different ways. For example, one may wish to measure such a closeness by the following quantity (maximum norm):

$$E_\infty = \max_{1 \leq i \leq m} |y_i - P_n(x_i)|.$$

Another choices can be with the following:

$$E_1 = \sum_{i=1}^m |y_i - P_n(x_i)|$$

or

$$E_2 = \sum_{i=1}^m |y_i - P_n(x_i)|^2.$$

All of these can be a tool to measure how good is your fitting for the data by the polynomial. In the finite dimensional space, all of the measure introduced above are equivalent. Namely, there exists some constants c_1 and c_2 such that the following holds true:

$$c_1 E_\alpha \leq E_\beta \leq c_2 E_\alpha,$$

where α and β can be 1, 2 or ∞ .

The *least squares* approach is to determine the best approximating line by minimizing the closeness measure E_2 . We consider the best least squares line to a collection of data $\{x_i, y_i\}_{i=1}^m$ involves minimizing the total error,

$$E_2(a_0, a_1) = \sum_{i=1}^m [y_i - (a_1x_i + a_0)]^2.$$

For a minimum to occur, we need

$$(16.2) \quad \frac{\partial}{\partial a_0} E_2(a_0, a_1) = \frac{\partial}{\partial a_1} E_2(a_0, a_1) = 0$$

We then obtain two equations and two unknowns to be determined naturally.

Example 16.1. Find the best least squares constant for the data $(1, 1)$ and $(2, 2)$.

The equation derived above, (16.2) is called the normal equation.

16.1. Derivation of Normal Equation. To formulate some general abstract framework, we recast the above problem as the general best approximation problem. Let V be an inner product space with an inner product (\cdot, \cdot) and an induced norm $\|\cdot\| = (\cdot, \cdot)^{1/2}$. We now consider the subspace P of V .

Now we consider the following question:

- For a given $f \in V$, does there exist $p \in P$ minimizing $\|f - p\|$?
- Could there exist more than one minimizer ?
- Can the (or a) minimizer be computed ?
- What can we say about the error ?

The best discrete least squares approximation can be characterized as follows :

Theorem 16.1. Let V be an inner product space, P be a finite dimensional subspace, and $f \in V$, then there exists a unique $p \in P$ minimizing $\|f - p\|$ over P . It is characterized by the normal equations,

$$(p, q) = (f, q), \quad \forall q \in P.$$

Proof. To obtain the characterization, note that

$$\|f - p + \epsilon q\|^2 = \|f - p\|^2 + 2\epsilon(f - p, q) + \epsilon^2\|q\|^2$$

achieves its minimum at $\epsilon = 0$. If p and p^* are both best approximations, then we have

$$(p - p^*, q) = 0 \quad \forall q \in P.$$

This means that $p = p^*$. □

In the course of the proof, we showed that the normal equations admit the unique solution. To obtain the solution, we select a basis ϕ_1, \dots, ϕ_n of P , we then set

$$p = \sum_{i=1}^n a_i \phi_i.$$

Now

$$(16.3) \quad \sum_{j=1}^n (\phi_j, \phi_i) a_j = (f, \phi_i).$$

The equation (16.3) is oftentimes called the normal equation. This is clearly a nonsingular matrix equation.

16.1.1. *Discrete Least Squares by Polynomials.* Now, we get back to our best least square fitting problem by polynomials. To derive the normal equation explicitly, we need to specify the spaces V and P , an inner product on V and also the basis for P . Given distinct points $\{x_i\}_{i=1}^m$ in the interval $[a, b]$. We consider the space of polynomials determined by completely by the values at the given points. Namely,

$$V = \text{span}\{1, x, x^2, \dots, x^{m-1}\}.$$

Define an inner product on V by the following:

$$(f, g) = \sum_{i=1}^m f(x_i)g(x_i), \quad f, g \in V.$$

We also define an induced norm $\|f\|^2 = (f, f)$. Check if it is a norm.

We now consider the subspace of V denoted by P of the space of polynomial of degree n with $n \leq m - 1$. We can then interpret our discrete least squares by the polynomials as to find the polynomial $p \in P$ such that

$$\min_{p \in P} \|f - p\|^2.$$

It is shown that the unique existence of the minimizer denoted by p and it satisfies the following characterization:

$$(16.4) \quad (p, q) = (f, q) \quad \forall q \in P.$$

We need to have the basis. Because we wish to find a_0, a_1, \dots, a_n which are coefficients of polynomials x^i with $i = 0, 1, \dots, n$. We simply consider the set of polynomials

$$P = \text{span}\{1, x, x^2, \dots, x^n\}.$$

We now set $p = \sum_{i=0}^n a_i x^i$. Using the equation (16.4), we obtain the following $n + 1$ equation:

$$(16.5) \quad \sum_{i=0}^n a_i (x^i, x^k) = (f, x^k) \quad \forall k = 0, \dots, n.$$

In a matrix format, we have the following linear system of equations:

$$(16.6) \quad \begin{pmatrix} (x^0, x^0) & (x^1, x^0) & (x^2, x^0) & \cdots & (x^n, x^0) \\ (x^0, x^1) & (x^1, x^1) & (x^2, x^1) & \cdots & (x^n, x^1) \\ (x^0, x^2) & (x^1, x^2) & (x^2, x^2) & \cdots & (x^n, x^2) \\ \vdots & \cdots & \cdots & \cdots & \cdots \\ (x^0, x^n) & (x^1, x^n) & (x^2, x^n) & \cdots & (x^n, x^n) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} (f, x^0) \\ (f, x^1) \\ (f, x^2) \\ \vdots \\ (f, x^n) \end{pmatrix},$$

where $x^0 = 1$,

$$(x^i, x^j) = \sum_{k=1}^m x_k^i x_k^j \quad \text{and} \quad (f, x^i) = \sum_{k=1}^m f(x_k) x_k^i, \quad \forall i, j = 0, \dots, n$$

16.2. Another look. Now, we shall look at the above problem in a different point of view. Given data $b = (f(x_1), f(x_2), \dots, f(x_m))$ and $Ay = (p(x_1), p(x_2), \dots, p(x_m))$, with $y = (a_0, a_1, \dots, a_n)$, then A can be explicitly written as follows:

$$(16.7) \quad A = \begin{pmatrix} 1 & x_1 & \cdots & x_1^n \\ 1 & x_2 & \cdots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & \cdots & x_m^n \end{pmatrix} \in \mathbb{R}^{m \times (n+1)}$$

The polynomial least square data fitting problem can be recast into the following equation : Find $y \in \mathbb{R}^{n+1}$ such that

$$\min_{y \in \mathbb{R}^{n+1}} \|b - Ay\|_{\ell^2},$$

Now the characterization of the minimizer for the problem leads to

$$(Ay, Ax) = (b, Ax) \quad \forall x \in \mathbb{R}^n.$$

From this, we have

$$(A^T Ay, x) = (A^T b, x), \quad \forall x \in \mathbb{R}^n.$$

Therefore, we have the following equation:

$$(16.8) \quad A^T Ay = A^T b.$$

The equation (16.8) is called the normal equation of $Ay = b$. One can in particular show that the matrix given in the equation (16.6) is the same with $A^T A$ and the right hand side is equal to $A^T b$.

17. THE SOLUTION METHOD FOR THE NORMAL EQUATION : QR FACTORIZATION AND GRAM SCHMIDT PROCESS

In this section, we discuss how to solve the normal equation of $Ax = b$ in general, where $A \in \mathbb{R}^{m \times n}$ with $m \geq n$.

17.1. QR decomposition by Gram-Schmidt process. The method that shall be taken is so-called the QR decomposition.

Theorem 17.1. *Let A be m by n with $m \geq n$. Suppose that A has full column rank. Then A can be QR decomposed. More precisely, there exist a unique m by n*

orthogonal matrix Q with $(Q^T Q = I)$ and a unique n by n upper triangular matrix R with positive diagonal $r_{ii} > 0$ such that

$$A = QR.$$

This is called the *QR decomposition* of A .

17.1.1. *Illustrations.* The algorithm is indeed contained in the proof of the above theorem.

The main tool to decompose A into QR is as discussed the Gram-Schmidt process. We shall illustrate the procedure by taking three by three matrices.

Given a matrix $A = [a_1, a_2, a_3]$ with a_i 's are linearly independent. The Gram-Schmidt process makes q_1, q_2 and q_3 so that $(q_i, q_j) = q_i^T q_j = 0$ for $i \neq j$. Namely, q_i 's are mutually orthogonal.

We first set $\tilde{q}_1 = a_1$ and define

$$(17.1) \quad q_1 = \frac{1}{\|\tilde{q}_1\|} \tilde{q}_1.$$

We then look for the orthogonal vector q_2 to q_1 in the plane spanned by a_2 and q_1 . Set first

$$\tilde{q}_2 = a_2 - \alpha q_1.$$

Using the requirement that $\tilde{q}_2 \cdot q_1 = 0$, we obtain that $0 = q_1 \cdot \tilde{q}_2 = q_1 \cdot a_2 - \alpha$, hence

$$\alpha = q_1 \cdot a_2.$$

We then conclude that $\tilde{q}_2 = a_2 - (q_1, a_2)q_1$. Set now that

$$(17.2) \quad q_2 = \frac{1}{\|\tilde{q}_2\|} (a_2 - (q_1, a_2)q_1).$$

Finally, we look for orthogonal vector of both q_1 and q_2 from the space spanned by a_3 and q_1 and q_2 .

$$\tilde{q}_3 = a_3 - \beta q_1 - \gamma q_2.$$

Here, β and γ can be determined by the following two requirements:

$$0 = q_1 \cdot \tilde{q}_3 = q_1 \cdot a_3 - \beta q_1 \cdot q_1 - \gamma q_1 \cdot q_2$$

and

$$0 = q_2 \cdot \tilde{q}_3 = q_2 \cdot a_3 - \beta q_2 \cdot q_1 - \gamma q_2 \cdot q_2.$$

Namely,

$$\beta = q_1 \cdot a_3 \text{ and } \gamma = q_2 \cdot a_3.$$

Finally, we set that

$$q_3 = \frac{1}{\|\tilde{q}_3\|} (a_3 - (q_1, a_3)q_1 - (q_2, a_3)q_2).$$

We now write it in terms of a_1, a_2 and a_3 to obtain that

$$\begin{aligned} a_1 &= \|\tilde{q}_1\|q_1 \\ a_2 &= \|\tilde{q}_2\|q_2 + (q_1, a_2)q_1 \\ a_3 &= \|\tilde{q}_3\|q_3 + (q_1, a_3)q_1 + (q_2 \cdot a_3)q_2. \end{aligned}$$

In the matrix form

$$(17.3) \quad A = [a_1, a_2, a_3] = [q_1, q_2, q_3] \begin{pmatrix} \|\tilde{q}_1\| & (q_1, a_2) & (q_1, a_3) \\ 0 & \|\tilde{q}_2\| & (q_2, a_3) \\ 0 & 0 & \|\tilde{q}_3\| \end{pmatrix}.$$

We can deduce the general form, namely, if $A = [a_1, a_2, \dots, a_{n-1}, a_n]$, with a_i 's being linearly independent, then we can decompose A into QR as follows:

$$A = [q_1, q_2, \dots, q_n] \begin{pmatrix} \|\tilde{q}_1\| & (q_1, a_2) & (q_1, a_3) & \cdots & (q_1, a_n) \\ 0 & \|\tilde{q}_2\| & (q_2, a_3) & \cdots & (q_2, a_n) \\ 0 & 0 & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \cdots & \|\tilde{q}_n\| \end{pmatrix},$$

where $\tilde{q}_j = a_j - \sum_{k=1}^{j-1} (q_k, a_j)q_k$.

Example 17.1. Apply QR decomposition of the following matrix.

$$(17.4) \quad A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

The solution is given here.

$$q_1 = \frac{1}{\|\tilde{q}_1\|} \tilde{q}_1 = \begin{pmatrix} 2/\sqrt{5} \\ -1/\sqrt{5} \\ 0 \end{pmatrix} \quad \text{with} \quad \tilde{q}_1 = a_1 = \begin{pmatrix} 2 \\ -1 \\ 0 \end{pmatrix}$$

Note that

17.2. The solution by QR factorization. In this section, we discuss how to apply QR decomposition to solve the linear system of equation. Recall the normal equation is given by

$$A^T A x = A^T b$$

We decompose QR for A. Then,

$$(QR)^T (QR)x = (QR)^T b \Rightarrow R^T Q^T QRx = R^T Q^T b \Rightarrow Rx = Q^T b$$

Hence the solution x is given by

$$x = R^{-1} Q^T b.$$

Example 17.2. Solve the following minimization problem by QR factorization :

$$(17.5) \quad \min_{x \in \mathbb{R}^2} \|b - Ax\|_{\ell^2},$$

where

$$(17.6) \quad A = \begin{pmatrix} 1 & 0 \\ -1 & 2 \\ 0 & -1 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}.$$

Note that the solution $x = (1, 1)^T$.

18. A SHORT INTRODUCTION TO DISCRETE FOURIER APPROXIMATION AND FAST FOURIER TRANSFORMATION (FFT)

In this section, we shall discuss the approximation of the continuous function by trigonometric polynomials.

18.1. Continuous least squares approximations by trigonometric polynomials. Let V be an inner product space defined as follows:

$$V = \{f : \mathbb{R} \mapsto \mathbb{R} : f \in C[-\pi, \pi]\},$$

in which the inner product (\cdot, \cdot) is given through

$$(18.1) \quad (f, g) = \int_{-\pi}^{\pi} fg \, dx, \quad \forall f, g \in V$$

and the induced norm is defined by

$$\|f\| = (f, f)^{1/2} = \left(\int_{-\pi}^{\pi} f(x)^2 dx \right)^{1/2}.$$

We introduce a subspace W of V defined by

$$W = \text{span}\{\psi_0, \psi_1, \dots, \psi_{2n-1}\},$$

where

$$\psi_0(x) = 1, \psi_k(x) = \cos kx, \text{ for each } k = 1, 2, \dots, n,$$

and

$$\psi_{n+k}(x) = \sin kx, \text{ for each } k = 1, 2, \dots, n-1.$$

The space W is called the set of *trigonometric polynomials* of degree less than or equal to n . Note that some text includes the function $\psi_{2n} = \sin nx$.

We shall consider the following minimization problem : for a given $f \in V$,

$$(18.2) \quad \min_{q \in W} \|f - q\|.$$

Note that this type of problem is often called the *continuous least squares* approximation by functions in W . Recall the characterization of the minimizer implies the following equation holds true: Assume S_n is the minimizer, then

$$(S_n, q) = (f, q), \quad \forall q \in W.$$

Note that in general, S_n can be written as follows:

$$\begin{aligned} S_n &= a_0 + a_n \cos nx + \sum_{k=1}^{n-1} (a_k \cos kx + b_k \sin kx) \\ &= a_0 \psi_0 + a_n \psi_n + \sum_{k=1}^{n-1} (a_k \psi_k + b_k \psi_{n+k}) \\ &= \sum_{k=0}^{2n-1} c_k \psi_k, \end{aligned}$$

where $c_k = a_k$ for $k = 0, \dots, n$ and $c_{n+k} = b_k$ for $k = 1, \dots, n-1$.

Therefore, we need to determine $2n$ coefficients and the normal equation leads to exactly $2n$ equations:

$$(S_n, \psi_i) = (f, \psi_i) \quad \forall i = 0, \dots, 2n-1.$$

We may write it as the linear system as before :

$$SC = F,$$

where

$$\mathcal{S} = \begin{pmatrix} (\psi^0, \psi^0) & (\psi^1, \psi^0) & (\psi^2, \psi^0) & \dots & (\psi^{2n-1}, \psi^0) \\ (\psi^0, \psi^1) & (\psi^1, \psi^1) & (\psi^2, \psi^1) & \dots & (\psi^{2n-1}, \psi^1) \\ (\psi^0, \psi^2) & (\psi^1, \psi^2) & (\psi^2, \psi^2) & \dots & (\psi^{2n-1}, \psi^2) \\ \vdots & \dots & \dots & \dots & \dots \\ (\psi^0, \psi^{2n-1}) & (\psi^1, \psi^{2n-1}) & (\psi^2, \psi^{2n-1}) & \dots & (\psi^{2n-1}, \psi^{2n-1}) \end{pmatrix}$$

$$C = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-1} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} (f, \psi^0) \\ (f, \psi^1) \\ (f, \psi^2) \\ \vdots \\ (f, \psi^{2n-1}) \end{pmatrix}.$$

One crucial property of the basis ψ^i is that they are orthogonal each other with respect to the inner product (\cdot, \cdot) . Namely,

$$(\psi_i, \psi_j) = 0 \quad \text{if } i \neq j.$$

The reason for this is from the following relation:

$$\begin{aligned}\sin(kx) \sin(jx) &= \frac{1}{2}[\cos(kx - jx) - \cos(kx + jx)], \\ \cos(kx) \cos(jx) &= \frac{1}{2}[\cos(kx - jx) + \cos(kx + jx)], \\ \sin(kx) \cos(jx) &= \frac{1}{2}[\sin(kx - jx) + \sin(kx + jx)].\end{aligned}$$

Furthermore, we can easily compute (ψ_i, ψ_i) for all $i = 0, \dots, 2n - 1$. Observe that

$$\begin{aligned}\int_{-\pi}^{\pi} \psi_0 \psi_0 dx &= 2\pi. \\ \int_{-\pi}^{\pi} \psi_k \psi_k dx &= \int_{-\pi}^{\pi} \frac{1}{2} dx = \pi.\end{aligned}$$

Therefore, we have

$$C = S^{-1}F.$$

Namely,

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) dx$$

and for all $k = 1, \dots, 2n - 1$, we have

$$c_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \psi_k dx.$$

Definition 18.1. $\lim_{n \rightarrow \infty} S_n$ is called the Fourier series of f .

Example 18.1. Find the continuous least squares trigonometric polynomial $S_2(x)$ for $f(x) = |x|$ on $[-\pi, \pi]$.

Set $S_2 = \sum_{k=0}^{2*2-1} c_k \psi_k$, where $\psi_0 = 1$, $\psi_k = \cos kx$ for $k = 1, 2$ and $\psi_k = \sin kx$ for $k = 1$.

$$\begin{aligned}c_0 &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |x| dx = \frac{\pi}{2} \\ c_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} |x| \cos x dx = \frac{2}{\pi k^2} [(-1)^k - 1] \quad \text{for } k = 1, 2 \\ c_k &= \frac{1}{\pi} \int_{-\pi}^{\pi} |x| \sin(kx) dx = 0, \quad \text{for } k = 3.\end{aligned}$$

Therefore, we have

$$S_2(x) = \frac{\pi}{2} - \frac{\pi}{4} \cos x$$

Example 18.2. Find the continuous least squares trigonometric polynomial $S_2(x)$ for $f(x) = x^2$ on $[-\pi, \pi]$.

18.2. discrete least squares approximation. In this section, we study the discrete least squares method. One may think that the function f is not known everywhere but known at some specific nodes, i.e., a finite set of data is available. More precisely, one has the following $2m$ data set :

$$\{(x_i, f(x_i))\}_{i=0}^{2m-1} = \{(x_i, y_i)\}_{i=0}^{2m-1}.$$

For simplicity, we shall assume that x_i 's are in $[-\pi, \pi]$ and moreover,

$$x_i = -\pi + \frac{i}{m}\pi.$$

To make our discussion more rigorous, we may need to redefine the space V as follows :

$$V_h \subset V,$$

in which V_h is the set of functions that are completely determined by the values specified at $2m$ nodes. We can then define the corresponding discrete ℓ^2 inner product on V_h , namely,

$$(f, g) = \sum_{i=0}^{2m-1} f(x_i)g(x_i) \quad \forall f, g \in V_h.$$

This shall naturally, induce a norm on V_h :

$$\|f\| = (f, f)^{1/2} \quad \forall f \in V_h.$$

Now we shall consider the following minimization problem:

$$\min_{S_n \in W_h} \|f - S_n\|,$$

where $m > n$ in general (**Note :** We assume $m > n$ so that $2n \neq 2m$) and

$$S_n = \sum_{k=0}^{2n-1} c_k \psi_k.$$

Note that the norm $\|\cdot\|$ is nothing else than the discrete ℓ^2 norm, i.e.,

$$\|f - S_n\|^2 = \sum_{k=0}^{2m-1} (f(x_k) - S_n(x_k))^2.$$

To find the coefficients c_k for $k = 0, \dots, 2n-1$, we consider the characterization of the minimizer, namely, the minimizer S_n satisfies the following characterization property:

$$(S_n, \psi_k) = (f, \psi_k), \quad \forall k = 0, \dots, 2n-1.$$

The solution to the above problem is called the discrete least squares problem since the evaluation of the inner product is made by the finite set of data sets. The determination of the constants is simplified by the fact that the set $\{\psi_0, \dots, \psi_{2n-1}\}$ is orthogonal with respect to summation over the equally spaced points $\{x_i\}_{i=0}^{2m-1}$ in $[-\pi, \pi]$. By this we mean that for $k \neq \ell$, then

$$(\psi_k, \psi_\ell) = \sum_{j=0}^{2m-1} \psi_k(x_j) \psi_\ell(x_j) = 0.$$

If $k = \ell$, then we have

$$(\psi_k, \psi_k) = m.$$

More precisely, we have

Lemma 18.1. *If an integer r is not a multiple of $2m$, then*

$$\sum_{j=0}^{2m-1} \cos(rx_j) = 0 = \sum_{j=0}^{2m-1} \sin(rx_j).$$

Moreover, if r is not a multiple of m , then

$$\sum_{j=0}^{2m-1} (\cos(rx_j))^2 = m = \sum_{j=0}^{2m-1} (\sin(rx_j))^2.$$

Proof. The idea is to consider the Euler's identity:

$$\exp^{iz} = \cos z + i \sin z.$$

and

$$1 + r + r^2 + \dots + r^n = \frac{1 - r^{n+1}}{1 - r}.$$

Note that

$$\begin{aligned} \sum_{j=0}^{2m-1} \cos(rx_j) + i \sin(rx_j) &= \sum_{j=0}^{2m-1} \exp^{irx_j} \\ &= \sum_{j=0}^{2m-1} \exp^{ir(-\pi + \frac{j}{m}\pi)} \\ &= \exp^{-ir\pi} \sum_{j=0}^{2m-1} \exp^{ir(\frac{j}{m}\pi)} = \exp^{-ir\pi} \sum_{j=0}^{2m-1} K^j, \\ &= \exp^{-ir\pi} \frac{1 - K^{2m}}{1 - K} = 0 \end{aligned}$$

where $K = \exp^{ir(\frac{\pi}{m})}$. Note that

$$K^{2m} = \exp^{ir\frac{2m}{m}\pi} = 1.$$

If r is not a multiple of m , then we have that from the trigonometric identity :

$$\sum_{j=0}^{2m-1} (\cos(rx_j))^2 = \sum_{j=0}^{2m-1} \frac{1}{2}(1 + \cos(2rx_j)) = m.$$

□

By a simple application of the Theorem, we obtain that

Theorem 18.1. *The constants in the summation*

$$S_n(x) = a_0 + a_n \cos(nx) + \sum_{k=1}^{n-1} (a_k \cos(kx) + b_k \sin(kx)).$$

that solves the following minimization problem :

$$E_\ell = \min_{q \in W_h} \|f - q\|_{\ell^2}$$

is given by $a_0 = \frac{1}{2m} \sum_{j=0}^{2m-1} y_j$, and

$$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos(kx_j) \quad \text{for each } k = 1, \dots, n,$$

$$b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin(kx_j) \quad \text{for each } k = 1, \dots, n-1.$$

Proof. Set $S_n = \sum_{j=0}^{2m-1} c_j \psi_j$, where $c_k = a_k$ for $k = 0, \dots, n$, $c_{n+k} = b_k$ for $k = 1, \dots, n-1$ and $\psi_k = \cos(kx)$ for $k = 0, \dots, n$ and $\psi_{n+k} = \sin(kx)$ for $k = 1, \dots, n-1$. Since S_n satisfies the following characterization property:

$$(S_n, q) = (f, q), \quad \forall q \in W_h,$$

where $f(x_i) = y_i$ for $i = 0, \dots, 2m-1$, the normal equation can be formed as follows: We may write it as the linear system as before :

$$SC = F,$$

where

$$S = \begin{pmatrix} (\psi^0, \psi^0) & (\psi^1, \psi^0) & (\psi^2, \psi^0) & \dots & (\psi^{2n-1}, \psi^0) \\ (\psi^0, \psi^1) & (\psi^1, \psi^1) & (\psi^2, \psi^1) & \dots & (\psi^{2n-1}, \psi^1) \\ (\psi^0, \psi^2) & (\psi^1, \psi^2) & (\psi^2, \psi^2) & \dots & (\psi^{2n-1}, \psi^2) \\ \vdots & \dots & \dots & \dots & \dots \\ (\psi^0, \psi^{2n-1}) & (\psi^1, \psi^{2n-1}) & (\psi^2, \psi^{2n-1}) & \dots & (\psi^{2n-1}, \psi^{2n-1}) \end{pmatrix}$$

$$C = \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{2n-1} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} (f, \psi^0) \\ (f, \psi^1) \\ (f, \psi^2) \\ \vdots \\ (f, \psi^{2n-1}) \end{pmatrix},$$

where the inner product is the discrete ℓ^2 inner product at nodes $\{x_i\}_{i=0}^{2m-1}$.

We shall first see that

$$(\psi_0, \psi_0) = \sum_{j=0}^{2m-1} 1 = 2m.$$

Now for $k = 1, \dots, n-1$, we have

$$\begin{aligned} (\psi_k, \psi_k) &= \sum_{j=0}^{2m-1} \cos(kx_j) \cos(kx_j) \\ &= \sum_{j=0}^{2m-1} \frac{1}{2} (1 + \cos(2kx_j)) = m. \end{aligned}$$

Furthermore, for $k = 1, \dots, n-1$, we have

$$\begin{aligned} (\psi_{n+k}, \psi_{n+k}) &= \sum_{j=0}^{2m-1} \sin(kx_j) \sin(kx_j) \\ &= \sum_{j=0}^{2m-1} \frac{1}{2} (1 - \cos(2kx_j)) = m. \end{aligned}$$

Now, for $k = n$,

$$(\psi_n, \psi_n) = \sum_{j=0}^{2m-1} \cos(nx_j) \cos(nx_j) = \sum_{j=0}^{2m-1} \frac{1}{2} (1 + \cos(2nx_j)) = m.$$

The normal equation shall be written as follows:

□

Example 18.3. Let $f(x) = 2x^2 - 9$ for $x \in [-\pi, \pi]$. Find the discrete least squares trigonometric polynomial of degree 2, $S_2(x)$. For $m = 3$.

For $m = 3$, the nodes are

$$x_j = -\pi + \frac{j}{m}\pi \quad y_j = f(x_j) = 2x_j^2 - 9 \quad \text{for } j = 0, 1, 2, 3, 4, 5.$$

The trigonometric polynomial is then

$$S_2(x) = \frac{1}{2}a_0 + a_2 \cos(2x) + (a_1 \cos x + b_1 \sin x),$$

where

$$a_k = \frac{1}{3} \sum_{j=0}^5 y_j \cos(kx_j), \quad k = 0, 1, 2,$$

and

$$b_1 = \frac{1}{3} \sum_{j=0}^5 y_j \sin(kx_j).$$

18.3. Fast Fourier Transform (FFT) by Cooley and Tukey. The interpolation of large amounts of equally spaced data by trigonometric polynomials can produce very accurate results. Note however that the direct calculation technique requires approximately $(2m)^2$ multiplications and $(2m)^2$ additions. More calculations produce more round-off errors. In 1965, Cooley and Tukey wrote a paper on new method that requires only $O(m \log_2 m)$ multiplications and additions. This method by Cooley and Tukey is known as either as the Cooley-Tukey algorithm and the fast Fourier transform (FFT) algorithm.

In this section, we assume that $n = m$. The following so-called the Euler's Identity shall be crucially used.

$$e^{iz} = \cos z + i \sin z.$$

The main idea of FFT is given as follows:

Step 1. Compute the complex coefficients c_k in

$$(18.3) \quad \frac{1}{m} \sum_{k=0}^{2m-1} c_k e^{ikx},$$

where

$$(18.4) \quad c_k = \sum_{j=0}^{2m-1} y_j e^{ik\pi \frac{j}{m}}, \quad \text{for each } k = 0, 1, \dots, 2m-1.$$

Step 2. For each $k = 0, 1, \dots, m$, we have

$$\frac{1}{m} c_k (-1)^k = \frac{1}{m} c_k e^{-i\pi k} = a_k + ib_k.$$

Example 18.4. Consider the FFT technique applied to $4 = 2^2$ data points $\{(x_i, y_i)\}_{i=0}^3$ where

$$x_j = -\pi + \pi \frac{j}{2}, \quad j = 0, 1, 2, 3.$$

Since we assumed that $n = m = 2$, we have

$$S_2(x) = \frac{1}{2} a_0 + a_2 \cos(2x) + a_1 \cos x + b_1 \sin x.$$

To apply the FFT, we first compute

$$c_k = \sum_{j=0}^3 y_j e^{ik\pi\frac{j}{2}}, \quad k = 0, 1, 2, 3.$$

To recover a_0, a_1, a_2, b_1 , we use the relation that

$$a_k + ib_k = c_k \frac{(-1)^k}{m} \quad k = 0, 1, 2.$$

REFERENCES

- [1] D.N. Arnold *A Concise Introduction to Numerical Analysis* Unpublished class note.
- [2] J. Xu *Multilevel Finite Element Theory* Unpublished class note.