Introductory Numerical Analysis
Lecture Notes

Sudipta Mallik
September 9, 2019
## Contents

1. **Introduction to**  
   1.1 Floating Point Numbers ........................................... 1  
   1.2 Computational Errors ............................................. 2  
   1.3 Algorithm ....................................................... 3  
   1.4 Calculus Review .................................................. 3

2. **Root Finding**  
   2.1 Bisection Method ................................................ 5  
   2.2 Fixed Point Iteration ............................................ 8  
   2.3 Newton-Raphson Method ......................................... 13  
   2.4 Order of Convergence ............................................ 16

3. **Interpolation**  
   3.1 Lagrange Polynomials ........................................... 19  
   3.2 Cubic Spline Interpolation ..................................... 23

4. **Numerical Differentiation and Integration**  
   4.1 Numerical Differentiation ....................................... 26  
   4.2 Elements of Numerical Integration ............................... 30  
   4.3 Composite Numerical Integration ................................. 33

5. **Differential Equations**  
   5.1 Euler’s Method ................................................... 36  
   5.2 Higher-order Taylor’s Method ................................... 39  
   5.3 Runge-Kutta Method ............................................... 41

6. **Linear Algebra**  
   6.1 Introduction to Linear Algebra ................................. 45  
   6.2 Systems of Linear Equations: Gaussian Elimination ........... 48  
   6.3 Jacobi and Gauss-Seidel Methods ............................... 52  
   6.4 Eigenvalues: Power Method ..................................... 55

7. **Additional Topics**  
   7.1 Parabolic PDE: Heat Equation ................................... 58  
   7.2 Elliptic PDE: Laplace Equation .................................. 60  
   7.3 Least Squares Approximation ................................... 63
1 Introduction to $\approx$

Sometimes we cannot solve a problem analytically. For example,

Find the root $x^*$ of $f(x) = e^x - x - 2$ on the interval $[0, 2]$.

Also we do not have a general analytic formula or technique to find roots of a polynomial of degree 5 or more (See Galois Theory). We solve these kinds of problem numerically:

- Construct a sequence $\{x_n\}$ that converges to $x^*$, i.e., $\lim_{n \to \infty} x_n = x^*$.
- Approximate $x^*$ by finding $x_k$ for some $k$ for which $f(x_k) = e^{x_k} - x_k - 2 \approx 0$.

Numerical Analysis includes study of the following:

- Construct a sequence $\{x_n\}$ that converges to the solution (Iteration formula)
- Determine how fast $\{x_n\}$ converges to the solution (Rate of convergence)
- Find bounds of error committed at a certain iteration $x_n$ (Error analysis)

We will cover numerical methods for the following topics:

- Root finding
- Interpolation
- Differentiation and integration
- Differential equations
- Linear algebra

1.1 Floating Point Numbers

We know $\pi = 3.14159265358979 \cdots$, where the decimal digits never terminate. For numerical calculations, we consider only a finite number of digits of a number. A $t$-digit floating point number of base 10 is of the form

$$ \pm 0.a_1a_2 \ldots a_t \cdot 10^e, $$

where $0.a_1a_2 \ldots a_t$ is called the mantissa and $e$ is called the exponent. Usually the mantissa $0.a_1a_2 \ldots a_t$ is normalized, i.e., $a_1 \neq 0$. For example, the normalized 15-digit floating point number of $\pi$ is

$$ fl(\pi) = 0.314159265358979 \cdot 10^1. $$
Note that floating point numbers are approximation of the exact numbers obtained by either **chopping** or **rounding up** the digits. The error in calculations caused by the use of floating point numbers is called **roundoff error**. For example, a computer may calculate the following

\[ 2 - (\sqrt{2})^2 = -0.444089209850063 \cdot 10^{-15}, \]

which is just a roundoff error. Note that since floating point numbers are rational numbers, a computer cannot express any irrational number without errors. Also note that almost all computers use binary floating point numbers \(^1\).

### 1.2 Computational Errors

When we approximate a number \( x^* \) by a number \( x \), there are a few ways to measure errors:

**Absolute Error**: \(|x^* - x|\)

**Relative Error**: \(\frac{|x^* - x|}{|x|}, x \neq 0\)

For example, if we approximate \( x^* = 1.24 \) by \( x = 1.25 \), then the absolute error is \(|x^* - x| = |1.24 - 1.25| = 0.01\) and the relative error is \(\frac{|x^* - x|}{|x|} = \frac{|1.24 - 1.25|}{|1.25|} = 0.008\).

The relative error gives us information about the number of decimal digits of \( x^* \) and \( x \) match. We approximate \( x^* \) by \( x \) to \( n \) **significant digits** if \( n \) is the largest nonnegative integer for which

\[ \frac{|x^* - x|}{|x|} < 5 \cdot 10^{-n}. \]

Since \( \frac{|x^* - x|}{|x|} = 0.008 < 5 \cdot 10^{-2} \) and \( \frac{|x^* - x|}{|x|} = 0.008 \not< 5 \cdot 10^{-3} \), we have the largest nonnegative integer \( n = 2 \). Thus \( x^* = 1.24 \) and \( x = 1.25 \) agree to 2 significant digits.

---

\(^1\)A 64-bit computer uses the IEEE 754-2008 standard which defines the following format for 64-bit binary floating point numbers (Double-precision floating point format):

\[ s \quad x \quad f \]

where

\[ (1 \text{- bit sign}) \quad (11 \text{- bit exponent}) \quad (52 \text{- bit fraction}) \]

converts to the following decimal number

\[ (-1)^s \cdot 2^{x_{10} - 1023} \left( 1 + (f_1 \cdot 2^{-1} + f_2 \cdot 2^{-2} + \cdots + f_{52} \cdot 2^{-52}) \right), \]

where \( x_{10} \) is the decimal number of \( x \). For example, \( 1 \ 1000000000 \ 1100000 \cdots 0 \) converts to

\[ (-1)^1 \cdot 2^{1026 - 1023} \left( 1 + (1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + \cdots + 0 \cdot 2^{-52}) \right) = -2^3 \left( 1 + \frac{1}{2} + \frac{1}{4} \right) = -14. \]

In this format the magnitude of the largest and smallest decimal numbers is \( 1.797693134862231 \cdot 10^{308} \). If a number has magnitude bigger than that, a 64-bit computer stops working and it is called an **overflow**. Note that the single-precision floating point format uses 32 bits including 23 bits fraction.
1.3 Algorithm

An algorithm is a set of ordered logical operations that applies to a problem defined by a set of data (input data) to produce a solution (output data). An algorithm is usually written in an informal way (pseudocode) before writing it with syntax of a computer language such as C, Java, Python etc. The following is an example of a pseudocode to find \( n! \):

Algorithm you-name-it
Input: nonnegative integer \( n \)
Output: \( n! \)
\[
\text{fact} = 1 \\
\text{for } i = 2 \text{ to } n \\
\quad \text{fact} = \text{fact} \times i \\
\text{end for} \\
\text{return } \text{fact}
\]

Stopping criteria: Sometimes we need a stopping criteria to terminate an algorithm. For example, when an algorithm approximates a solution \( x^* \) by constructing a sequence \( \{x_n\} \), the algorithm needs to stop after finding \( x_k \) for some \( k \). There is no universal stopping criteria as it depends on the problem, acceptable error (i.e., error < tolerance, say \( 10^{-4} \)), the maximum number of iterations etc.

1.4 Calculus Review

You should revise the limit definitions of the derivative and the Riemann integral of a function from a standard text book. The following are some theorems which will be used later.

Theorem. Let \( f \) be a differentiable function on \([a,b]\).

- \( f \) is increasing on \([a,b]\) if and only if \( f'(x) > 0 \) for all \( x \in [a,b] \).
- If \( f \) has a local maximum or minimum value at \( c \), then \( f'(c) = 0 \) (\( c \) is a critical number).
- If \( f'(c) = 0 \) and \( f''(c) < 0 \), then \( f(c) \) is a local maximum value.
- If \( f'(c) = 0 \) and \( f''(c) > 0 \), then \( f(c) \) is a local minimum value.

Theorem. Let \( f \) be a continuous function on \([a,b]\). If \( f(c) \) is the absolute maximum or minimum value \( f \) on \([a,b]\), then either \( f'(c) \) does not exist or \( f'(c) = 0 \) or \( c = a \) or \( c = b \).
Intermediate Value Theorem: Let $f$ be a function such that

(i) $f$ is continuous on $[a, b]$, and

(ii) $N$ is a number between $f(a)$ and $f(b)$.

Then there is at least one number $c$ in $(a, b)$ such that $f(c) = N$.

In the particular case when $f(a)f(b) < 0$, i.e., $f(a)$ and $f(b)$ are of opposite signs, there is at least one root $c$ of $f$ in $(a, b)$.

Mean Value Theorem: Let $f$ be a function such that

(i) $f$ is continuous on $[a, b]$, and

(ii) $f$ is differentiable on $(a, b)$.

Then there is a number $c$ in $(a, b)$ such that

$$\frac{f(b) - f(a)}{b - a} = f'(c).$$

Taylor’s Theorem: Let $n \geq 1$ be an integer and $f$ be a function such that

(i) $f^{(n)}$ is continuous on $[a, b]$, and

(ii) $f^{(n)}$ is differentiable on $(a, b)$.

Let $c$ be a number in $(a, b)$. Then for all $x$ in $[a, b]$, we have

$$f(x) = f(c) + \frac{f'(c)}{1!}(x - c) + \frac{f''(c)}{2!}(x - c)^2 + \cdots + \frac{f^{(n)}(c)}{n!}(x - c)^n + \frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - c)^{n+1},$$

for some number $\xi$ (depends on $x$) between $c$ and $x$.

Sometimes we simply write $f(x) = T_n(x) + R_n(x)$, where $T_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x - c)^k$ is the Taylor polynomial of degree $n$ and $R_n(x) = \frac{f^{(n+1)}(\xi)}{(n + 1)!}(x - c)^{n+1}$ is the remainder term.
2 Root Finding

In this chapter we will find roots of a given function \( f \), i.e., \( x^* \) for which \( f(x^*) = 0 \).

2.1 Bisection Method

Suppose \( f \) is a continuous function on \([a, b]\) and \( f(a) \) and \( f(b) \) have the opposite signs. Then by the IVT (Intermediate Value Theorem), there is at least one root \( x^* \) of \( f \) in \((a, b)\). For simplicity, let’s assume the root \( x^* \) is unique. Set \( a_1 = a \) and \( b_1 = b \). Let \( x_1 = \frac{a_1 + b_1}{2} \) which breaks \([a_1, b_1]\) into two subintervals \([a_1, x_1]\) and \([x_1, b_1]\). Then there are three possibilities:

1. \( x_1 = x^* \): \( f(x_1) = 0 \) and we are done.
2. \( x^* \in (a_1, x_1) \): \( f(x_1) \) and \( f(a_1) \) have the opposite signs and set \( a_2 = a_1 \), \( b_2 = x_1 \).
3. \( x^* \in (x_1, b_1) \): \( f(x_1) \) and \( f(b_1) \) have the opposite signs and set \( a_2 = x_1 \), \( b_2 = b_1 \).

Set \( x_2 = \frac{a_2 + b_2}{2} \). We can continue this process of bisecting an interval \([a_n, b_n]\) containing \( x^* \) and getting an approximation \( x_n = \frac{a_n + b_n}{2} \) of \( x^* \). We will show that \( \{x_n\} \) converges to \( x^* \).

Example. Do five iterations by the Bisection Method to approximate the root \( x^* \) of \( f(x) = e^x - x - 2 \) on the interval \([0, 2]\).

Solution.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( a_n )</th>
<th>( f(a_n) )</th>
<th>( b_n )</th>
<th>( f(b_n) )</th>
<th>( x_n = \frac{a_n + b_n}{2} )</th>
<th>( f(x_n) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>-</td>
<td>2</td>
<td>+</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>+</td>
<td>1.5</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
<td>1.5</td>
<td>+</td>
<td>1.25</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-</td>
<td>1.25</td>
<td>+</td>
<td>1.125</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>1.125</td>
<td>-</td>
<td>1.25</td>
<td>+</td>
<td>1.1875</td>
<td>+</td>
</tr>
</tbody>
</table>
Since $|x_5 - x_4| = |1.1875 - 1.125| = 0.0625 < 0.1 = 10^{-1}$, we can (roughly) say that the root is $x_5 = 1.1875$ correct to one decimal place. But why roughly?

If $x_n$ is correct to $t$ decimal places, then $|x^* - x_n| < 10^{-t}$. But the converse is not true.

For example, $x^* = 1.146193$ is correct to 6 decimal places (believe me!). So $x_{11} = 1.145507$ is only correct to 2 decimal places but $|x^* - x_{11}| < 10^{-3}$. Similarly if $x^* = 1.000$ is approximated by $x = 0.999$, then $|x^* - x| = 0.001 < 10^{-2}$. But $x = 0.999$ is not correct to any decimal places of $x^* = 1.000$. Also note that we computed $|x_5 - x_4|$, not $|x^* - x_5|$ (which is not known for $x^*$).

It would be useful to know the number of iteration that guarantees to achieve a certain accuracy of the root, say within $10^{-3}$. That is to find $n$ for which $|x^* - x_n| < 10^{-3}$, i.e., $x^* - 10^{-3} < x_n < x^* + 10^{-3}$. So we have to find an upper bound of the absolute error for the $n$th iteration $x_n$.

**The Maximum Error:** Let $\varepsilon_n$ be the absolute error for the $n$th iteration $x_n$. Then

$$\varepsilon_n = |x^* - x_n| \leq \frac{b - a}{2^n}.$$ 

**Proof.** Since $x^* \in [a_n, b_n]$ and $b_n - a_n = (b_{n-1} - a_{n-1})/2$ for all $n$,

$$\varepsilon_n = |x^* - x_n| \leq \frac{b_n - a_n}{2} = \frac{b_{n-1} - a_{n-1}}{2^2} = \cdots = \frac{b_1 - a_1}{2^n} = \frac{b - a}{2^n}.$$ 

**Example.** Find the number of iteration by the Bisection Method that guarantees to approximate the root $x^*$ of $f(x) = e^x - x - 2$ on $[0, 2]$ with accuracy within $10^{-3}$.

**Solution.**

$$\varepsilon_n = |x^* - x_n| \leq \frac{2 - 0}{2^n} = \frac{2}{2^n} < 10^{-3}$$

$$\Rightarrow 2 \cdot 10^3 < 2^n$$

$$\Rightarrow \ln(2 \cdot 10^3) < \ln 2^n$$

$$\Rightarrow \ln 2 + 3 \ln(10) < n \ln 2$$

$$\Rightarrow \frac{\ln 2 + 3 \ln(10)}{\ln 2} < n$$

$$\Rightarrow 10.97 \approx \frac{\ln 2 + 3 \ln(10)}{\ln 2} < n$$

Thus 11th iteration guarantees to achieve accuracy of the root within $10^{-3}$.

Note that the root is $x^* = 1.146193$ correct to 6 decimal places. So $x_{10} = 1.146484$ and $x_{11} = 1.145507$ both have accuracy within $10^{-3}$ (check: $|x^* - x_{10}| < 10^{-3}$, $|x^* - x_{11}| < 10^{-3}$). Thus accuracy within $10^{-3}$ is reached even before 11th iteration.

It is interesting to note that $x_{10} = 1.146484$ is correct to 3 decimal places whereas $x_{11} = 1.145507$ is only correct to 2 decimal places.
Convergence: The sequence \( \{x_n\} \) constructed by the bisection method converges to the solution \( x^* \) because

\[
\lim_{n \to \infty} |x^* - x_n| \leq \lim_{n \to \infty} \frac{b - a}{2^n} = 0 \implies \lim_{n \to \infty} |x^* - x_n| = 0.
\]

But it converges really slowly in comparison to other methods (See Section 2.4).

Algorithm Bisection-Method
Input: \( f(x) = e^x - x - 2 \), interval \([0, 2]\), tolerance \(10^{-3}\), maximum number of iterations 50
Output: an approximate root of \( f \) on \([0, 2]\) within \(10^{-3}\) or a message of failure

set \( a = 0 \) and \( b = 2 \);
\( x_{old} = a \);
for \( i = 1 \) to 50
\( x = (a + b)/2 \);
if \( |x - x_{old}| < 10^{-3} \) % checking required accuracy
    FoundSolution= true; % done
    break; % leave for environment
end if
if \( f(a)f(x) < 0 \)
    a=a and b=x
else
    a=x and b=b
end if
\( x_{old} = x \) % update \( x_{old} \) for the next iteration
end for
if FoundSolution
    return \( x \)
else
    print ‘the required accuracy is not reached in 50 iterations’
end if
2.2 Fixed Point Iteration

A number \( p \) is a fixed point of a function \( g \) if \( g(p) = p \). For example, if \( g(x) = x^2 - 2 \), then solving \( g(x) = x^2 - 2 = x \) we get \( x = -1, 2 \). We can easily check \( g(-1) = -1 \) and \( g(2) = 2 \). Thus \(-1\) and \( 2 \) are fixed points of \( g \). Note that fixed points of \( g \) are the \( x \)-value of the points of intersection of the curve \( y = g(x) \) and the line \( y = x \).

The following shows the equivalence of root finding and finding fixed points.

**Observation.** \( p \) is a fixed point of a function \( g \) if and only if \( p \) is a root of \( f(x) = x - g(x) \).

If \( p \) is a fixed point of a function \( g \), then \( g(p) = p \) and consequently \( f(p) = p - g(p) = 0 \). The converse follows similarly. Note that there are multiple choices for \( f \) such as \( f(x) = e^x(x - g(x)) \), \( f(x) = -1 + e^x - g(x) \) etc.

The following theorem gives us sufficient conditions for existence and uniqueness of a fixed point:

**Theorem 2.1** (Fixed Point Theorem). Let \( g \) be a function on \([a, b] \).

1. *(Existence)* If \( g \) is continuous and \( a \leq g(x) \leq b \) for all \( x \in [a, b] \), then \( g \) has a fixed point in \([a, b] \).

2. *(Uniqueness)* Moreover, if \( |g'(x)| < 1 \) for all \( x \in (a, b) \), then \( g \) has a unique fixed point in \([a, b] \).

**Proof.** Suppose \( g \) is continuous and \( a \leq g(x) \leq b \) for all \( x \in [a, b] \). If \( g(a) = a \) or \( g(b) = b \), then \( a \) or \( b \) is a fixed point of \( g \). Otherwise \( g(a) > a \) and \( g(b) < b \) because \( a \leq g(x) \leq b \) for all \( x \in [a, b] \).

Define a new function \( h \) by \( h(x) = x - g(x) \). Since \( g \) is continuous, \( h \) is also continuous. Also note that \( h(a) = a - g(a) < 0 \) and \( h(b) = b - g(b) > 0 \). By the IVT, \( h(c) = 0 \) for some \( c \in (a, b) \). Now \( h(c) = c - g(c) = 0 \implies g(c) = c \), i.e., \( c \) is a fixed point of \( g \).
Suppose \(|g'(x)| < 1\) for all \(x \in (a,b)\). To show uniqueness of \(c\), suppose \(d\) is another fixed point of \(g\) in \([a,b]\). WLOG let \(d > c\). Applying the MVT on \(g\) on \([c,d]\), we find \(t \in (c,d)\) such that
\[
g(d) - g(c) = g'(t)(d - c).
\]
Since \(|g'(x)| < 1\) for all \(x \in (a,b)\),
\[
g(d) - g(c) = g'(t)(d - c) < (d - c).
\]
Since \(g(c) = c\) and \(g(d) = d\), we have \(g(d) - g(c) = d - c\) which contradicts that \(g(d) - g(c) < (d - c)\).

Suppose \(a \leq g(x) \leq b\) for all \(x \in [a,b]\) and \(|g'(x)| \leq k < 1\) for all \(x \in (a,b)\). For any initial approximation \(x_0\) in \([a,b]\), the fixed point iteration constructs a sequence \(\{x_n\}\), where
\[
x_{n+1} = g(x_n), \ n = 0, 1, 2, \ldots
\]
to approximate the unique fixed point \(x^*\) of \(g\) in \([a,b]\). We will show \(\{x_n\}\) converges to \(x^*\).

Example. This problem approximates the root \(x^*\) of \(f(x) = e^x - x - 2\) on the interval \([0, 2]\).

(a) Find a function \(g\) that has a unique fixed point which is the root \(x^*\) of \(f(x) = e^x - x - 2\) on the interval \([0, 2]\).

(b) Do six iterations by the fixed point iteration to approximate the root \(x^*\) of \(f(x) = e^x - x - 2\) on the interval \([0, 2]\) using \(x_0 = 1\).

Solution. (a) We need to find a function \(g\) such that

(i) \(0 \leq g(x) \leq 2\) for all \(x \in [0, 2]\), and

(ii) \(|g'(x)| < 1\) for all \(x \in (0, 2)\).
Since \( f(x) = e^x - x - 2 = 0 \implies x = e^x - 2 \). But \( g(x) = e^x - 2 \) does not satisfy (i) as \( g(0) = -2 < 0 \) and also (ii) as \( g'(2) = e^2 > 1 \). Note that
\[
f(x) = e^x - x - 2 = 0 \implies e^x = x + 2 \implies x = \ln(x + 2).
\]

Take \( g(x) = \ln(x + 2) \). Then \( g \) is an increasing function and \( g(0) = \ln 2 > 0 \) and \( g(2) = \ln 4 < 2 \). Thus \( 0 \leq g(x) \leq 2 \) for all \( x \in [0, 2] \). Also \( |g'(x)| = \frac{1}{x+2} \leq \frac{1}{2} < 1 \) for all \( x \in (0, 2) \). Thus \( g \) has a unique fixed point in \([0, 2]\) which is the root \( x^* \) of \( f(x) = e^x - x - 2 \) on the interval \([0, 2]\).

(b) Use \( g(x) = \ln(x + 2) \) and \( x_0 = 1 \).

<table>
<thead>
<tr>
<th>( n )</th>
<th>( x_{n-1} )</th>
<th>( x_n = g(x_{n-1}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1.0986</td>
</tr>
<tr>
<td>2</td>
<td>1.0986</td>
<td>1.1309</td>
</tr>
<tr>
<td>3</td>
<td>1.1309</td>
<td>1.1413</td>
</tr>
<tr>
<td>4</td>
<td>1.1413</td>
<td>1.1446</td>
</tr>
<tr>
<td>5</td>
<td>1.1446</td>
<td>1.1456</td>
</tr>
<tr>
<td>6</td>
<td>1.1457</td>
<td>1.1460</td>
</tr>
</tbody>
</table>

Since \( |x_6 - x_5| = |1.1460 - 1.1456| = 0.0004 < 0.001 = 10^{-3} \), we can say that the root is \( x_6 = 1.1460 \) roughly correct to three decimal places (which is true indeed as \( x^* = 1.146193 \)).

**The Maximum Error:** Let \( \varepsilon_n \) be the absolute error for the \( n \)th iteration \( x_n \). Then
\[
\varepsilon_n = |x^* - x_n| \leq k^n \max\{x_0 - a, b - x_0\} \quad \text{and} \quad \varepsilon_n = |x^* - x_n| \leq \frac{k^n}{1 - k}|x_1 - x_0|.
\]

**Proof.** Applying the MVT on \( g \) on \([x^*, x_{n-1}]\), we find \( \xi_{n-1} \in (x^*, x_{n-1}) \) such that
\[
g(x^*) - g(x_{n-1}) = g'(\xi_{n-1})(x^* - x_{n-1}).
\]
Then
\[
|x^* - x_n| = |g(x^*) - g(x_{n-1})| = |g'(\xi_{n-1})||x^* - x_{n-1}| \leq k|x^* - x_{n-1}|.
\]
Continuing this process, we get
\[
|x^* - x_n| \leq k|x^* - x_{n-1}| \leq k^2|x^* - x_{n-2}| \leq \cdots \leq k^n|x^* - x_0|.
\]
Since \( x^*, x_0 \in (a, b) \), we have \( |x^* - x_0| \leq \max\{x_0 - a, b - x_0\} \). Thus \( \varepsilon_n = |x^* - x_n| \leq k^n \max\{x_0 - a, b - x_0\} \).

Note that similarly applying the MVT on \( g \) on \([x_n, x_{n+1}]\), we can show that
\[
|x_{n+1} - x_n| \leq k^n|x_1 - x_0|.
\]
For the other bound, let \( m > n \geq 0 \). Then
\[
|x_m - x_n| = |x_m - x_{m-1} + x_{m-1} - x_{m-2} + \cdots + x_{n+1} - x_n|
\leq |x_m - x_{m-1}| + |x_{m-1} - x_{m-2}| + \cdots + |x_{n+1} - x_n|
\leq k^{m-1}|x_1 - x_0| + k^{m-2}|x_1 - x_0| + \cdots + k^n|x_1 - x_0|
\leq k^n|x_1 - x_0|(1 + k + \cdots + k^{m-n-1})
\leq k^n|x_1 - x_0| \sum_{i=0}^{m-n-1} k^i
\]
Thus
\[
\lim_{m \to \infty} |x_m - x_n| \leq \lim_{m \to \infty} k^n|x_1 - x_0| \sum_{i=0}^{m-n-1} k^i = k^n|x_1 - x_0| \sum_{i=0}^{\infty} k^i = \frac{k^n}{1-k}|x_1 - x_0| \quad \text{(as } 0 < k < 1) \quad \square
\]

**Convergence:** The sequence \( \{x_n\} \) constructed by the fixed point iteration converges to the unique fixed point \( x^* \) irrespective of choice of the initial approximation \( x_0 \) because
\[
\lim_{n \to \infty} |x^* - x_n| \leq \lim_{n \to \infty} k^n \max\{x_0 - a, b - x_0\} = 0 \quad \text{(as } 0 < k < 1) \quad \implies \lim_{n \to \infty} |x^* - x_n| = 0.
\]
It converges really fast when \( k \) is close to 0.

**Example.** Find the number of iteration by the fixed point iteration that guarantees to approximate the root \( x^* \) of \( f(x) = e^x - x - 2 \) on \([0, 2]\) using \( x_0 = 1 \) with accuracy within \( 10^{-3} \).

**Solution.** Consider \( g(x) = \ln(x + 2) \) on \([0, 2]\) where \( |g'(x)| \leq \frac{1}{2} = k < 1 \) for all \( x \in (0, 2) \).
\
\[
\varepsilon_n = |x^* - x_n| \leq k^n \max\{x_0 - a, b - x_0\} = \left(\frac{1}{2}\right)^n \cdot 1 < 10^{-3}
\]
\[
\implies 10^3 < 2^n \quad \implies \ln 10^3 < \ln 2^n \quad \implies 3 \ln 10 < n \ln 2 \quad \implies \frac{3 \ln 10}{\ln 2} \approx 9.97 < n
\]
Thus 10th iteration guarantees to achieve accuracy of the root within \( 10^{-3} \). But note that \( |x^* - x_5| = |1.1461 - 1.1457| = 0.0004 < 10^{-3} \). So Thus accuracy within \( 10^{-3} \) is reached at 5th iteration (way before 10th iteration). Also note that the other bound of \( \varepsilon_n = |x^* - x_n| \leq \frac{k^n}{1-k}|x_1 - x_0| \) gives 10.94 < \( n \) which does not improve our answer.
Algorithm Fixed-point-Iteration
Input: $g(x) = \ln(x + 2)$, interval $[0, 2]$, an initial approximation $x_0$, tolerance $10^{-3}$, the maximum number of iterations 50
Output: an approximate fixed point of $g$ on $[0, 2]$ within $10^{-3}$ or a message of failure

set $x = x_0$ and $xold = x_0$;
for $i = 1$ to $50$
    $x = g(x);$  
    if $|x - xold| < 10^{-3}$  
        FoundSolution= true;  
        break;  
    end if
    $xold = x;$  
end for
if FoundSolution
    return $x$
else
    print ‘the required accuracy is not reached in 50 iterations’
end if
2.3 Newton-Raphson Method

Suppose \( f \) is a function with a unique root \( x^* \) in \([a, b]\). Assume \( f'' \) is continuous in \([a, b]\).

To find \( x^* \), let \( x_0 \) be a “good” initial approximation (i.e., \( |x^* - x_0|^2 \approx 0 \)) where \( f'(x_0) \neq 0 \). Using Taylor’s Theorem on \( f \) about \( x_0 \), we get

\[
f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(\xi)}{2!}(x - x_0)^2,
\]

for some number \( \xi \) (depends on \( x \)) between \( x_0 \) and \( x \). Plugging \( x = x^* \), we get

\[
0 = f(x^*) = f(x_0) + \frac{f'(x_0)}{1!}(x^* - x_0) + \frac{f''(\xi)}{2!}(x^* - x_0)^2
\]

\[
\implies f(x_0) + \frac{f'(x_0)}{1!}(x^* - x_0) = -\frac{f''(\xi)}{2!}(x^* - x_0)^2 \approx 0 \quad \text{(since } |x^* - x_0|^2 \approx 0)\]

Now solving for \( x^* \), we get

\[
x^* \approx x_0 - \frac{f(x_0)}{f'(x_0)}.
\]

So \( x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \) is an approximation to \( x^* \). Using \( x_1 \), similarly we get \( x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \).

Continuing the process, we get a sequence \( \{x_n\} \) to approximate \( x^* \) where

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots
\]

**Observation.** \( x_{n+1} \) is the \( x \)-intercept of the tangent line to \( y = f(x) \) at \((x_n, f(x_n))\).

The equation of the tangent line to \( y = f(x) \) at \((x_n, f(x_n))\) is

\[
y = f(x_n) + f'(x_n)(x - x_n).
\]
For the $x$-intercept, $y = 0$. So

$$0 = f(x_n) + f'(x_n)(x - x_n) \implies x = x_n - \frac{f(x_n)}{f'(x_n)},$$

Thus $x_{n+1}$ is the $x$-intercept of the tangent line to $y = f(x)$ at $(x_n, f(x_n))$.

**Remark.** We will show later that $\{x_n\}$ converges to $x^*$ for a good choice of an initial approximation $x_0$. If $x_0$ is far from $x^*$, then $\{x_n\}$ may not converge to $x^*$. For example, $f(x) = x^2 - 5$ has a unique root $x^* = \sqrt{5}$ in $[-2, 3]$. But the Newton-Raphson Method constructs a sequence $\{x_n\}$ using $x_0 = -2$ that converges to $-\sqrt{5} \neq x^*$.

**Example.** Do four iterations by the Newton-Raphson Method to approximate the root $x^*$ of $f(x) = e^x - x - 2$ on the interval $[0, 2]$ using $x_0 = 1$.

**Solution.** $f(x) = e^x - x - 2 \implies f'(x) = e^x - 1$ and $f'(1) = e - 1 \neq 0$. Thus

$$x_{n+1} = x_n - \frac{e^{x_n} - x_n - 2}{e^{x_n} - 1}, \quad n = 0, 1, 2, \ldots$$

<table>
<thead>
<tr>
<th>$n$</th>
<th>$x_n$</th>
<th>$f(x_n)$</th>
<th>$f'(x_n)$</th>
<th>$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>-0.2817</td>
<td>1.7182</td>
<td>1.1639</td>
</tr>
<tr>
<td>1</td>
<td>1.1639</td>
<td>0.0384</td>
<td>2.2023</td>
<td>1.1464</td>
</tr>
<tr>
<td>2</td>
<td>1.1464</td>
<td>0.0004</td>
<td>2.1468</td>
<td>1.1462</td>
</tr>
<tr>
<td>3</td>
<td>1.1462</td>
<td>0.0000</td>
<td>2.1462</td>
<td>1.1462</td>
</tr>
</tbody>
</table>

Since $|x_4 - x_3| < 10^{-4}$, we can say that the root is $x_4 = 1.1462$ **roughly** correct to four decimal places (almost true as $x^* = 1.146193$). Actually $x_4$ is correct to 12 decimal places! Note that this sequence converges to the root faster than that of other methods (why?).

**Convergence:** Suppose $f$ is a function with a simple root $x^*$ in $[a, b]$, i.e., $f'(x^*) \neq 0$. Assume $f''$ is continuous in $[a, b]$. Then there is a $\delta > 0$ such that the sequence $\{x_n\}$ constructed by the Newton-Raphson Method converges to $x^*$ for any choice of the initial approximation $x_0 \in (x^* - \delta, x^* + \delta)$.

**Proof.** Consider the following function $g$ on $[a, b]$:

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

First note that $x^*$ is a fixed point of $g$ in $[a, b]$. Since $f'(x^*) \neq 0$, by the continuity of $f'$ in $[a, b]$, there is a $\delta_1 > 0$ such that $f'(x) \neq 0$ for all $x \in [x^* - \delta_1, x^* + \delta_1]$. So $g$ is continuous on $[x^* - \delta_1, x^* + \delta_1]$. 


The convergence follows by that of the Fixed Point Iteration of \(g\) on \([x^*-\delta, x^*+\delta]\) if we can show that there is a positive \(\delta < \delta_1\) such that (i) \(x^*-\delta \leq g(x) \leq x^*+\delta\) for all \(x \in [x^*-\delta, x^*+\delta]\) and (ii) \(|g'(x)| < 1\) for all \(x \in (x^*-\delta, x^*+\delta)\). To prove (ii), note that

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

\[
\implies g'(x) = \frac{f(x)f''(x)}{(f'(x))^2} = 0 \quad (\text{since } f(x^*) = 0, \ f'(x^*) \neq 0)
\]

Since \(g'(x^*) = 0\), by the continuity of \(g'\) in \([x^*-\delta, x^*+\delta]\), there is a positive \(\delta < \delta_1\) such that \(|g'(x)| < 1\) for all \(x \in (x^*-\delta, x^*+\delta)\). So we have (ii). To show (i), take \(x \in [x^*-\delta, x^*+\delta]\). By the MVT on \(g\), we have \(g(x) - g(x^*) = \frac{g'(\xi)(x-x^*)}{2}\) for some \(\xi\) between \(x\) and \(x^*\). Thus

\[
|g(x) - x^*| = |g(x) - g(x^*)| = |\frac{g'(\xi)}{2}| |x-x^*| < |x-x^*|
\]

Since \(x \in [x^*-\delta, x^*+\delta]\), i.e., \(|x-x^*| \leq \delta\), we have \(|g(x) - x^*| < |x-x^*| \leq \delta\), i.e.,

\[x^*-\delta \leq g(x) \leq x^*+\delta\]

Note that if the root \(x^*\) of \(f\) is not simple, i.e., \(f'(x^*) = 0\), then this proof does not work but still \(\{x_n\}\) may converge to \(x^*\). For multiple root \(x^*\) of \(f\) we use a modified Newton-Raphson iteration formula.

**The Secant Method:** In the iteration formula by the Newton-Raphson Method

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, 2, \ldots,
\]

we need to calculate the derivative \(f'(x_n)\) which may be difficult sometimes. To avoid such calculations, the Secant Method modifies the above formula by replacing \(f'(x_n)\) with its approximation. Note that

\[
f'(x_n) = \lim_{x \to x_n} \frac{f(x) - f(x_n)}{x - x_n}.
\]

If \(x_{n-1}\) is close to \(x_n\), then

\[
f'(x_n) \approx \frac{f(x_{n-1}) - f(x_n)}{x_{n-1} - x_n} = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.
\]

Thus

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

Thus the iteration formula by the Secant Method is

\[
x_{n+1} = x_n - \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}, \quad n = 1, 2, 3, \ldots
\]

Note that geometrically \(x_{n+1}\) is the \(x\)-intercept of the secant line joining \((x_n, f(x_n))\) and \((x_{n-1}, f(x_{n-1}))\). Also note that the convergence of the sequence \(\{x_n\}\) by the Secant Method is slower than that of the Newton-Raphson Method.
2.4 Order of Convergence

In the preceding three sections we learned techniques to construct a sequence \( \{x_n\} \) that converges to the root \( x^* \) of a function. But the speed of their convergence are different. In this section we will compare them by their order of convergence.

**Definition.** The convergence of a sequence \( \{x_n\} \) to \( x^* \) is of order \( p \) if

\[
\lim_{n \to \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^p} = \lim_{n \to \infty} \frac{\varepsilon_{n+1}}{\varepsilon_n^p} = c,
\]

for some constant \( c > 0 \).

For higher order convergence (i.e., larger \( p \)), the sequence converges more rapidly. The rate of convergence is called linear, quadratic, and superlinear if \( p = 1 \), \( p = 2 \), and \( 1 < p < 2 \) respectively. We say \( \{x_n\} \) converges linearly, quadratically, or superlinearly to \( x^* \). In quadratic convergence, we have \( \varepsilon_{n+1} \approx c\varepsilon_n^2 \) and then accuracy of \( x_n \) to \( x^* \) gets roughly double at each iteration.

**Example.** The sequence \( \{\frac{1}{2^n}\} \) converges linearly to 0.

\[
\lim_{n \to \infty} \frac{|2^{-(n+1)} - 0|}{|2^{-n} - 0|^p} = \lim_{n \to \infty} 2^{p(n-1)} = \begin{cases} 
0 & \text{if } p < 1 \\
1/2 & \text{if } p = 1 \\
\infty & \text{if } p > 1.
\end{cases}
\]

**Example.** The order of convergence of the Bisection Method is linear.

Recall that

\[
\varepsilon_n = |x^* - x_n| \leq \frac{b - a}{2^n}.
\]

So the rate of convergence of the sequence \( \{x_n\} \) is similar to (at least) that of \( \{\frac{1}{2^n}\} \). We denote this by \( x_n = x^* + O\left(\frac{1}{2^n}\right) \). Since \( \{\frac{1}{2^n}\} \) converges linearly, \( \{x_n\} \) also converges linearly.

**Example.** The order of convergence of the Fixed Point Iteration is linear when \( g'(x^*) \neq 0 \).

Consider the sequence \( \{x_n\} \), where \( x_{n+1} = g(x_n) \), that converges to \( x^* \). We have shown before by applying the MVT on \( g \) on \([x^*, x_n]\), that we find \( \xi_n \in (x^*, x_n) \) such that

\[
x^* - x_{n+1} = g(x^*) - g(x_n) = g' (\xi_n)(x^* - x_n).
\]

Since \( \{x_n\} \) converges to \( x^* \) and \( \xi_n \in (x^*, x_n), \{\xi_n\} \) also converges to \( x^* \). Then

\[
\lim_{n \to \infty} \frac{|x^* - x_{n+1}|}{|x^* - x_n|} = \lim_{n \to \infty} |g'(\xi_n)| = |g'(\lim_{n \to \infty} \xi_n)| = |g'(x^*)| \quad (\text{Assuming continuity of } g').
\]
**Example.** The order of convergence of the Newton-Raphson Method to find a simple root is **quadratic**.

Recall that to find a simple root \( x^* \) of \( f \) (i.e., \( f'(x^*) \neq 0 \)), we used the Fixed Point Iteration on

\[
g(x) = x - \frac{f(x)}{f'(x)}
\]

for any choice of the initial approximation \( x_0 \in (x^* - \delta, x^* + \delta) \) for small \( \delta > 0 \). Also recall that since \( f(x^*) = 0 \) and \( f'(x^*) \neq 0 \), \( g(x^*) = 0 \). By Taylor’s Theorem on \( g \) about \( x^* \), we get

\[
g(x) = g(x^*) + g'(x^*)\frac{(x - x^*)}{1!} + g''(\xi)(x - x^*)^2,
\]

for some \( \xi \) between \( x \) and \( x^* \). For \( x = x_n \), we get \( \xi_n \) between \( x_n \) and \( x^* \) such that

\[
x_{n+1} = x^* + g'(x^*)\frac{(x_n - x^*)}{1!} + g''(\xi_n)(x_n - x^*)^2 \quad \text{and} \quad x_{n+1} - x^* = g''(\xi_n)\frac{(x_n - x^*)^2}{2!} \quad \text{(since } g'(x^*) = 0)\]

\[
\lim_{n \to \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \lim_{n \to \infty} \frac{|g''(\xi_n)|}{2!} = \frac{|g''(\lim_{n \to \infty} \xi_n)|}{2!} \quad \text{(Assuming continuity of } g''\text{)}
\]

Since \( \{x_n\} \) converges to \( x^* \) and \( \xi_n \) lies between \( x_n \) and \( x^* \), \( \{\xi_n\} \) also converges to \( x^* \). Thus

\[
\lim_{n \to \infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \frac{|g''(x^*)|}{2} = \frac{|f''(x^*)|}{2|f'(x^*)|}.
\]

**Remark.**

1. If the root is **not simple**, the Newton-Raphson Method still may converge but the order of convergence is **not quadratic** but linear. For example, to approximate the double root zero of \( f(x) = x^2 \) using \( x_0 = 1 \) by the Newton-Raphson Method, it constructs the sequence \( \{\frac{1}{2^n}\} \) which converges to 0 linearly.

2. A modified Newton-Raphson Method to find a multiple root has **linear convergence**.

3. The order of convergence of the Secant Method is **superlinear** \( (p = (1+\sqrt{5})/2 \approx 1.6) \) which is in between that of the Bisection Method and the Newton-Raphson Method. (Proof skipped)
3 Interpolation

Suppose we have a set of data in which for each \( x_i \) you have \( y_i \):

\[
\begin{array}{cc}
  x_0 & y_0 \\
  x_1 & y_1 \\
  x_2 & y_2 \\
  x_3 & y_3 \\
  \vdots & \vdots \\
  x_n & y_n \\
\end{array}
\]

So there is an unknown function \( f \) for which \( f(x_i) = y_i \), \( i = 0, 1, \ldots, n \). With this data we would like to predict \( f(x^*) \) for a given \( x^* \in [x_0, x_n] \) where \( x^* \neq x_i \), \( i = 0, 1, \ldots, n \). This method of finding an untabulated data from a given table of data is called interpolation.

How do we find or approximate the unknown function \( f \)? One easy answer is to get a piecewise linear function \( f^* \) such that \( f^*(x) = y_i + (x-x_i) \frac{y_i-y_{i-1}}{x_i-x_{i-1}} \) for all \( x \in [x_{i-1}, x_i] \). But this is too naive because it assumes the functional values are changing at a constant rate \( \frac{y_i-y_{i-1}}{x_i-x_{i-1}} \) on the entire interval \( [x_{i-1}, x_i] \).

There are multiple techniques of approximating \( f \). We will mainly focus on approximating \( f \) by a polynomial \( P_n \) of degree \( n \), called the interpolating polynomial and the method is called the polynomial interpolation. The polynomial interpolation is suggested by the following theorem:

**Theorem** (Weierstrass Approximation Theorem). For a given continuous function \( f \) on \( [a, b] \) and a small positive number \( \varepsilon \), there exists a polynomial \( P \) such that

\[
|f(x) - P(x)| < \varepsilon,
\]

i.e., \( P(x) - \varepsilon < f(x) < P(x) + \varepsilon \), for all \( x \) in \( [a, b] \).

How to find such a polynomial \( P \)? What is the maximum error in approximating \( f(x^*) \) by \( P(x^*) \)?
3.1 Lagrange Polynomials

For two distinct points \((x_0, y_0)\) and \((x_1, y_1)\), there is a unique polynomial \(P_1\) of degree 1 such that \(P_1(x_0) = y_0\) and \(P_1(x_1) = y_1\). It can be verified that

\[
P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0},
\]

whose graph is the straight line joining \((x_0, y_0)\) and \((x_1, y_1)\). We can extend this idea to \(n + 1\) distinct points:

**Theorem.** Suppose \((x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\) are \(n + 1\) distinct points where \(x_0, x_1, \ldots, x_n\) are distinct and \(f\) is a function such that \(f(x_i) = y_i, \ i = 0, 1, \ldots, n\). Then there is a unique polynomial \(P_n\) of degree at most \(n\) such that \(P_n(x_i) = f(x_i), \ i = 0, 1, \ldots, n\).

**Proof.** (Sketch) Consider a polynomial \(P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n\) for which \(P_n(x_i) = f(x_i) = y_i, \ i = 0, 1, \ldots, n\). It gives us a system of \(n + 1\) equations in \(n + 1\) variables \(a_0, a_1, \ldots, a_n\):

\[
a_0 + a_1x_i + a_2x_i^2 + \cdots + a_nx_i^n = y_i, \ i = 0, 1, \ldots, n.
\]

Its matrix form is \(A\overrightarrow{x} = \overrightarrow{b}\) where \(\overrightarrow{x} = [a_0, \ldots, a_n]^T\), \(\overrightarrow{b} = [y_0, \ldots, y_n]^T\), and \(A\) is the Vandermonde matrix

\[
A = \begin{bmatrix}
1 & x_0 & \cdots & x_0^n \\
1 & x_1 & \cdots & x_1^n \\
\vdots & \vdots & \ddots & \vdots \\
1 & x_n & \cdots & x_n^n
\end{bmatrix}.
\]

The determinant \(\det(A) = \prod_{0 \leq i < j \leq n} (x_j - x_i) \neq 0\) as \(x_0, x_1, \ldots, x_n\) are distinct. So \(A\) is invertible and we have a unique solution \([a_0, \ldots, a_n]^T = \overrightarrow{x} = A^{-1}\overrightarrow{b}\) giving a unique polynomial \(P_n\) of degree at most \(n\). \(\square\)

Note that there are **infinitely many** polynomials \(P\) of degree **more than** \(n\) for which \(P(x_i) = f(x_i) = y_i, \ i = 0, 1, \ldots, n\). One construction of the polynomial \(P_n\) of degree at most \(n\) such that \(P_n(x_i) = f(x_i) = y_i, \ i = 0, 1, \ldots, n\) is given by Joseph Lagrange:

**Lagrange Polynomial:** \(P_n(x) = y_0L_0(x) + y_1L_1(x) + \cdots + y_nL_n(x),\) where \(L_i\) is given by

\[
L_i(x) = \prod_{\substack{j=0, \ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} = \frac{(x - x_0) \cdots (x - x_{i-1}) (x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0) \cdots (x_i - x_{i-1}) (x_i - x_{i+1}) \cdots (x_i - x_n)}.
\]

Note that \(L_i(x_i) = 1\) and \(L_i(x_j) = 0\) for all \(j \neq i\). Thus \(P_n(x_i) = y_i = f(x_i), \ i = 0, 1, \ldots, n\).
Example. Approximate $f(2)$ by constructing the Lagrange polynomial $P_2$ of degree 2 from the following data:

<table>
<thead>
<tr>
<th>$x$</th>
<th>$f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>4.39</td>
</tr>
<tr>
<td>4</td>
<td>5.54</td>
</tr>
</tbody>
</table>

Solution. $P_2$ is given by $P_2(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x)$, where

$L_0(x) = \frac{(x-3)(x-4)}{(1-3)(1-4)} = \frac{(x-3)(x-4)}{6},$
$L_1(x) = \frac{(x-1)(x-4)}{(3-1)(3-4)} = \frac{(x-1)(x-4)}{-2},$
$L_2(x) = \frac{(x-1)(x-3)}{(4-1)(4-3)} = \frac{(x-1)(x-3)}{3}.$

Thus

$P_2(x) = y_0L_0(x) + y_1L_1(x) + y_2L_2(x)$

$= 0 \frac{(x-3)(x-4)}{6} + 4.39 \frac{(x-1)(x-4)}{-2} + 5.54 \frac{(x-1)(x-3)}{3}$

$= 4.39 \frac{(x-1)(x-4)}{-2} + 5.54 \frac{(x-1)(x-3)}{3}.$

Thus $f(2) \approx P_2(2) = 2.54.$
**Maximum Error:** If a function $f$ that has continuous $f^{(n+1)}$ on $[x_0, x_n]$ is interpolated by the Lagrange polynomial $P_n$ using $n + 1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$, then the error is given by the following for each $x \in [x_0, x_n]$:

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)(x - x_1) \cdots (x - x_n),$$

where $\xi \in (x_0, x_n)$ depends on $x$.

**Proof.** (Sketch) If $x = x_i$, $i = 0, 1, \ldots, n$, then

$$f(x_i) - P_n(x_i) = 0 = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x_i - x_0)(x_i - x_1) \cdots (x_i - x_n).$$

For a fixed $x \neq x_i$, $i = 0, 1, \ldots, n$, define a function $g$ on $[x_0, x_n]$ by

$$g(t) = f(t) - P_n(t) - [f(x) - P_n(x)] \prod_{j=0}^{n} \frac{(t - x_j)}{(x - x_j)}.$$

Verify that $g^{(n+1)}$ is continuous on $[x_0, x_n]$ and $g$ is zero at $x, x_0, x_1, \ldots, x_n$. Then by the Generalized Rolle’s Theorem, $g^{(n+1)}(\xi) = 0$ for some $\xi \in (x_0, x_n)$ which implies (steps skipped)

$$f^{(n+1)}(\xi) - 0 - [f(x) - P_n(x)] \frac{(n+1)!}{\prod_{j=0}^{n} (x - x_j)} = 0.$$

Now solving for $f(x)$, we get

$$f(x) - P_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0)(x - x_1) \cdots (x - x_n). \quad \square$$

So the maximum error is given by

$$|f(x) - P_n(x)| \leq \frac{MK}{(n+1)!}$$

where $M = \max_{x \in [x_0, x_n]} |f^{(n+1)}(x)|$ and $K = \max_{x \in [x_0, x_n]} |(x - x_0)(x - x_1) \cdots (x - x_n)|$.

This error bound does not have a practical application as $f$ is unknown. But it shows that the error decreases when we take more points $n$ (most of the times!).

Note that if $f$ is a polynomial of degree at most $n$, then $f^{(n+1)} = 0$ and consequently $M = 0$ giving no error.
Example. Find the maximum error in approximating \( f(x) = 4 \ln x \) on \([1, 4]\) by the Lagrange polynomial \( P_2 \) using points \( x = 1, 3, 4 \).

Solution. Since \( |f'''(x)| = 8/x^3 \) is decreasing on \([1, 4]\), \( M = \max_{x \in [1,4]} |f'''(x)| = |f'''(1)| = 8 \).

Now we find extremum values of \( g(x) = (x - 1)(x - 3)(x - 4) = x^3 - 8x^2 + 19x - 12 \).

\[ g'(x) = 3x^2 - 16x + 19 = 0 \implies x = (8 \pm \sqrt{7})/3. \]

Note that \((8 \pm \sqrt{7})/3 \in [1, 4]\) and

\[ g((8 + \sqrt{7})/3) = (20 - 14\sqrt{7})/27 \]
\[ g((8 - \sqrt{7})/3) = (20 + 14\sqrt{7})/27 \]
\[ g(1) = 0 \]
\[ g(4) = 0. \]

Since \(|g((8 - \sqrt{7})/3)| > |g((8 + \sqrt{7})/3)|\),

\[ K = \max_{x \in [1,4]} |(x - 1)(x - 3)(x - 4)| = (20 + 14\sqrt{7})/27. \]

Thus the maximum error is

\[ \frac{MK}{(n+1)!} = \frac{8 \cdot (20 + 14\sqrt{7})/27}{3!} = 2.81. \]

The last example has the table for \( f(x) = 4 \ln x \). So \( f(2) = 4 \ln 2 = 2.77 \) and the absolute error is \(|P_2(2) - f(2)| = |2.54 - 2.77| = 0.23 \). But approximating \( f(x) \) by \( P_2(x) \) for any \( x \in [1, 4] \) will have the maximum error 2.81.

Note that another construction of the unique polynomial \( P_n \) of degree at most \( n \) such that \( P_n(x_i) = f(x_i), \ i = 0, 1, \ldots, n \) is given by Issac Newton:

\[ P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0)\cdots(x - x_{n-1}), \]

where \( a_i, \ i = 0, \ldots, n \) are found by Divided Differences.
3.2 Cubic Spline Interpolation

There are some problems in approximating an unknown function $f$ on $[x_0, x_n]$ by a single polynomial $P_n$ using $n + 1$ points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$:

The values $P_n(x)$ may oscillate near the end points (Runge’s phenomenon) and then the maximum error $|f(x) - P_n(x)| \to \infty$ as $n \to \infty$, i.e., the error grows for more points. For example, consider the Runge’s function $f(x) = 1/(1 + 25x^2)$ on $[-1, 1]$.

To avoid these problems, we use a piecewise interpolating polynomial $S$ on the intervals $[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]$. This is called **piecewise polynomial interpolation**. But in the piecewise linear interpolation, the piecewise linear polynomial $S$ is not smooth i.e., $S'(x)$ is not continuous at $x_0, x_1, \ldots, x_n$. If $S$ is piecewise quadratic, then we get the smoothness but it does not work when $S'(x_0)$ and $S'(x_n)$ are given. So we will study piecewise cubic interpolation.

A spline $S$ of degree $k$ is a piecewise polynomial of degree at most $k$ such that $S^{(k-1)}$ is continuous. A **cubic spline** is a piecewise cubic with continuous first and second derivatives:
1. \( S(x) = S_i(x) \) on \([x_{i-1}, x_i]\), for \( i = 1, \ldots, n \)

2. \( S_i(x_i) = y_i = S_{i+1}(x_i) \) for \( i = 1, \ldots, n-1 \),
   
   \( S_1(x_0) = y_0 \), and \( S_n(x_n) = y_n \).

3. \( S'_i(x_i) = S'_{i+1}(x_i) \) for \( i = 1, \ldots, n-1 \)

4. \( S''_i(x_i) = S''_{i+1}(x_i) \) for \( i = 1, \ldots, n-1 \)

A cubic spline \( S \) is called **natural** if \( S''(x_0) = S''(x_n) = 0 \) and **clamped** if \( S'(x_0) \) and \( S'(x_n) \) are provided.

**Example.** Approximate \( f(2) \) by constructing a natural cubic spline \( S \) from the following data:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

**Solution.** We define \( S \) piecewise on \([1, 3]\) as follows:

\[
S(x) = \begin{cases} 
  S_1(x) = a_1 + b_1(x - 1) + c_1(x - 1)^2 + d_1(x - 1)^3 & \text{if } x \in [1, 3] \\
  S_2(x) = a_2 + b_2(x - 3) + c_2(x - 3)^2 + d_2(x - 3)^3 & \text{if } x \in [3, 4]. 
\end{cases}
\]

Using the conditions of cubic spline together with the natural boundary conditions, we get a system of 8 equations in 8 variables:

- \( S_1(1) = -1 \quad \implies a_1 = -1 \)
- \( S_1(3) = 0 \quad \implies a_1 + 2b_1 + 4c_1 + 8d_1 = 0 \)
- \( S_2(3) = 0 \quad \implies a_2 = 0 \)
- \( S_2(4) = 1 \quad \implies a_2 + b_2 + c_2 + d_2 = 1 \)
- \( S'_1(3) = S'_2(3) \quad \implies b_1 + 4c_1 + 12d_1 = b_2 \)
- \( S''_1(3) = S''_2(3) \quad \implies 2c_1 + 12d_1 = 2c_2 \)
- \( S'_1(1) = 0 \quad \implies 2c_1 = 0 \)
- \( S''_1(4) = 0 \quad \implies 2c_2 + 6d_2 = 0 \)

Solving (using linear algebra) we get the unique solution

\[
(a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2) = \left(-1, \frac{1}{3}, 0, \frac{1}{24}, 0, \frac{5}{6}, \frac{1}{4}, -\frac{1}{12}\right).
\]

Thus

\[
S(x) = \begin{cases} 
  -1 + \frac{1}{3}(x - 1) + \frac{1}{24}(x - 1)^3 & \text{if } x \in [1, 3] \\
  \frac{5}{6}(x - 3) + \frac{1}{4}(x - 3)^2 - \frac{1}{12}(x - 3)^3 & \text{if } x \in [3, 4]. 
\end{cases}
\]

Thus \( f(2) \approx S(2) = -\frac{15}{24}. \)
Note in the preceding example that we got a unique solution and hence a unique natural cubic spline. But we did not just get lucky because this is the case always:

**Theorem.** There is a unique natural and clamped cubic spline passing through \( n + 1 \) points \((x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\), \( n \geq 2 \).

**Proof.** (Sketch) We have total \( 4n \) unknown coefficients \( a_i, b_i, c_i, d_i, i = 1, \ldots, n \) in \( n \) cubics \( S_1, \ldots, S_n \). Using \( 4n - 2 \) conditions of cubic spline together with 2 natural or clamped boundary conditions, we get a system of \( 4n \) equations in \( 4n \) variables. Using algebraic substitutions and linear algebra (steps skipped), we get a unique solution.

Note that a clamped cubic spine usually gives better approximation than that of a natural cubic spine near the endpoints of \([x_0, x_n]\).
4 Numerical Differentiation and Integration

In this chapter we will learn numerical methods for derivative and integral of a function.

4.1 Numerical Differentiation

In this section we will numerically find $f'(x)$ evaluated at $x = x_0$. We need numerical techniques for derivatives when $f'(x)$ has a complicated expression or $f(x)$ is not explicitly given. By the limit definition of derivative,

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}.$$  

So when $h > 0$ is small, we have

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h},$$

which is called the two-point forward difference formula (FDF). Similarly the two-point backward difference formula (BDF) is

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}.$$  

Taking the average of the FDF and BDF, we get the two-point centered difference formula (CDF) is

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}.$$  

![Graph showing numerical differentiation formulas](image)

Note that CDF gives better accuracy than FDF and BDF (explained later). But CDF does not work if $f(x)$ is not known in one side of $x_0$. All the difference formulas suffer from round-off errors when $h$ is too small.
Example. $f(x) = x^2 e^x$. Approximate $f'(1)$ using the FDF, BDF, and CDF with $h = 0.2$.

Solution.

Two-point FDF: $f'(1) \approx \frac{f(1 + 0.2) - f(1)}{0.2} = \frac{4.78 - 2.71}{0.2} = 10.35$

Two-point BDF: $f'(1) \approx \frac{f(1) - f(1 - 0.2)}{0.2} = \frac{2.71 - 1.42}{0.2} = 6.45$

Two-point CDF: $f'(1) \approx \frac{f(1 + 0.2) - f(1 - 0.2)}{2(0.2)} = \frac{4.78 - 1.42}{0.4} = 8.4$

Analytically we know $f'(1) = 3e$. So the absolute errors are $|10.35 - 3e| = 2.19$, $|6.45 - 3e| = 1.7$, and $|8.4 - 3e| = 0.24$ respectively. So CDF gives the least error.

Errors in finite difference formulas: By the Taylor’s theorem on $f$ about $x_0$, we get

$$f(x) = f(x_0) + \frac{f'(x_0)}{1!}(x - x_0) + \frac{f''(\xi)}{2!}(x - x_0)^2.$$  

Plugging $x = x_0 + h$, we get

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f''(\xi_1)}{2!}h^2,$$

for some $\xi_1 \in (x_0, x_0 + h)$. Solving for $f'(x_0)$, we get

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{f''(\xi_1)}{2}h.$$  

The maximum error in FDF is

$$\frac{h}{2} \max_{x \in (x_0, x_0 + h)} |f''(x)|.$$  

So the error in FDF is $O(h)$ (i.e., absolute error $\leq ch$ for some $c > 0$). It means small step size $h$ results in more accurate derivative. We say FDF is first-order accurate. Similarly BDF is also first-order accurate with the maximum error

$$\frac{h}{2} \max_{x \in (x_0 - h, x_0)} |f''(x)|.$$  

For CDF, note that

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 + \frac{f'''(\xi_1)}{3!}h^3,$$

$$f(x_0 - h) = f(x_0) - \frac{f'(x_0)}{1!}h + \frac{f''(x_0)}{2!}h^2 - \frac{f'''(\xi_2)}{3!}h^3,$$

for some $\xi_1 \in (x_0, x_0 + h)$ and $\xi_2 \in (x_0 - h, x_0)$. Subtracting we get,
\[ f(x_0 + h) - f(x_0 - h) = 2f'(x_0)h + \frac{f'''(\xi_1) + f'''(\xi_2)}{6} h^3 \]
\[ \frac{f(x_0 + h) - f(x_0 - h)}{2h} = f'(x_0) + \frac{f'''(\xi_1) + f'''(\xi_2)}{12} h^2 \]
\[ f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{f'''(\xi_1) + f'''(\xi_2)}{12} h^2. \]

Assuming continuity of \( f''' \) and using the IVT on \( f''' \), we get
\[ f'''(\xi) = \frac{f'''(\xi_1) + f'''(\xi_2)}{2}, \]
for some \( \xi \in (\xi_2, \xi_1) \subset (x_0 - h, x_0 + h). \) Thus
\[ f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} - \frac{f'''(\xi)}{6} h^2. \]

The maximum error in CDF is
\[ \frac{h^2}{6} \max_{x \in (x_0-h, x_0+h)} |f'''(x)|. \]

So the error in CDF is \( O(h^2) \), i.e., CDF is second-order accurate which is better than first-order accurate as \( h^2 << h \) for small \( h > 0 \).

**Example.** Consider \( f(x) = x^2e^x \). Find the maximum error in approximating \( f'(1) \) by the FDF, BDF, and CDF with \( h = 0.2 \).

*Solution.* \( f''(x) = (x^2 + 4x + 2)e^x \) and \( f'''(x) = (x^2 + 6x + 6)e^x \) are increasing functions for \( x > 0 \). So \( \max_{x \in (1,1.2)} |f''(x)| = |f''(1.2)| = 27.3 \).

- Maximum error in two-point FDF: \( \frac{0.2}{2} \max_{x \in (1,1.2)} |f''(x)| = 0.1|f''(1.2)| = 2.73 \)
- Maximum error in two-point BDF: \( \frac{0.2}{2} \max_{x \in (0,0.8,1)} |f''(x)| = 0.1|f''(1)| = 1.9 \)
- Maximum error in two-point CDF: \( \frac{(0.2)^2}{6} \max_{x \in (0.8,1.2)} |f''(x)| = \frac{0.04}{6}|f''(1.2)| = 0.32 \)

**Derivative from Lagrange polynomial:** If \( f \) is not explicitly given but we know \((x_i, f(x_i))\) for \( i = 0, 1, \ldots, n \), then \( f \) is approximated by the Lagrange polynomial:
\[ f(x) = \sum_{i=0}^{n} f(x_i)L_i(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \]
where $\xi \in (x_0, x_n)$ and $L_i(x) = \prod_{j=0}^{n} \frac{(x - x_j)}{(x_i - x_j)}$. Differentiating both sides and evaluating at $x = x_j$, we get (steps skipped but note that $\frac{d}{dx} \prod_{i=0}^{n}(x - x_i) \bigg|_{x=x_j} = \prod_{i=0}^{n} (x_j - x_i)$)

$$f'(x_j) = \sum_{i=0}^{n} f(x_i) L'_i(x_j) + \frac{f(n+1)(\xi)}{(n+1)!} \prod_{i=0, i \neq j}^{n} (x_j - x_i).$$

If the points $x_0, x_1, \ldots, x_n$ are equally-spaced, i.e., $x_j = x_0 + jh$, then we get

$$f'(x_j) = \sum_{i=0}^{n} f(x_i) L'_i(x_j) + \frac{f(n+1)(\xi)}{(n+1)!} O(h^n). \quad (1)$$

It can be verified that two-point FDF and BDF are obtained from (1) using $n = 1$. Similarly $n = 2$ gives the three-point FDF and BDF and two-point CDF whose errors are $O(h^2)$:

Three-point FDF: $f'(x_0) \approx \frac{-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)}{2h}$

Three-point BDF: $f'(x_0) \approx \frac{3f(x_0) - 4f(x_0 - h) + f(x_0 - 2h)}{2h}$

Example. From the following table approximate $f'(1)$ by the three-point FDF and BDF.

<table>
<thead>
<tr>
<th>x</th>
<th>0.6</th>
<th>0.8</th>
<th>1</th>
<th>1.2</th>
<th>1.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(x)</td>
<td>0.65</td>
<td>1.42</td>
<td>2.71</td>
<td>4.78</td>
<td>7.94</td>
</tr>
</tbody>
</table>

Solution. Here $h = 0.2$.

Three-point FDF: $f'(1) \approx \frac{-3f(1) + 4f(1 + 0.2) - f(1 + 2(0.2))}{2(0.2)} = 7.62$

Three-point BDF: $f'(1) \approx \frac{3f(1) - 4f(1 - 0.2) + f(1 - 2(0.2))}{2(0.2)} = 7.75$

Note that the table is given for $f(x) = x^2 e^x$. So $f'(1) = 3e$. Then the absolute errors are $|7.62 - 3e| = 0.53$ and $|7.75 - 3e| = 0.4$ respectively. Notice that three-point FDF and BDF give less error than two-point FDF and BDF respectively.
4.2 Elements of Numerical Integration

Sometimes it is hard to calculate a definite integral analytically. For example, \( \int_0^1 e^{x^2} \, dx \). To approximate such an integral we break \([a, b]\) into \(n\) subintervals \([x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]\) where \(x_i = a + ih\) and \(h = (b - a)/n\). Then we approximate the integral by a finite sum given by a quadrature rule (or, quadrature formula):

\[
\int_a^b f(x) \, dx \approx \sum_{i=0}^n c_i f(x_i).
\]

A quadrature rule you have seen before is the **Midpoint Rule**:

\[
\int_a^b f(x) \, dx \approx (b - a)f\left(\frac{a + b}{2}\right).
\]

It approximates the area given by \(\int_a^b f(x) \, dx\) by the area of the rectangle with length \((b - a)\) and width \(f\left(\frac{a + b}{2}\right)\).

Let’s discuss other quadrature rules. Recall that we can approximate \(f(x)\) by the Lagrange polynomial \(P_n(x)\) of degree \(n\) using \(n + 1\) points \(a = x_0, x_1, \ldots, x_n = b\):

\[
f(x) \approx \sum_{i=0}^n f(x_i)L_i(x),
\]

where \(L_i(x) = \prod_{j=0}^n \frac{(x - x_j)}{(x_i - x_j)}\). Integrating both sides, we get

\[
\int_a^b f(x) \, dx \approx \int_a^b \sum_{i=0}^n f(x_i)L_i(x) \, dx = \sum_{i=0}^n f(x_i) \left[ \int_a^b L_i(x) \, dx \right]
\]
\[ \int_a^b f(x) \, dx \approx \sum_{i=0}^n c_i f(x_i), \]

where \( c_i = \int_a^b L_i(x) \, dx \). We will discuss the quadrature rules given by \( n = 1 \) and 2.

For \( n = 1 \), we have \( n + 1 = 2 \) points \( a = x_0, x_1 = b \) and then

\[
\begin{align*}
f(x) & \approx P_1(x) = f(x_0) \frac{(x - x_1)}{(x_0 - x_1)} + f(x_1) \frac{(x - x_0)}{(x_1 - x_0)} = -f(a) \frac{(x - b)}{(b - a)} + f(b) \frac{(x - a)}{(b - a)} \\
\Rightarrow \int_a^b f(x) \, dx & \approx -\frac{f(a)}{b-a} \int_a^b (x-b) \, dx + \frac{f(b)}{b-a} \int_a^b (x-a) \, dx \\
& = -\frac{f(a)}{b-a} \left[ \frac{(x-b)^2}{2} \right]_a^b + \frac{f(b)}{b-a} \left[ \frac{(x-a)^2}{2} \right]_a^b \\
& = (b-a) \left[ \frac{f(a) + f(b)}{2} \right]
\end{align*}
\]

So we get the **Trapezoidal Rule**:

\[
\int_a^b f(x) \, dx \approx (b-a) \left[ \frac{f(a) + f(b)}{2} \right].
\]

It approximates the area given by \( \int_a^b f(x) \, dx \) by the area of the trapezoid with height \( (b-a) \) and bases \( f(a) \) and \( f(b) \).

The error in the trapezoidal rule is the integral of the error term for the Lagrange polynomial:

\[
E_T = \int_a^b \frac{f''(\xi(x))}{2!} (x-a)(x-b) \, dx.
\]

By the Weighted MVT (where \( f''(\xi(x)) \) is continuous and \( (x-a)(x-b) \) does not change sign in \([a,b]\)), we get a constant \( c \in (a,b) \) such that
\[ E_T = \frac{f''(c)}{2} \int_a^b (x - a)(x - b) \, dx = -\frac{f''(c)(b - a)^3}{12}. \]

Similarly for \( n = 2 \), we have \( n + 1 = 3 \) points \( a = x_0, x_1 = (a + b)/2, x_2 = b \) and then

\[
f(x) \approx P_2(x) = f(x_0) \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + f(x_1) \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + f(x_2) \frac{(x - x_0)(x - x_1)}{(x_2 - x_1)(x_2 - x_1)}.\]

Integrating we get the **Simpson’s Rule**:

\[
\int_a^b f(x) \, dx \approx \frac{b - a}{6} \left[ f(a) + 4f \left( \frac{a + b}{2} \right) + f(b) \right]
\]

where the error in the Simpson’s Rule (obtained from the Taylor polynomial \( T_3(x) \) of \( f \) about \( x = (a + b)/2 \) with the error term) is

\[
E_S = -\frac{(b - a)^5}{90 \cdot 24} f^{(4)}(c).
\]

Note from \( E_T \) that if \( f(x) \) is a polynomial of degree at most 1, then \( f'' = 0 \) and consequently \( E_T = 0 \). So the trapezoidal rule gives the exact integral. Similarly if \( f(x) \) is a polynomial of degree at most 3, then \( E_S = 0 \) and consequently the Simpson’s rule gives the exact integral.

**Example.** Approximate \( \int_0^2 x^3 \, dx \) by the Midpoint Rule, Trapezoidal Rule, Simpson’s Rule.

**Solution.** First of all let’s find the exact integral:

\[
\int_0^2 x^3 \, dx = \frac{x^4}{4} \bigg|_0^2 = 4.
\]

**Midpoint:**

\[
\int_0^2 x^3 \, dx \approx (b - a)f \left( \frac{a + b}{2} \right) = (2 - 0) \left( \frac{0 + 2}{2} \right)^3 = 2
\]

**Trapezoidal:**

\[
\int_0^2 x^3 \, dx \approx (b - a) \left[ \frac{f(a) + f(b)}{2} \right] = (2 - 0) \left[ \frac{0 + 2^3}{2} \right] = 8
\]

**Simpson’s:**

\[
\int_0^2 x^3 \, dx \approx \frac{(b - a)}{6} \left[ f(a) + 4f \left( \frac{a + b}{2} \right) + f(b) \right] = \frac{(2 - 0)}{6} \left[ 0 + 4 \cdot 1^3 + 2^3 \right] = 4
\]

The Simpson’s Rule gives the best approximation which turns out to be the exact integral.

Note that the error of the Midpoint Rule is always half of that of the Trapezoidal Rule. Because the Midpoint Rule is obtained by integrating the Taylor polynomial \( T_1(x) \) of \( f \) about \( x = (a + b)/2 \) and integrating its remainder term, we can show that

\[
E_M = \frac{(b - a)^3}{24} f''(c).
\]
4.3 Composite Numerical Integration

Approximating $\int_a^b f(x) \, dx$ by quadrature rules like trapezoidal, Simpson’s will give large error when the interval $[a, b]$ is large. We can modify those rules by using $n + 1$ points instead of 2 or 3 points. Then the Lagrange polynomial of degree $n$ might give large error near the end points for large $n$. So we use a composite quadrature rule that breaks $[a, b]$ into $n$ subintervals $[x_0, x_1], [x_1, x_2], \ldots, [x_{n-1}, x_n]$ ($x_i = a + ih$ and $h = (b - a)/n$) and approximates the integral by applying quadrature rules on each subinterval and adding them up:

\[
\int_a^b f(x) \, dx = \int_{x_0}^{x_1} f(x) \, dx + \int_{x_1}^{x_2} f(x) \, dx + \cdots + \int_{x_{n-1}}^{x_n} f(x) \, dx.
\]

Applying trapezoidal rule on each subinterval $[x_{i-1}, x_i]$, we get

\[
\int_{x_{i-1}}^{x_i} f(x) \, dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x) \, dx \approx \sum_{i=1}^{n} \frac{h}{2} \left( f(x_{i-1}) + f(x_i) \right)
\]

\[
= \frac{h}{2} \left[ f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n) \right].
\]

So the Composite Trapezoidal Rule is

\[
\int_a^b f(x) \, dx \approx \frac{h}{2} \left[ f(x_0) + 2f(x_1) + \cdots + 2f(x_{n-1}) + f(x_n) \right].
\]

Similarly the Composite Midpoint Rule is

\[
\int_a^b f(x) \, dx \approx h \sum_{i=1}^n f \left( \frac{x_{i-1} + x_i}{2} \right).
\]
For the **Composite Simpson’s Rule**, we take even \( n \) and apply simple Simpson’s Rule to the subintervals \([x_0, x_2], [x_2, x_4], \ldots, [x_{n-2}, x_n]\):

\[
\int_a^b f(x) \, dx = \sum_{i=1}^{n/2} \int_{x_{2i-2}}^{x_{2i}} f(x) \, dx
\]

\[
= \frac{h}{3} \left[ f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i}) \right]
\]

\[
= \frac{h}{3} \left[ (f(x_0)+4f(x_1)+f(x_2)) + (f(x_2)+4f(x_3)+f(x_4)) + \cdots + (f(x_{n-2})+4f(x_{n-1})+f(x_n)) \right]
\]

\[
= \frac{h}{3} \left[ f(x_0)+f(x_n) + 4 \sum_{i=1}^{n/2} f(x_{2i-1}) + 2 \sum_{i=1}^{(n-2)/2} f(x_{2i}) \right]
\]

**Example.** Approximate \( \int_0^2 e^x \, dx \) using 4 subintervals in (a) Composite Trapezoidal Rule, (b) Composite Midpoint Rule, (c) Composite Simpson’s Rule.

**Solution.** First of all let’s find the exact integral: \( \int_0^2 e^x \, dx = e^x \bigg|_0^2 = e^2 - 1 \approx 6.389. \)

\( n = 4 \implies h = (2 - 0)/4 = 0.5 \) and the 4 subintervals are \([0, 0.5], [0.5, 1], [1, 1.5], [1.5, 2]\).

- **CTR:** \( \int_0^2 e^x \, dx \approx \frac{0.5}{2} \left[ e^0 + 2e^{0.5} + 2e^1 + 2e^{1.5} + e^2 \right] = 6.52 \)
- **CMR:** \( \int_0^2 e^x \, dx \approx 0.5 \left[ e^{0.25} + e^{0.75} + e^{1.25} + e^{1.75} \right] = 6.32 \)
- **CSR:** \( \int_0^2 e^x \, dx \approx \frac{0.5}{3} \left[ e^0 + 4e^{0.5} + 2e^1 + 4e^{1.5} + e^2 \right] = 6.39 \)

The error in the composite trapezoidal rule (using \( n \) subintervals) is given by

\[
E_{T_n} = -\sum_{i=1}^{n} f''(c_i) \frac{(x_i - x_{i-1})^3}{12} = -\frac{h^3}{12} \sum_{i=1}^{n} f''(c_i).
\]

Assuming continuity of \( f'' \) on \((a, b)\), by the IVT we can find \( c \in (a, b) \) such that

\[
\frac{1}{n} \sum_{i=1}^{n} f''(c_i) = f''(c).
\]
Thus \( \sum_{i=1}^{n} f''(c_i) = nf''(c) \) and then

\[
E_T_n = -\frac{nh^3}{12} f''(c).
\]

Note that \( n = (b - a)/h \). Then the error for the composite trapezoidal rule becomes:

\[
E_T_n = -\frac{(b-a)}{12} h^2 f''(c).
\]

Similarly we get errors in the composite midpoint and Simpson’s rule:

\[
E_M_n = \frac{(b-a)}{24} h^2 f''(c) \quad E_S_n = -\frac{(b-a)}{180} h^4 f^{(4)}(c)
\]

Note that since errors are \( O(h^2) \) and \( O(h^4) \), small step sizes lead to more accurate integral.

**Example.** Find the step size \( h \) and the number of subintervals \( n \) required to approximate \( \int_{0}^{2} e^x \, dx \) correct within \( 10^{-2} \) using (a) Composite Trapezoidal Rule, (b) Composite Midpoint Rule, (c) Composite Simpson’s Rule.

**Solution.** Note \( f''(x) = f^{(4)}(x) = e^x \) which have the maximum absolute value \( e^2 \) on \([0, 2]\).

\[
|E_T_n| = \left| -\frac{(b-a)}{12} h^2 f''(c) \right| \leq \frac{(b-a)}{12} h^2 \cdot \max_{[0,2]} |f''(x)| = \frac{(2-0)}{12} \left( \frac{2-0}{n} \right)^2 \cdot e^2 < 10^{-2}
\]

\( \Rightarrow n > \sqrt{200e^2/3} = 22.19 \)

Thus for the CTR we need \( n = 23 \) and \( h = (2-0)/23 = 2/23 \approx 0.087 \).

Similarly for the CMR we need \( n = 16 \) and \( h = (2-0)/16 = 0.125 \),

and for the CSR we need \( n = 4 \) and \( h = (2-0)/4 = 0.5 \).

---

**Algorithm Composite-Simpson’s**

**Input:** function \( f \), interval \([a, b]\), an even number \( n \) of subintervals

**Output:** an approximation of \( \int_{a}^{b} f(x) \, dx \)

set \( h = (b-a)/n; \)
set \( I = f(a) + f(b); \)
for \( i = 1 \) to \( n/2 \)
\[
I = I + 4 \cdot f(a + (2i - 1) \cdot h)
\]
end for
for \( i = 1 \) to \( (n-2)/2 \)
\[
I = I + 2 \cdot f(a + 2i \cdot h)
\]
end for
return \( I \cdot h/3 \)
5 Differential Equations

In this chapter we numerically solve the following IVP (initial value problem):

\[ \frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = c \]  \hspace{1cm} (2)

Instead of finding \( y = y(t) \) on \([a, b]\), we break \([a, b]\) into \(n\) subintervals \([t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]\) and approximate \( y(t_i), \quad i = 0, 1, \ldots, n \). But if we need \( y(t) \), it can be approximated by the Lagrange polynomial \( P_n \) using \( t_0, t_1, \ldots, t_n \).

Before approximating a solution \( y = y(t) \), we must ask if (2) has a solution and it is unique on \([a, b]\). The answer is given by the Existence and Uniqueness Theorem:

**Theorem 5.1.** The IVP (2) has a unique solution \( y(t) \) on \([a, b]\) if

1. \( f \) is continuous on \( D = \{(t, y) | a \leq t \leq b, -\infty < y < \infty\} \), and
2. \( f \) satisfies a Lipschitz condition on \( D \) with constant \( L \):

\[ |f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|, \quad \text{for all} \quad (t, y_1), (t, y_2) \in D. \]

When we approximate \( y(t_i), \quad i = 0, 1, \ldots, n \) for the unique solution \( y(t) \), we might commit some round-off errors. So we ask if the IVP (2) is well-posed: a small change in the problem (i.e., small change in \( f, c \)) gives a small change in the solution.

It can be proved that the IVP (2) is well-posed if \( f \) satisfies a Lipschitz condition on \( D \). Also note that if \( |f_y(t, y)| \leq L \) on \( D \), then \( f \) satisfies a Lipschitz condition on \( D \) with constant \( L \).

5.1 Euler’s Method

We break \([a, b]\) into \(n\) subintervals \([t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]\) where \( t_i = a + ih \) and \( h = (b - a)/n \). The Euler’s Method finds \( y_0, y_1, \ldots, y_n \) such that \( y_i \approx y(t_i), \quad i = 0, 1, \ldots, n \):

\[
\begin{align*}
y_0 &= c \\
y_{i+1} &= y_i + hf(t_i, y_i), \quad i = 0, 1, \ldots, n - 1.
\end{align*}
\]

To justify the iterative formula, use Taylor’s theorem on \( y \) about \( t = t_i \):

\[
\begin{align*}
y(t) &= y(t_i) + (t - t_i)y'(t_i) + \frac{(t - t_i)^2}{2} y''(\xi_i) \\
\Rightarrow y(t_{i+1}) &= y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i) \\
&= y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2} y''(\xi_i) \\
\Rightarrow y(t_{i+1}) &\approx y_i + hf(t_i, y_i) =: y_{i+1}
\end{align*}
\]
**Example.** Use Euler’s method with step size $h = 0.5$ to approximate the solution of the following IVP:

$$\frac{dy}{dt} = t^2 - y, \ 0 \leq t \leq 3, \ y(0) = 1$$

**Solution.** We have $h = 0.5$, $t_0 = 0$, $y_0 = 1$ and $f(t, y) = t^2 - y$. So

$$y_{i+1} = y_i + 0.5(t_i^2 - y_i) = 0.5(t_i^2 + y_i).$$

$t_1 = 0 + 1 \cdot 0.5 = 0.5$, \quad $y_1 = 0.5(t_0^2 + y_0) = 0.5(0 + 1) = 0.5$

$t_2 = 0 + 2 \cdot 0.5 = 1$, \quad $y_2 = 0.5(t_1^2 + y_1) = 0.5(0.25 + 0.5) = 0.375$ etc.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$t_i$</th>
<th>$y_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.375</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>0.6875</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.46875</td>
</tr>
<tr>
<td>5</td>
<td>2.5</td>
<td>2.734375</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>4.4921875</td>
</tr>
</tbody>
</table>

**Geometric Interpretation:** The tangent line to the solution $y = y(t)$ at the point $(t_0, y_0)$ has slope $\frac{dy}{dt}(t_0, y_0) = f(t_0, y_0)$. So an equation of the tangent line is

$$y = y_0 + (t - t_0)f(t_0, y_0).$$

If $t_1$ is close to $t_0$, then $y_1 = y_0 + (t_1 - t_0)f(t_0, y_0) = y_0 + hf(t_0, y_0)$ would be a good approximation to $y(t_1)$. Similarly if $t_2$ is close to $t_1$, then $y(t_2) \approx y_2 = y_1 + hf(t_1, y_1)$. 

37
Maximum error: Suppose $D = \{(t,y) \mid a \leq t \leq b, -\infty < y < \infty\}$ and $f$ satisfies a Lipschitz condition on $D$ with constant $L$. Suppose $y(t)$ is the unique solution to (2) where $|y''(t)| \leq M$ for all $t \in [a,b]$. Then for the approximation $y_i$ of $y(t_i)$ by the Euler’s method with step size $h$, we have

$$|y(t_i) - y_i| \leq \frac{hM}{2L} \left[-1 + e^{L(t_i - a)}\right], \quad i = 0, 1, \ldots, n.$$  

Proof. Use Taylor’s theorem and some inequality. See a standard textbook.

Example. Find the maximum error in approximating $y(1)$ by $y_2$ in the preceding example. Compare it with the actual absolute error using the solution $y = (t^2 - 2t + 2) - e^{-t}$.

Solution. $f(t, y) = t^2 - y$ $\implies$ $|f_y| = |-1| \leq 1 = L$ for all $y$. Thus $f$ satisfies a Lipschitz condition on $D = (0, 3) \times (-\infty, \infty)$ with the constant $L = 1$. Now

$$y = (t^2 - 2t + 2) - e^{-t} \quad \implies \quad y'' = 2 - e^{-t}.$$  

Since $y''' = e^{-t} > 0$, $y''$ is an increasing function and then

$$|y''| = |2 - e^{-t}| \leq 2 - e^{-3} = 1.95 = M \quad \text{for all } t \in [0, 3]$$

Note $h = 0.5$, $t_2 = 1$, and $a = 0$. Thus

$$|y(1) - y_2| \leq \frac{hM}{2L} \left[-1 + e^{L(t_2 - a)}\right] = \frac{0.5 \cdot 1.95}{2 \cdot 1} \left[-1 + e^{(1-0)}\right] = 0.83$$

Using the solution $y = (t^2 - 2t + 2) - e^{-t}$, we get the actual absolute error

$$|y(1) - y_2| = |(1 - e^{-1}) - 0.375| = 0.25.$$
5.2 Higher-order Taylor’s Method

Recall that the Euler’s method was derived by approximating \( y(t) \) by its Taylor polynomial of degree 1 about \( t = t_i \). Similarly we can approximate \( y(t) \) by its Taylor polynomial of degree \( k \) for any given integer \( k \geq 2 \). By Taylor’s theorem on \( y \) about \( t = t_i \),

\[
y(t) = y(t_i) + (t - t_i)y'(t_i) + \frac{(t-t_i)^2}{2!}y''(t_i) + \cdots + \frac{(t-t_i)^k}{k!}y^{(k)}(t_i) + \frac{(t-t_i)^{k+1}}{(k+1)!}y^{(k+1)}(\xi_i)
\]

Indeed

\[
y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2!}y''(t_i) + \cdots + \frac{h^k}{k!}y^{(k)}(t_i) + \frac{h^{k+1}}{(k+1)!}y^{(k+1)}(\xi_i)
\]

Since \( y' = f(t, y) \), \( y'' = f'(t, y) \), \ldots, \( y^{(k)} = f^{(k-1)}(t, y) \). Thus

\[
y(t_{i+1}) = y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2!}f'(t_i, y(t_i)) + \cdots + \frac{h^k}{k!}f^{(k-1)}(t_i, y(t_i)) + \frac{h^{k+1}}{(k+1)!}f^{(k)}(\xi_i, y(\xi_i))
\]

\[
\approx y_i + h\left[f(t_i, y_i) + \frac{h}{2!}f'(t_i, y_i) + \cdots + \frac{h^{k-1}}{k!}f^{(k-1)}(t_i, y_i)\right] + O(h^{k+1})
\]

\[
\implies y(t_{i+1}) \approx y_i + hT_k(t_i, y_i) =: y_{i+1},
\]

where \( T_k(t_i, y_i) = f(t_i, y_i) + \frac{h}{2!}f'(t_i, y_i) + \cdots + \frac{h^{k-1}}{k!}f^{(k-1)}(t_i, y_i) \).

Thus the Taylor’s Method of order \( k \) finds \( y_0, y_1, \ldots, y_n \) such that \( y_i \approx y(t_i), i = 0, 1, \ldots, n: \)

\[
\begin{align*}
y_0 &= c \\
y_{i+1} &= y_i + hT_k(t_i, y_i), i = 0, 1, \ldots, n-1.
\end{align*}
\]

**Example.** Use Taylor’s method of order 2 with step size \( h = 0.5 \) to approximate the solution of the following IVP:

\[
\frac{dy}{dt} = t^2 - y, \quad 0 \leq t \leq 2, \quad y(0) = 1
\]

**Solution.** We have \( h = 0.5, t_0 = 0, y_0 = 1 \) and \( y' = f(t, y) = t^2 - y \).

Then \( f'(t, y) = 2t - y' = 2t - (t^2 - y) = -t^2 + 2t + y \). So by the Taylor’s method of order 2,

\[
y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2!}f'(t_i, y_i)
\]

\[
y_{i+1} = y_i + 0.5(t_i^2 - y_i) + \frac{(0.5)^2}{2}(-t_i^2 + 2t_i + y_i) = \frac{3t_i^2 + 2t_i + 5y_i}{8}
\]

\[
t_1 = 0 + 1 \cdot 0.5 = 0.5, \quad y_1 = (3t_0^2 + 2t_0 + 5y_0)/8 = 0.625
\]

\[
t_2 = 0 + 2 \cdot 0.5 = 1, \quad y_2 = (3t_1^2 + 2t_1 + 5y_1)/8 = 0.6093 \text{ etc.}
\]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( t_i )</th>
<th>( y_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.625</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.6093</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.0058</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.8474</td>
</tr>
</tbody>
</table>
Using the solution \( y = (t^2 - 2t + 2) - e^{-t} \), we get the actual absolute error for \( y(1) \):

\[
|y(1) - y_2| = |(1 - e^{-1}) - 0.6093| = 0.02.
\]

So the Taylor’s method of order 2 is better than the Taylor’s method of order 1, i.e., the Euler’s method.

**Example.** Using Taylor’s method of order 3 with step size \( h = 0.5 \), set up an iteration formula for \( \{y_i\} \) to approximate the solution of the following IVP:

\[
\frac{dy}{dt} = t^2 - y, \quad 0 \leq t \leq 2, \quad y(0) = 1
\]

**Solution.** We have \( h = 0.5 \), \( t_0 = 0 \), \( y_0 = 1 \) and \( y' = f(t,y) = t^2 - y \). Then

\[
\begin{align*}
    f'(t,y) &= 2t - y' = 2t - (t^2 - y) = -t^2 + 2t + y \\
    f''(t,y) &= -2t + 2 + y' = -2t + 2 + (t^2 - y) = t^2 - 2t + 2 - y
\end{align*}
\]

So by the Taylor’s method of order 3, we have

\[
\begin{align*}
    y_{i+1} &= y_i + hf(t_i, y_i) + \frac{h^2}{2!} f'(t_i, y_i) + \frac{h^3}{3!} f''(t_i, y_i) \\
    y_{i+1} &= y_i + 0.5(t_i^2 - y_i) + \frac{(0.5)^2}{2}(-t_i^2 + 2t_i + y_i) + \frac{(0.5)^3}{6}(t_i^2 - 2t_i + 2 - y_i) \\
    y_{i+1} &= \frac{19t_i^2 + 10t_i + 2 + 29y_i}{48}, \quad y_0 = 1. \quad (t_i = 0 + 0.5i)
\end{align*}
\]
5.3 Runge-Kutta Method

The Euler’s method is the simplest way to solve the IVP (2) but it has error $O(h^2)$. The Taylor’s method of order $k$ has error $O(h^{k+1})$ which is small, but it involves calculating higher order derivative which is computationally expensive. So in 1901 Carl Runge and Martin Kutta developed a method with high accuracy that does not involve calculating derivatives.

A general form of the second order Runge-Kutta method is the following:

$$y_{i+1} = y_i + h\left[c_1 k_1 + c_2 k_2\right],$$

where

$$k_1 = f(t_i, y_i), \quad k_2 = f(t_i + \alpha h, y_i + \beta h k_1).$$

Different values of $c_1, c_2, \alpha, \beta$ give different second order Runge-Kutta methods.

Plugging $k_1, k_2$, we get

$$y_{i+1} = y_i + h\left[c_1 k_1 + c_2 k_2\right] = y_i + c_1 hf(t_i, y_i) + c_2 hf(t_i + \alpha h, y_i + \beta h k_1) \quad (3)$$

By Taylor’s theorem on $f(t, y)$ about $(t_i, y_i)$, we get

$$f(t, y) = f(t_i, y_i) + (t - t_i)f_i(t_i, y_i) + (y - y_i)f_y(t_i, y_i) + R_1(t, y).$$

Plugging $(t, y) = (t_i + \alpha h, y_i + \beta h k_1)$, we get

$$f(t_i + \alpha h, y_i + \beta h k_1) = f(t_i, y_i) + \alpha hf_i(t_i, y_i) + \beta h k_1 f_y(t_i, y_i) + O(h^2)$$

Plugging this $f(t_i + \alpha h, y_i + \beta h k_1)$ in (3), we get

$$y_{i+1} = y_i + c_1 hf(t_i, y_i) + c_2 h f(t_i, y_i) + \alpha h f_i(t_i, y_i) + \beta h k_1 f_y(t_i, y_i) + O(h^2)$$

$$= y_i + c_1 hf(t_i, y_i) + c_2 hf(t_i, y_i) + c_2 \alpha h f_i(t_i, y_i) + c_2 \beta h^2 k_1 f_y(t_i, y_i) + O(h^3)$$

$$= y_i + (c_1 + c_2) hf(t_i, y_i) + c_2 h^2 \left[\alpha f_i(t_i, y_i) + \beta k_1 f_y(t_i, y_i)\right] + O(h^3) \quad (4)$$

Now by Taylor’s theorem on $y$ about $t = t_i$,

$$y(t) = y(t_i) + (t - t_i)y'(t_i) + \frac{(t-t_i)^2}{2!}y''(t_i) + R_2$$

$$\implies y(t_{i+1}) = y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(t_i) + O(h^3)$$

$$\approx y_i + hf(t_i, y_i) + \frac{h^2}{2} f_y(t_i, y_i) + O(h^3)$$

$$= y_i + hf(t_i, y_i) + \frac{h^2}{2} \left[f_i(t_i, y_i) + f_y(t_i, y_i) f(t_i, y_i)\right] + O(h^3),$$

since $y' = f(t, y) \implies y'' = \frac{d}{dt} f(t, y) = f_t + f_y y' = f_t + f_y f$. Plugging $k_1 = f(t_i, y_i)$, we get

$$y(t_{i+1}) \approx y_i + hf(t_i, y_i) + \frac{h^2}{2} \left[f_i(t_i, y_i) + f_y(t_i, y_i) k_1\right] + O(h^3) \quad (5)$$
Since \( y(t_{i+1}) \approx y_{i+1} \), comparing (4) and (5), we get
\[ c_1 + c_2 = 1 \]
\[ c_2 \alpha = \frac{1}{2} \]
\[ c_2 \beta = \frac{1}{2} \]
There are many solutions for \( c_1, c_2, \alpha, \beta \). One solution is
\[ (c_1, c_2, \alpha, \beta) = \left( \frac{1}{2}, \frac{1}{2}, 1, 1 \right) \]
which gives a special second order Runge-Kutta method with error \( O(h^3) \) called the modified Euler’s method or “RK2”:
\[
y_{i+1} = y_i + \frac{h}{2} \left[ k_1 + k_2 \right],
\]
where
\[
k_1 = f(t_i, y_i),
k_2 = f(t_i + h, y_i + hk_1).
\]
Another solution \( (c_1, c_2, \alpha, \beta) = (0, 1, \frac{1}{2}, \frac{1}{2}) \) gives another second order Runge-Kutta method called the midpoint method:
\[
y_{i+1} = y_i + h k_2, \text{ where } k_1 = f(t_i, y_i), \quad k_2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2} k_1).
\]

**Example.** Use Runge-Kutta method of order 2 with step size \( h = 0.5 \) to approximate the solution of the following IVP:
\[
\frac{dy}{dt} = t^2 - y, \quad 0 \leq t \leq 2, \quad y(0) = 1
\]

**Solution.** We have \( h = 0.5, \quad t_0 = 0, \quad y_0 = 1 \) and \( y' = f(t, y) = t^2 - y \). So we have
\[
k_1 = f(t_i, y_i) = t_i^2 - y_i,
k_2 = f(t_i + h, y_i + h k_1)
\]
\[
= f(t_i + 0.5, y_i + 0.5(t_i^2 - y_i))
\]
\[
= (t_i + 0.5)^2 - (y_i + 0.5(t_i^2 - y_i))
\]
\[
= 0.25(2t_i^2 + 4t_i + 1 - 2y_i),
\]
\[
y_{i+1} = y_i + \frac{h}{2} \left[ k_1 + k_2 \right]
\]
\[
= y_i + \frac{0.5}{2} \left[ (t_i^2 - y_i) + 0.25(2t_i^2 + 4t_i + 1 - 2y_i) \right] = \frac{6t_i^2 + 4t_i + 1 + 10y_i}{16}.
\]
\[
t_1 = 0 + 1 \cdot 0.5 = 0.5, \quad y_1 = \frac{(6t_0^2 + 4t_0 + 1 + 10y_0)}{16} = 0.6875
\]
\[
t_2 = 0 + 2 \cdot 0.5 = 1, \quad y_2 = \frac{(6t_1^2 + 4t_1 + 1 + 10y_1)}{16} = 0.7109 \text{ etc.}
\]
Geometric interpretation:

At \((t_0, y_0)\) we draw a line with slope \(k_1 = y'(t_0) = f(t_0, y_0)\): \(y = y_0 + k_1(t - t_0)\). At \(t = t_1\), we get the point \((t_1, y_0 + k_1 h)\) on the line. Now at \((t_1, y_0 + k_1 h)\) we draw a line with slope \(k_2 = f(t_1, y_0 + k_1 h)\): \(y = (y_0 + k_1 h) + k_2(t - t_1)\). Now we go back to the point \((t_0, y_0)\) and draw a line with slope \((k_1 + k_2)/2\): \(y = y_0 + \frac{k_1 + k_2}{2}(t - t_0)\). At \(t = t_1\), the height of the last line is \(y_0 + \frac{k_1 + k_2}{2} h =: y_1\).

The most widely known Runge-Kutta method is “RK4”, the following fourth order Runge-Kutta method with error \(O(h^5)\):

\[
y_{i+1} = y_i + \frac{h}{6} \left[ k_1 + 2k_2 + 2k_3 + k_4 \right],
\]

where

\[
k_1 = f(t_i, y_i),
\]

\[
k_2 = f \left( t_i + \frac{h}{2}, y_i + k_1 \frac{h}{2} \right),
\]

\[
k_3 = f \left( t_i + \frac{h}{2}, y_i + k_2 \frac{h}{2} \right),
\]

\[
k_4 = f(t_i + h, y_i + k_3 h).
\]

So the basic idea is to draw a line at \((t_i, y_i)\) with slope \(m\): \(y = y_i + m(t - t_i)\) where \(m\) is the weighted average of slopes \(k_1, k_2, k_3, k_4\). Then height of the line at \(t = t_{i+1}\) is \(y_i + mh =: y_{i+1}\).
Example. Use Runge-Kutta method of order 4 with step size $h = 0.5$ to approximate the solution of the following IVP:

$$\frac{dy}{dt} = t^2 - y, \quad 0 \leq t \leq 2, \quad y(0) = 1$$

Solution. We have $h = 0.5$, $t_0 = 0$, $y_0 = 1$ and $y' = f(t, y) = t^2 - y$. So we have

$$k_1 = f(t_i, y_i) = t_i^2 - y_i,$$

$$k_2 = f\left(t_i + \frac{h}{2}, y_i + \frac{k_1 h}{2}\right) = f(t_i + 0.25, y_i + 0.25k_1) = (t_i + 0.25)^2 - (y_i + 0.25k_1),$$

$$k_3 = f\left(t_i + \frac{h}{2}, y_i + k_2 h\right) = f(t_i + 0.25, y_i + 0.25k_2) = (t_i + 0.25)^2 - (y_i + 0.25k_2),$$

$$k_4 = f(t_i + h, y_i + k_3 h) = f(t_i + 0.5, y_i + 0.5k_3) = (t_i + 0.5)^2 - (y_i + 0.5k_3)$$

$$y_{i+1} = y_i + \frac{h}{6} \left[k_1 + 2k_2 + 2k_3 + k_4\right] = y_i + \frac{0.5}{6} \left[k_1 + 2k_2 + 2k_3 + k_4\right].$$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$t_i$</th>
<th>$y_i$</th>
<th>$k_1$</th>
<th>$k_2$</th>
<th>$k_3$</th>
<th>$k_4$</th>
<th>$y_{i+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-1</td>
<td>-0.6875</td>
<td>-0.7656</td>
<td>-0.3671</td>
<td>0.6438</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.6438</td>
<td>-0.3938</td>
<td>0.0170</td>
<td>-0.0856</td>
<td>0.3989</td>
<td>0.6328</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0.6328</td>
<td>0.3671</td>
<td>0.8378</td>
<td>0.7201</td>
<td>0.1257</td>
<td>1.0278</td>
</tr>
<tr>
<td>3</td>
<td>1.5</td>
<td>1.0278</td>
<td>1.2221</td>
<td>1.7290</td>
<td>1.6023</td>
<td>2.1709</td>
<td>1.8658</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1.8658</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
6 Linear Algebra

In this chapter we will learn introductory linear algebra and numerical methods for solving system of linear equations and finding eigenvalues of a matrix.

6.1 Introduction to Linear Algebra

Matrix: An $m \times n$ matrix $A$ is an $m$-by-$n$ array of scalars of the form

$$A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn}
\end{bmatrix}.$$  

The order (or size) of $A$ is $m \times n$ (read as $m$ by $n$). It means $A$ has $m$ rows and $n$ columns. The $(i,j)$-entry of $A = [a_{i,j}]$ is $a_{i,j}$.  

Example. $A = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 0 & -1 \end{bmatrix}$ is a $2 \times 3$ real matrix. The $(2,3)$-entry of $A$ is $-1$. 

The identity matrix of order $n$, denoted by $I_n$, is the $n \times n$ diagonal matrix whose diagonal entries are 1. Example. $I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is the $3 \times 3$ identity matrix.

An $n \times 1$ matrix is called a column matrix or $n$-dimensional (column) vector. It is denoted by $\vec{x}$. Example. $\vec{x} = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}$ is a 3-dimensional vector which represents the position vector of the point $(1,0,-2)$ in $\mathbb{R}^3$. 

Now we will learn some operations on matrices.

- Norm: We will use only two norms of a vector $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$:

  1. $l_2$ norm (Euclidean norm): $\|\vec{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}$

  2. $l_\infty$ norm: $\|\vec{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$

For the above $\vec{x}$, $\|\vec{x}\|_2 = \sqrt{5}$ and $\|\vec{x}\|_\infty = 2$. Note that $\|\vec{x}\|_\infty \leq \|\vec{x}\|_2$ for all $\vec{x}$. 

45
- **Transpose:** The transpose of an $m \times n$ matrix $A$, denoted by $A^T$, is an $n \times m$ matrix whose columns are corresponding rows of $A$, i.e., $(A^T)_{ij} = A_{ji}$.

Example. If $A = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 0 & -1 \end{bmatrix}$, then $A^T = \begin{bmatrix} 1 & -3 \\ 2 & 0 \\ 0 & -1 \end{bmatrix}$.

- **Scalar Multiplication:** Let $A$ be a matrix and $c$ be a scalar. The scalar multiple, denoted by $cA$, is the matrix whose entries are $c$ times the corresponding entries of $A$.

Example. If $A = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 0 & -1 \end{bmatrix}$, then $-2A = \begin{bmatrix} -2 & -4 & 0 \\ 6 & 0 & 2 \end{bmatrix}$.

- **Sum:** If $A$ and $B$ are $m \times n$ matrices, then the sum $A + B$ is the $m \times n$ matrix whose entries are the sum of the corresponding entries of $A$ and $B$, i.e., $(A + B)_{ij} = A_{ij} + B_{ij}$.

Example. If $A = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 0 & -1 \end{bmatrix}$ and $B = \begin{bmatrix} 0 & -2 & 0 \\ 3 & 0 & 2 \end{bmatrix}$, then $A + B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Exercise. Find $2A - B$.

- **Multiplication:**

1. Matrix-vector multiplication: If $A$ is an $m \times n$ matrix and $\vec{x}$ is an $n$-dimensional vector, then their product $A\vec{x}$ is an $m$-dimensional vector whose $(i,1)$-entry is $a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n$, the dot product of the row $i$ of $A$ and $\vec{x}$. Note that

$$A\vec{x} = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \end{bmatrix} = x_1 \begin{bmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{bmatrix} + x_2 \begin{bmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{bmatrix} + \cdots + x_n \begin{bmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{bmatrix}.$$

Example. If $A = \begin{bmatrix} 1 & 2 & 0 \\ -3 & 0 & -1 \end{bmatrix}$ and $\vec{x} = \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}$, then $A\vec{x} = \begin{bmatrix} -1 \\ -3 \end{bmatrix}$ which is a linear combination of first and second columns of $A$ with weights 1 and $-1$ respectively.

2. Matrix-matrix multiplication: If $A$ is an $m \times n$ matrix and $B$ is an $n \times p$ matrix, then their product $AB$ is an $m \times p$ matrix whose $(i,j)$-entry is the dot product the row $i$ of $A$ and the column $j$ of $B$.

$$(AB)_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{in}b_{nj}$$

Note that column $i$ of $AB$ is $A$(column $i$ of $B$). Also note $AB \neq BA$ in general.

Example. If $A = \begin{bmatrix} 1 & 2 & 2 \\ 0 & 2 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & -2 \\ 0 & 0 \\ 1 & 1 \end{bmatrix}$, then $AB = \begin{bmatrix} 4 & 0 \\ 2 & 2 \end{bmatrix}$. 
**Determinant:** The determinant of an $n \times n$ matrix $A$ is denoted by $\det A$ and $|A|$. It is defined recursively. By hand we will only find determinant of order 2 and 3.

\[
\begin{vmatrix}
  a_{11} & a_{12} \\
  a_{21} & a_{22}
\end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.
\]

\[
\begin{vmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{vmatrix} = a_{11} \begin{vmatrix}
  a_{22} & a_{23} \\
  a_{32} & a_{33}
\end{vmatrix} - a_{12} \begin{vmatrix}
  a_{21} & a_{23} \\
  a_{31} & a_{33}
\end{vmatrix} + a_{13} \begin{vmatrix}
  a_{21} & a_{22} \\
  a_{31} & a_{32}
\end{vmatrix}.
\]

Example. \[
\begin{vmatrix}
  2 & 1 & 7 \\
  -3 & 0 & -8 \\
  0 & 1 & -3
\end{vmatrix} = 2 \begin{vmatrix}
  0 & -8 \\
  1 & -3
\end{vmatrix} - 1 \begin{vmatrix}
  -3 & -8 \\
  0 & -3
\end{vmatrix} + 7 \begin{vmatrix}
  -3 & 0 \\
  0 & 1
\end{vmatrix} = -14.
\]

**Inverse of a matrix:** An $n \times n$ matrix $A$ is called invertible if there is an $n \times n$ matrix $B$ such that $AB = BA = I_n$. Here $B$ is called the inverse of $A$ which is denoted by $A^{-1}$. So

\[AA^{-1} = A^{-1}A = I_n.\]

Example. \[
\begin{bmatrix}
  a & b \\
  c & d
\end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix}
  d & -b \\
  -c & a
\end{bmatrix}.
\]

Theorem. An $n \times n$ matrix $A$ is invertible iff $\det A \neq 0$. 

47
6.2 Systems of Linear Equations: Gaussian Elimination

A system of linear equations with \( n \) variables \( x_1, \ldots, x_n \) and \( n \) equations can be written as follows:

\[
\begin{align*}
    a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\
    a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\
    \vdots & \quad \vdots & \quad \vdots \\
    a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n.
\end{align*}
\]

The above linear system is equivalent to the matrix equation \( A \overrightarrow{x} = \overrightarrow{b} \), where

\[
A = \begin{bmatrix}
    a_{11} & a_{12} & \cdots & a_{1n} \\
    a_{21} & a_{22} & \cdots & a_{2n} \\
    \vdots & \vdots & \ddots & \vdots \\
    a_{n1} & a_{n2} & \cdots & a_{nn}
\end{bmatrix}, \quad \overrightarrow{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \text{and} \quad \overrightarrow{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.
\]

\( A \) is called the coefficient matrix and \( [A \overrightarrow{b}] \) is called the augmented matrix.

Example.

\[
\begin{align*}
2x_1 + x_2 &= 7 \\
-3x_1 + x_2 + 2x_3 &= 0 \\
x_2 + 2x_3 &= -3
\end{align*}
\]

The above linear system is equivalent to \( A \overrightarrow{x} = \overrightarrow{b} \), where \( A = \begin{bmatrix} 2 & 1 & 0 \\ -3 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix} \),

\[
\overrightarrow{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \overrightarrow{b} = \begin{bmatrix} 7 \\ -8 \\ -3 \end{bmatrix}, \quad \text{and the augmented matrix is} \ [A \overrightarrow{b}] = \begin{bmatrix} 2 & 1 & 0 & 7 \\ -3 & 0 & 1 & -8 \\ 0 & 1 & 2 & -3 \end{bmatrix}.
\]

Throughout this section assume

(1) \( A \overrightarrow{x} = \overrightarrow{b} \) has a unique solution and (2) \( a_{ii} \neq 0 \) for \( i = 1, \ldots, n \).

Techniques for solving \( A \overrightarrow{x} = \overrightarrow{b} \):

There are multiple ways to solve \( A \overrightarrow{x} = \overrightarrow{b} \). First we learn Gaussian elimination as illustrated in the following example:

Example. By Gaussian elimination find the unique solution of the following system of equations.

\[
\begin{align*}
2x_1 - 4x_2 + 2x_3 &= 2 \\
x_1 + x_2 + 2x_3 &= 0 \\
-3x_1 + 8x_2 - 3x_3 &= -3
\end{align*}
\]
Solution.

\[ E_1 : 2x_1 - 4x_2 + 2x_3 = 2 \]
\[ E_2 : x_1 + x_2 + 2x_3 = 0 \]
\[ E_3 : -3x_1 + 8x_2 - 3x_3 = -3 \]

\[ -\frac{1}{2} E_1 + E_2 \rightarrow E_1 : 2x_1 - 4x_2 + 2x_3 = 2 \]
\[ \frac{3}{2} E_1 + E_3 \rightarrow E_2 : 3x_2 + x_3 = -1 \]
\[ E_3 : 2x_2 = 0 \]

\[ -\frac{3}{2} E_2 + E_3 \rightarrow E_1 : 2x_1 - 4x_2 + 2x_3 = 2 \]
\[ E_2 : 3x_2 + x_3 = -1 \]
\[ E_3 : -\frac{2}{3}x_3 = \frac{2}{3} \]

Note that if \( a_{ii} = 0 \) for some \( i \) and \( a_{ji} \neq 0 \) for some \( j > i \), then we interchange \( E_i \) and \( E_j \). This is called partial pivoting. If \( a_{ji} = 0 \) for all \( j \geq i \), then there is no unique solution.

Now we do backward substitutions:

\[ E_3 \implies x_3 = \frac{2/3}{-2/3} = -1 \]
\[ E_2 \implies x_2 = \frac{1}{3}(-1 - x_3) = \frac{1}{3}(-1 + 1) = 0 \]
\[ E_1 \implies x_1 = \frac{1}{2}(2 + 4x_2 - 2x_3) = \frac{1}{2}(2 + 4 \cdot 0 - 2(-1)) = 2 \]

Thus the unique solution is \((x_1, x_2, x_3) = (2, 0, -1)\).

Gaussian elimination steps:

1. For each \( i = 1, \ldots, n-1 \), \( E_j = E_j - \frac{a_{ji}}{a_{ii}} E_i \), \( j = i+1, \ldots, n \).

2. \( x_n = \frac{b_n}{a_{nn}} \) and \( x_i = \frac{1}{a_{ii}} \left( b_i - \sum_{j=i+1}^{n} a_{ij} x_j \right) \), \( i = n-1, n-2, \ldots, 1 \). (backward substitution)

Operation Counts:

In step 1, for each \( i = 1, \ldots, n-1 \), we do \((n-i)\) divisions \( \frac{a_{ji}}{a_{ii}} \), \( j = i+1, \ldots, n \). Then for each \( j = i+1, \ldots, n \), to get \( \frac{b_i}{a_{ii}} E_i \), we do \( n-i+1 \) multiplication when we multiply \( n-i+1 \) numbers \( a_{i,i+1}, \ldots, a_{i,n}, b_i \) in \( E_i \) by \( \frac{a_{ii}}{a_{ii}} \). So for each \( i = 1, \ldots, n-1 \), the total number of multiplications/divisions is

\[(n-i) + (n-i)(n-i+1) = (n-i)(n-i+2)\].
For a fixed $i = 1, \ldots, n - 1$, we do $(n - i + 1)$ subtraction for $E_j = E_j - \frac{a_{ji}}{a_{ii}} E_i$ for each $j = i + 1, \ldots, n$, i.e., total $(n - i)(n - i + 1)$ subtractions.

So the total number of multiplications/divisions is

$$
\sum_{i=1}^{n-1} (n - i)(n - i + 2) = \frac{2n^3 + 3n^2 - 5n}{6}.
$$

The total number of subtractions is

$$
\sum_{i=1}^{n-1} (n - i)(n - i + 1) = \frac{n^3 - n}{3}.
$$

Note that in a computer the time for a multiplication/division is greater than that of an addition/subtraction. So the total number of operations in step 1 is $\frac{2n^3 + 3n^2 - 5n}{6} + \frac{n^3 - n}{3} = \frac{4n^3 + 3n^2 - 7n}{6}$ which is $O(n^3)$. Similarly we can show the total number of operations in step 2 is $\frac{n^2 + n}{2} + \frac{n^2 - n}{2} = n^2$. Thus total number of operations in Gaussian elimination is $\frac{4n^3 + 3n^2 - 7n}{6} + n^2 = \frac{4n^3 + 9n^2 - 7n}{6}$ which is $O(n^3)$.

**LU-factorization:**

If $M = [A \rightarrow \overrightarrow{b}]$, the first step transforms it into $M' = [U \rightarrow \overrightarrow{b}]$ where $U$ is an upper triangular matrix with nonzero diagonals. This is obtained by premultiplying $M$ by elementary matrices. For example, premultiplying $M$ by the elementary matrix $E_1 = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ gives

$$
E_1 M = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ 1 & 1 & 2 \\ -3 & 8 & -3 \end{bmatrix} = \begin{bmatrix} 2 & -4 & 2 \\ 0 & 3 & 1 \\ -3 & 8 & -3 \end{bmatrix}.
$$

Similarly using the elementary matrices $E_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{3}{2} & 0 & 1 \end{bmatrix}$ and $E_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{2}{3} & 1 \end{bmatrix}$, we get

$$
E_3 E_2 E_1 M = E_3 E_2 E_1 [A \rightarrow \overrightarrow{b}] = \begin{bmatrix} 2 & -4 & 2 \\ 0 & 3 & 1 \\ 0 & -2/3 & 2/3 \end{bmatrix} = [U \rightarrow \overrightarrow{b}],
$$

which implies $E_3 E_2 E_1 A = U$ an upper-triangular matrix with nonzero diagonals. Now

$$
E_3 E_2 E_1 A = U \implies A = (E_3 E_2 E_1)^{-1} U = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ -3/2 & 2/3 & 1 \end{bmatrix} \begin{bmatrix} 2 & -4 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & -2/3 \end{bmatrix},
$$

50
where \( L = [l_{ij}] \) is a lower-triangular matrix with \( l_{ji} = a_{ji}^{(i)} / a_{ii}^{(i)} \). So we have the LU-factorization of \( A: A = LU \). Then \( A \vec{x} = \vec{b} \implies LU \vec{x} = \vec{b} \). Now we do steps to find \( \vec{x} \):

1. From \( L \vec{y} = \vec{b} \) solve for \( \vec{y} \) by forward substitution
2. From \( U \vec{x} = \vec{y} \) solve for \( \vec{x} \) by backward substitution

**Example.** Solve \( A \vec{x} = [2 \ 0 \ -3]^T \), where

\[
A = \begin{bmatrix}
1 & 0 & 0 \\
1/2 & 1 & 0 \\
-3/2 & 2/3 & 1
\end{bmatrix}
\begin{bmatrix}
2 & -4 & 2 \\
0 & 3 & 1 \\
0 & 0 & -2/3
\end{bmatrix}.
\]

**Solution.** With \( A = LU \), we solve for \( \vec{y} \) from \( L \vec{y} = \vec{b} \) by forward substitution:

\[
\begin{align*}
L \vec{y} = \vec{b} \implies \begin{bmatrix}
1 & 0 & 0 \\
1/2 & 1 & 0 \\
-3/2 & 2/3 & 1
\end{bmatrix} \begin{bmatrix}
y_1 \\
y_2 \\
y_3
\end{bmatrix} &= \begin{bmatrix}
2 \\
0 \\
-3
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
y_1 = 2 \\
y_1/2 + y_2 = 0 \\
-3y_1/2 + 2y_2/3 + y_3 = -3
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
y_1 = 2 \\
y_2 = 0 - y_1/2 = 0 - 2/2 = -1 \\
y_3 = -3 + 3y_1/2 - 2y_2/3 = -3 + 3 \cdot 2/2 - 2(-1)/3 = 2/3
\end{cases}
\end{align*}
\]

Now we solve for \( \vec{x} \) from \( U \vec{x} = \vec{y} \) by forward substitution:

\[
\begin{align*}
U \vec{x} = \vec{y} \implies \begin{bmatrix}
2 & -4 & 2 \\
0 & 3 & 1 \\
0 & 0 & -2/3
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix} &= \begin{bmatrix}
2 \\
-1 \\
2/3
\end{bmatrix}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
2x_1 - 4x_2 + 2x_3 = 2 \\
3x_2 + x_3 = -1 \\
-2x_3/3 = 2/3
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\begin{cases}
x_3 = 2/3 = -1 \\
x_2 = \frac{1}{3}(-1 - x_3) = \frac{1}{3}(-1 + 1) = 0 \\
x_1 = \frac{1}{2}(2 + 4x_2 - 2x_3) = \frac{1}{2}(2 + 4 \cdot 0 - 2(-1)) = 2
\end{cases}
\end{align*}
\]

Thus the unique solution is \( (x_1, x_2, x_3) = (2, 0, -1) \).

**Advantage of LU-factorization:** If we need to solve \( A \vec{x} = \vec{b}_1, A \vec{x} = \vec{b}_2, \ldots, A \vec{x} = \vec{b}_k \), then Gaussian elimination solves each problem separately. But once we know the LU-factorization of \( A = LU \), the solution \( \vec{x}_i \) of \( A \vec{x} = \vec{b}_i \) is obtained just from forward and backward substitution in \( L \vec{y}_i = \vec{b}_i \) and \( U \vec{x}_i = \vec{y}_i \) respectively.
6.3 Jacobi and Gauss-Seidel Methods

Consider a system of $n$ equations in $n$ variables $x_1, \ldots, x_n$:

$$A \mathbf{x} = \mathbf{b}.$$

Throughout this section assume (1) there is a unique solution and (2) $a_{ii} \neq 0$ for $i = 1, \ldots, n$.

The Jacobi method constructs a sequence $\{\mathbf{x}^{(k)}\}$ to approximate the unique solution of $A \mathbf{x} = \mathbf{b}$. First decompose $A$ into

$$A = D + R,$$

where $D = \text{diag}(a_{11}, \ldots, a_{nn})$ is the diagonal part of $A$ and $R = A - D$ is the remainder part of $A$. Since $a_{ii} \neq 0$ for $i = 1, \ldots, n$, $\det A = a_{11} \cdots a_{nn} \neq 0$ and hence $D^{-1}$ exists.

$$A \mathbf{x} = \mathbf{b} \implies (D + R) \mathbf{x} = \mathbf{b} \implies D \mathbf{x} = \mathbf{b} - R \mathbf{x} \implies \mathbf{x} = D^{-1}(\mathbf{b} - R \mathbf{x})$$

The iteration formula for the Jacobi method is

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - R \mathbf{x}^{(k)}), \quad k = 0, 1, 2, \ldots$$

If we write $\mathbf{x}^{(k)} = [x_1^{(k)}, \ldots, x_n^{(k)}]^T$, then the above iteration formula gives

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, \ldots, n.$$

**Example.** Do three iterations by the Jacobi method with $\mathbf{x}^{(0)} = [0, 0, 0]^T$ to approximate the unique solution of the following system of equations.

\[
\begin{align*}
7x_1 + x_2 - 3x_3 &= 6 \\
x_1 - 11x_2 + 6x_3 &= 3 \\
2x_1 + x_2 + 9x_3 &= 5
\end{align*}
\]

**Solution.** Note that $a_{11}, a_{22},$ and $a_{33}$ are all nonzero.

\[
\begin{align*}
x_1^{(k+1)} &= \frac{1}{7} \left( 6 - x_2^{(k)} + 3x_3^{(k)} \right) \\
x_2^{(k+1)} &= \frac{1}{-11} \left( 3 - x_1^{(k)} - 6x_3^{(k)} \right) \\
x_3^{(k+1)} &= \frac{1}{9} \left( 5 - 2x_1^{(k)} - x_2^{(k)} \right)
\end{align*}
\]
The actual solution is \((x_1, x_2, x_3) = (1, 0, \frac{1}{3})\).

Note in the Jacobi iteration formula that in time of computing \(x_i^{(k+1)}\), \(i \geq 1\), we have calculated \(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}\). So we can replace \(x_1^{(k)}, \ldots, x_{i-1}^{(k)}\) by \(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}\) respectively. That is how we get the iteration formula of the Gauss-Seidel method:

\[
x_i^{(k+1)} = \frac{1}{a_{ii}} \left[ b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right], \quad i = 1, \ldots, n.
\]

**Example.** Do three iterations by the Gauss-Seidel method with \(\vec{x}^{(0)} = [0, 0, 0]^T\) to approximate the unique solution of the following system of equations.

\[
\begin{align*}
7x_1 + x_2 - 3x_3 &= 6 \\
x_1 - 11x_2 + 6x_3 &= 3 \\
2x_1 + x_2 + 9x_3 &= 5
\end{align*}
\]

**Solution.** Note that \(a_{11}, a_{22}, \) and \(a_{33}\) are all nonzero.

\[
\begin{align*}
x_1^{(k+1)} &= \frac{1}{7} \left( 6 - x_2^{(k)} + 3x_3^{(k)} \right) \\
x_2^{(k+1)} &= \frac{1}{11} \left( 3 - x_1^{(k+1)} - 6x_3^{(k)} \right) \\
x_3^{(k+1)} &= \frac{1}{9} \left( 5 - 2x_1^{(k+1)} - x_2^{(k+1)} \right)
\end{align*}
\]

<table>
<thead>
<tr>
<th>(k)</th>
<th>(x_1^{(k)})</th>
<th>(x_2^{(k)})</th>
<th>(x_3^{(k)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.85</td>
<td>-0.19</td>
<td>0.38</td>
</tr>
<tr>
<td>2</td>
<td>1.04</td>
<td>0.02</td>
<td>0.32</td>
</tr>
<tr>
<td>3</td>
<td>0.99</td>
<td>0.00</td>
<td>0.33</td>
</tr>
</tbody>
</table>

The actual solution is \((x_1, x_2, x_3) = (1, 0, \frac{1}{3})\).
Convergence: The sequence \( \{x^{(k)}\} \) constructed by the Jacobi or Gauss-Seidel method does not always converge to the unique solution of \( Ax = b \). However the convergence is guaranteed for any \( x^{(0)} \) if \( A \) is strictly diagonally dominant (i.e., \( |a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \) for all \( i \)).

Example. For the following \( A \), determine if the Jacobi and Gauss-Seidel iterations converge

(a) \( A = \begin{bmatrix} 2 & 1 & 0 \\ -3 & 4 & 1 \\ 0 & 1 & 3 \end{bmatrix} \), (b) \( A = \begin{bmatrix} 2 & 1 & 0 \\ -3 & -5 & 1 \\ 0 & 1 & 3 \end{bmatrix} \).

Solution. (a) \( |2| > |1| + |0| \), but \( |4| \neq |-3| + |1| \). So \( A \) is not strictly diagonally dominant and convergence is not guaranteed.
(b) \( |2| > |1| + |0| \), \( |-5| > |-3| + |1| \), \( |3| > |0| + |1| \). So \( A \) is strictly diagonally dominant and convergence is guaranteed.

The spectral radius \( \rho(A) \) of a matrix \( A \) is the largest modulus of an eigenvalue of \( A \). There are necessary and sufficient spectral conditions for convergence:

- Jacobi: \( \rho(D^{-1}R) < 1 \)
- Gauss-Seidel: \( \rho(L^{-1}_sU) < 1 \) where \( U \) is the upper triangular part of \( A \) and \( L_s = A - U \)

Remark. A strictly diagonally dominant matrix satisfies the above two conditions. But the converse is not true. For example, \( A = \begin{bmatrix} 4 & 1 \\ 3 & 2 \end{bmatrix} \) is not strictly diagonally dominant but \( \rho(D^{-1}R) = \sqrt{3/8} < 1 \).

Theorem. For any \( x^{(0)} \in \mathbb{R}^d \), the sequence \( \{x^{(n)}\} \) defined by

\[ x^{(n)} = Ax^{(n-1)} + c, \quad n \geq 1, \]

converges to the unique solution \( x = Ax + c \) if and only if \( \rho(A) < 1 \).
6.4 Eigenvalues: Power Method

Let \( A \) be an \( n \times n \) matrix. If \( A \overrightarrow{x} = \lambda \overrightarrow{x} \) for some nonzero vector \( \overrightarrow{x} \) and some scalar \( \lambda \), then \( \lambda \) is an eigenvalue of \( A \) and \( \overrightarrow{x} \) is an eigenvector of \( A \) corresponding to \( \lambda \).

**Example.** Consider \( A = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \), \( \lambda = 3 \), \( \overrightarrow{v} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \), \( \overrightarrow{u} = \begin{bmatrix} -2 \\ 1 \end{bmatrix} \).

Since \( A \overrightarrow{v} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix} = 3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \lambda \overrightarrow{v} \), 3 is an eigenvalue of \( A \) and \( \overrightarrow{v} \) is an eigenvector of \( A \) corresponding to the eigenvalue 3.

Since \( A \overrightarrow{u} = \begin{bmatrix} 1 & 2 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix} \neq \lambda \begin{bmatrix} -2 \\ 1 \end{bmatrix} = \lambda \overrightarrow{u} \) for all scalars \( \lambda \), \( \overrightarrow{u} \) is not an eigenvector of \( A \).

The characteristic polynomial of \( A \) is \( \det(A - \lambda I) \) and its roots are the eigenvalues of \( A \). The eigenvectors of \( A \) corresponding to the eigenvalue \( \lambda \) are nonzero solutions of \( (A - \lambda I) \overrightarrow{x} = \overrightarrow{0} \). The eigenspace of \( A \) corresponding to \( \lambda \) is

\[
Nul(A - \lambda I) = \{ \overrightarrow{x} \mid (A - \lambda I) \overrightarrow{x} = \overrightarrow{0} \}.
\]

The dominant eigenvalue of \( A \) is the eigenvalue of \( A \) with the largest modulus. The spectral radius \( \rho(A) \) of \( A \) is the modulus of the dominant eigenvalue of \( A \). For example, if 3 and \(-4\) are the eigenvalues of \( A \), then \(-4\) is the dominant eigenvalue of \( A \) and \( \rho(A) = 4 \).

In this section consider an \( n \times n \) matrix \( A \) with real eigenvalues \( \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \). Assume

1. \( \lambda_1 \) is the dominant eigenvalue of multiplicity 1 with eigenvector \( \overrightarrow{v}_1 \)
2. There are \( n \) linearly independent eigenvectors \( \overrightarrow{v}_1, \ldots, \overrightarrow{v}_n \) of \( A \) (i.e., \( A \) is diagonalizable)

The Power Method constructs a sequence \( \{ \overrightarrow{x}^{(k)} \} \) defined by

\[
\overrightarrow{x}^{(k)} = \frac{A^k \overrightarrow{x}^{(0)}}{\|A^k \overrightarrow{x}^{(0)}\|}, \quad k \geq 1
\]

with an initial approximation \( \overrightarrow{x}^{(0)} \) to approximate an unit eigenvector corresponding to \( \lambda_1 \): Since \( \overrightarrow{v}_1, \ldots, \overrightarrow{v}_n \) are \( n \) linearly independent eigenvectors, they span \( \mathbb{R}^n \) and

\[
\overrightarrow{x}^{(0)} = c_1 \overrightarrow{v}_1 + c_2 \overrightarrow{v}_2 + \cdots + c_n \overrightarrow{v}_n,
\]

for some scalars \( c_1, c_2, \ldots, c_n \). We choose \( \overrightarrow{x}^{(0)} \) such that \( c_1 \neq 0 \). Then
\[ \overrightarrow{x}^{(0)} = c_1 \overrightarrow{v}_1 + c_2 \overrightarrow{v}_2 + \cdots + c_n \overrightarrow{v}_n \]
\[ \implies A \overrightarrow{x}^{(0)} = c_1 A \overrightarrow{v}_1 + c_2 A \overrightarrow{v}_2 + \cdots + c_n A \overrightarrow{v}_n \]
\[ \implies A \overrightarrow{x}^{(0)} = c_1 \lambda_1 \overrightarrow{v}_1 + c_2 \lambda_2 \overrightarrow{v}_2 + \cdots + c_n \lambda_n \overrightarrow{v}_n \]
\[ \implies A^2 \overrightarrow{x}^{(0)} = c_1 \lambda_1 A \overrightarrow{v}_1 + c_2 \lambda_2 A \overrightarrow{v}_2 + \cdots + c_n \lambda_n A \overrightarrow{v}_n \]
\[ \implies A^2 \overrightarrow{x}^{(0)} = c_1 \lambda_1^2 \overrightarrow{v}_1 + c_2 \lambda_2^2 \overrightarrow{v}_2 + \cdots + c_n \lambda_n^2 \overrightarrow{v}_n \]
\[ \vdots \]
\[ \implies A^k \overrightarrow{x}^{(0)} = c_1 \lambda_1^k \overrightarrow{v}_1 + c_2 \lambda_2^k \overrightarrow{v}_2 + \cdots + c_n \lambda_n^k \overrightarrow{v}_n \]
\[ \implies A^k \overrightarrow{x}^{(0)} = \lambda_1^k \left( c_1 \overrightarrow{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \overrightarrow{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \overrightarrow{v}_n \right) \]
\[ \overrightarrow{x}^{(k)} = \frac{A^k \overrightarrow{x}^{(0)}}{||A^k \overrightarrow{x}^{(0)}||} = \frac{\lambda_1^k \left( c_1 \overrightarrow{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \overrightarrow{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \overrightarrow{v}_n \right)}{\left( c_1 \overrightarrow{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \overrightarrow{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \overrightarrow{v}_n \right)} \]
\[ = \lambda_1^k \left( c_1 \overrightarrow{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \overrightarrow{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \overrightarrow{v}_n \right) \]
\[ = \pm \frac{c_1 \overrightarrow{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \overrightarrow{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \overrightarrow{v}_n}{||c_1 \overrightarrow{v}_1||} = \pm \frac{\overrightarrow{v}_1}{||\overrightarrow{v}_1||} \text{ as } k \to \infty. \]

Note that
\[ A \overrightarrow{v}_1 = \lambda_1 \overrightarrow{v}_1 \implies \overrightarrow{v}_1^T A \overrightarrow{v}_1 = \lambda_1 \overrightarrow{v}_1^T \overrightarrow{v}_1 \implies \lambda_1 = \frac{\overrightarrow{v}_1^T A \overrightarrow{v}_1}{\overrightarrow{v}_1^T \overrightarrow{v}_1} \]
If \( \overrightarrow{x}^{(k)} \approx \overrightarrow{v}_1 \), then
\[ \lambda_1 \approx \lambda_1^{(k)} := \frac{(\overrightarrow{x}^{(k)})^T A \overrightarrow{x}^{(k)}}{(\overrightarrow{x}^{(k)})^T \overrightarrow{x}^{(k)}}. \]

Remark.

1. The power method does not work for \( \overrightarrow{x}^{(0)} = c_1 \overrightarrow{v}_1 + c_2 \overrightarrow{v}_2 + \cdots + c_n \overrightarrow{v}_n \) when \( c_1 = 0 \).

2. The rate convergence of \( \{ \overrightarrow{x}^{(k)} \} \) is faster when \( |\frac{\lambda_2}{\lambda_1}| \) is smaller, i.e., \( |\lambda_2| << |\lambda_1| \).
Example. Do three iterations by the power method with \( \overrightarrow{x}^{(0)} = [2, 3]^T \) to approximate the dominant eigenvalue of \( A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \) with its eigenvector.

Solution.

\[
A \overrightarrow{x}^{(0)} = \begin{bmatrix} 8 \\ 7 \end{bmatrix} \implies \overrightarrow{x}^{(1)} = \frac{A \overrightarrow{x}^{(0)}}{||A \overrightarrow{x}^{(0)}||_\infty} = \frac{1}{8} \begin{bmatrix} 8 \\ 7 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.87 \end{bmatrix}
\]

\[
A \overrightarrow{x}^{(1)} = \begin{bmatrix} 2.74 \\ 2.87 \end{bmatrix} \implies \overrightarrow{x}^{(2)} = \frac{A \overrightarrow{x}^{(1)}}{||A \overrightarrow{x}^{(1)}||_\infty} = \frac{1}{2.87} \begin{bmatrix} 2.74 \\ 2.87 \end{bmatrix} = \begin{bmatrix} 0.95 \\ 1 \end{bmatrix}
\]

\[
A \overrightarrow{x}^{(2)} = \begin{bmatrix} 2.95 \\ 2.90 \end{bmatrix} \implies \overrightarrow{x}^{(3)} = \frac{A \overrightarrow{x}^{(2)}}{||A \overrightarrow{x}^{(2)}||_\infty} = \frac{1}{2.95} \begin{bmatrix} 2.95 \\ 2.9 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.98 \end{bmatrix}
\]

\[
\lambda_1^{(1)} = \frac{(\overrightarrow{x}^{(1)})^T A \overrightarrow{x}^{(1)}}{(\overrightarrow{x}^{(1)})^T \overrightarrow{x}^{(1)}} \approx 5.23/1.75 = 2.98
\]

\[
\lambda_1^{(2)} = \frac{(\overrightarrow{x}^{(2)})^T A \overrightarrow{x}^{(2)}}{(\overrightarrow{x}^{(2)})^T \overrightarrow{x}^{(2)}} \approx 5.70/1.90 = 3.00
\]

\[
\lambda_1^{(3)} = \frac{(\overrightarrow{x}^{(3)})^T A \overrightarrow{x}^{(3)}}{(\overrightarrow{x}^{(3)})^T \overrightarrow{x}^{(3)}} \approx 5.88/1.96 = 3.00
\]

The actual \( \lambda_1 = 3 \) and \( \overrightarrow{v_1} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \). Note that if \( \overrightarrow{x}^{(0)} = [1, -1]^T \) which is \( \overrightarrow{v_2} \), then the power method fails. Since \( \left| \frac{\lambda_2}{\lambda_1} \right| = \frac{1}{3} \) is small, the convergence is fast.

Applications: Power method is used to calculate Google PageRank and to find recommendations of who to follow in Twitter.
7 Additional Topics

In this chapter we cover a few additional topics and their numerical methods.

7.1 Parabolic PDE: Heat Equation

The one-dimensional heat equation is

\[
\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t), \quad 0 < x < l, \quad t > 0,
\]

\[u(0,t) = u(l,t) = 0, \quad t > 0, \quad \text{and} \quad u(x,0) = f(x), \quad 0 < x < l.
\]

First we make a grid of the domain of \((x,t)\) with step size \(h\) and \(k\) respectively:

\[
x_i = ih, \quad i = 0, 1, \ldots, m. \quad (h = (l - 0)/m)
\]

\[
t_j = jk, \quad j = 0, 1, \ldots
\]

Now we approximate \(u(x_i,t_j)\) by \(u_{i,j}\) using the Difference Method:

By the FDF in variable \(t\), we get

\[
\frac{\partial u}{\partial t}(x_i,t_j) = \frac{u(x_i,t_j + k) - u(x_i,t_j)}{k} - O(k) \approx \frac{u_{i,j+1} - u_{i,j}}{k}.
\]

Similarly by the CDF in variable \(x\), we get

\[
\frac{\partial^2 u}{\partial x^2}(x_i,t_j) = \frac{u(x_i + h,t_j) - 2u(x_i,t_j) + u(x_i - h,t_j)}{h^2} - O(h^2) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}.
\]

Plugging these \(u_t(x_i,t_j)\) and \(u_{xx}(x_i,t_j)\) into the heat equation, we get

\[
\frac{u_{i,j+1} - u_{i,j}}{k} = \alpha^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}
\]

\[\Rightarrow u_{i,j+1} = \lambda u_{i+1,j} + (1 - 2\lambda)u_{i,j} + \lambda u_{i-1,j} \quad (\lambda = k\alpha^2/h^2)
\]

\[\Rightarrow u_{i,j+1} = (1 - 2\lambda)u_{i,j} + \lambda(u_{i+1,j} + u_{i-1,j}), \quad i = 1, \ldots, m - 1, \quad j = 0, 1, 2, \ldots
\]

By the initial condition \(u(x,0) = f(x)\), we have

\[
u_{0,0} = f(x_0), \quad u_{1,0} = f(x_1), \ldots, \quad u_{m,0} = f(x_m).
\]

Using \(u_{0,0}, u_{1,0}, \ldots, u_{m,0}\), we can find \(u_{0,1}, u_{1,1}, \ldots, u_{m,1}\) and so on at different values of \(t_j\).
Approximate the solution of the following heat equation at example.

Note
\[
\begin{bmatrix}
(1 - 2\lambda) & \lambda & 0 & \ldots & 0 \\
\lambda & (1 - 2\lambda) & \ddots & \ddots & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \lambda \\
0 & \ldots & 0 & \lambda & (1 - 2\lambda)
\end{bmatrix}
\begin{bmatrix}
u_{1,j} \\
u_{2,j} \\
\vdots \\
u_{m-1,j}
\end{bmatrix}
= \begin{bmatrix}
u_{1,j+1} \\
u_{2,j+1} \\
\vdots \\
u_{m-1,j+1}
\end{bmatrix}
\]

If we write \( \overrightarrow{u}^{(j)} = [u_{1,j}, \ldots, u_{m-1,j}]^T \), then it becomes
\[
\overrightarrow{u}^{(j+1)} = A \overrightarrow{u}^{(j)}, \quad j = 0, 1, 2, \ldots
\]
\[
\overrightarrow{u}^{(0)} = [f(x_1), \ldots, f(x_{m-1})]^T.
\]

The order of error is \( O(k+h^2) \). It can be shown that the difference method is conditionally stable if \( \rho(A) < 1 \) which implies (steps skipped) if
\[
\lambda = \frac{k\alpha^2}{h^2} \leq \frac{1}{2}.
\]

**Example.** Approximate the solution of the following heat equation at \( t = 0.4 \) by the difference method using \( h = 0.25 \) and \( k = 0.2 \).

\[
\frac{\partial u}{\partial t}(x,t) = \frac{1}{16} \frac{\partial^2 u}{\partial x^2}(x,t), \quad 0 < x < 1, \quad t \geq 0,
\]
\[
u(0,t) = u(1,t) = 0, \quad t > 0, \quad \text{and} \quad u(x,0) = \sin(2\pi x), \quad 0 < x < 1.
\]

Compare the result at \( t = 0.4 \) using the exact solution \( u(x,t) = e^{-\pi^2 t/4} \sin(2\pi x) \).

**Solution.** \( h = 0.25, \quad k = 0.2, \quad \alpha^2 = 1/16. \) So \( \lambda = k\alpha^2/h^2 = 1/5 \) and \( m = (l-0)/h = 4 \).

\( x_0 = 0, x_1 = 0.25, x_2 = 0.5, x_3 = 0.75, x_4 = 1 \)

Here \( f(x) = \sin(2\pi x) \). So \( u_{0,0} = f(x_0) = 0, u_{1,0} = f(x_1) = 1, u_{2,0} = f(x_2) = 0, u_{3,0} = f(x_3) = -1, u_{4,0} = f(x_4) = 0 \).

For \( \overrightarrow{u}^{(j)} = [u_{1,j}, u_{2,j}, u_{3,j}]^T \),
\[
\overrightarrow{u}^{(j+1)} = A \overrightarrow{u}^{(j)}, \quad \text{i.e.,} \quad \begin{bmatrix}
u_{1,j+1} \\
u_{2,j+1} \\
u_{3,j+1}
\end{bmatrix} = \begin{bmatrix}0.6 & 0.2 & 0.2 & 0.6 & 0 \end{bmatrix} \begin{bmatrix}
u_{1,j} \\
u_{2,j} \\
u_{3,j}
\end{bmatrix}, \quad j = 0, 1, 2, \ldots
\]
\[
\overrightarrow{u}^{(0)} = [f(x_1), f(x_2), f(x_3)]^T, \quad \text{i.e.,} \quad \begin{bmatrix}
u_{1,0} \\
u_{2,0} \\
u_{3,0}
\end{bmatrix} = \begin{bmatrix}1 & 0 & -1
\end{bmatrix}.
\]

<table>
<thead>
<tr>
<th>i \</th>
<th>x_i \</th>
<th>u_{i,0} \</th>
<th>u_{i,1} \</th>
<th>u_{i,2} \</th>
<th>u(x_i, 0.4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>1</td>
<td>0.60</td>
<td>0.36</td>
<td>0.37</td>
</tr>
<tr>
<td>2</td>
<td>0.50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>-1</td>
<td>-0.60</td>
<td>-0.36</td>
<td>-0.37</td>
</tr>
<tr>
<td>4</td>
<td>1.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
7.2 Elliptic PDE: Laplace Equation

The steady-state heat distribution on a rectangular plate \( R = [0, a] \times [0, b] \) is modeled by Laplace equation:

\[
\frac{\partial^2 u}{\partial x^2}(x, y) + \frac{\partial^2 u}{\partial y^2}(x, y) = 0, \quad 0 < x < a, \quad 0 < y < b,
\]

\[
u(x, y) = g(x, y) \text{ on the boundary of } R.
\]

First we make a grid on \( R = [0, a] \times [0, b] \) with step size \( h \) and \( k \):

\[
x_i = ih, \quad i = 0, 1, \ldots, m. \quad (m = a/h)
\]

\[
y_j = jk, \quad j = 0, 1, \ldots, n. \quad (n = b/k)
\]

Now we approximate \( u(x_i, y_j) \) by \( u_{i,j} \) using the Difference Method:

By the CDF in variable \( x \), we get

\[
\frac{\partial^2 u}{\partial x^2}(x_i, y_j) = \frac{u(x_i + h, y_j) - 2u(x_i, y_j) + u(x_i - h, y_j)}{h^2} \approx \frac{u_{i+1, j} - 2u_{i,j} + u_{i-1, j}}{h^2}.
\]

Similarly for variable \( y \), we get

\[
\frac{\partial^2 u}{\partial y^2}(x_i, y_j) = \frac{u(x_i, y_j + k) - 2u(x_i, y_j) + u(x_i, y_j - k)}{k^2} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2}.
\]

Plugging these \( u_{xx}(x_i, y_j) \) and \( u_{yy}(x_i, y_j) \) into Laplace equation, we get

\[
u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} = 0
\]

\[
\Rightarrow u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + \frac{h^2}{k^2} (u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) = 0
\]

\[
\Rightarrow -2 \left[ 1 + \left( \frac{h}{k} \right)^2 \right] u_{i,j} + u_{i+1,j} + u_{i-1,j} + \left( \frac{h}{k} \right)^2 [u_{i,j+1} + u_{i,j-1}] = 0
\]

\[
\Rightarrow 2 \left[ 1 + \left( \frac{h}{k} \right)^2 \right] u_{i,j} - u_{i+1,j} - u_{i-1,j} - \left( \frac{h}{k} \right)^2 [u_{i,j+1} + u_{i,j-1}] = 0, \quad i = 1, \ldots, m - 1, \quad j = 1, \ldots, n - 1.
\]
By the initial condition $u(x, y) = g(x, y)$, we have
\[ u_{i,0} = u(x_i, 0), u_{i,n} = u(x_i, b), \quad i = 0, 1, \ldots, m, \]
\[ u_{0,j} = u(0, y_j), u_{m,j} = u(a, y_j), \quad j = 0, 1, \ldots, n. \]

Using $u_{i,0}, u_{i,n}, u_{0,j}, u_{m,j}$, we can find $u_{i,j}$ from $(m-1) \times (n-1)$ equations in $(m-1) \times (n-1)$ variables corresponding to $(m-1) \times (n-1)$ inner mesh points. We simplify the equations by introducing variables $w_k = u(i, j)$ for $k = i + (n-1-j)(m-1)$ where $i = 1, 2, \ldots, m-1$ and $j = 1, 2, \ldots, n-1$.

**Example.** Use $h = k = 1$ to approximate the steady-state heat distribution on a thin rectangular 4 m by 3 m plate modeled by the following PDE:
\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad 0 < x < 4, \quad 0 < y < 3, \]
\[ u(x, 0) = 0, \quad u(x, 3) = 1.5x, \quad 0 \leq x \leq 4, \quad \text{and} \quad u(0, y) = 0, \quad u(4, y) = 2y, \quad 0 \leq y \leq 3. \]

Compare the results with the actual solution $u(x, y) = xy/2$.

**Solution.** $h = k = 1 \implies (h/k)^2 = 1$, $m = a/h = 4$, and $n = b/k = 3$.

\[ x_0 = 0, \quad x_1 = 1, \quad x_2 = 2, \quad x_3 = 3, \quad x_4 = 4 \]
\[ y_0 = 0, \quad y_1 = 1, \quad y_2 = 2, \quad y_3 = 3 \]

Suppose $u_{i,j} \approx u(x_i, y_j)$ for all $i, j$. By the boundary conditions,
\[ u_{0,0} = u_{1,0} = u_{2,0} = u_{3,0} = u_{4,0} = u_{0,1} = u_{0,2} = u_{0,3} = 0. \]
\[ u_{1,3} = 1.5, \quad u_{2,3} = 3, \quad u_{3,3} = 4.5, \quad u_{4,3} = 6, \quad u_{4,1} = 2, \quad u_{4,2} = 4. \]

By the Difference Method,
\[ 2 \left[ 1 + \left( \frac{h}{k} \right)^2 \right] u_{i,j} - u_{i+1,j} - u_{i-1,j} - \left( \frac{h}{k} \right)^2 [u_{i,j+1} + u_{i,j-1}] = 0 \]
\[ \implies 4u_{i,j} - u_{i+1,j} - u_{i-1,j} - u_{i,j+1} - u_{i,j-1} = 0, \quad i = 1, 2, 3, \quad j = 1, 2. \]
Suppose \( w_1 = u_{1,2}, w_2 = u_{2,2}, w_3 = u_{3,2}, w_4 = u_{1,1}, w_5 = u_{2,1}, w_6 = u_{3,1} \).

At \((x_1, y_2)\) : \(4u_{1,2} - u_{2,2} - u_{0,2} - u_{1,3} - u_{1,1} = 0 \implies 4w_1 - w_2 - w_4 = 1.5\)

At \((x_2, y_2)\) : \(4u_{2,2} - u_{3,2} - u_{1,2} - u_{2,3} - u_{2,1} = 0 \implies -w_1 + 4w_2 - w_3 - w_5 = 3\)

At \((x_3, y_2)\) : \(4u_{3,2} - u_{4,2} - u_{2,2} - u_{3,3} - u_{3,1} = 0 \implies -w_2 + 4w_3 - w_6 = 8.5\)

At \((x_1, y_1)\) : \(4u_{1,1} - u_{2,1} - u_{0,1} - u_{1,2} - u_{1,0} = 0 \implies -w_1 + 4w_4 - w_5 = 0\)

At \((x_2, y_1)\) : \(4u_{2,1} - u_{3,1} - u_{1,1} - u_{2,2} - u_{2,0} = 0 \implies -w_2 - w_1 + 4w_5 - w_6 = 0\)

At \((x_3, y_1)\) : \(4u_{3,1} - u_{4,1} - u_{2,1} - u_{3,2} - u_{3,0} = 0 \implies -w_3 - w_5 + 4w_6 = 2\).

In matrix form,

\[
A \vec{w} = \vec{b}, \text{ i.e.,} \begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 \\ -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} = \begin{bmatrix} 1.5 \\ 3 \\ 8.5 \\ 0 \\ 0 \\ 2 \end{bmatrix}.
\]

Solving we get, \( \vec{w} = [1, 2, 3, 0.5, 1, 1.5]^T \).

\[
\begin{array}{cccc}
(x_i, y_j) & k & w_k & u(x_i, y_j) = x_i y_j/2 \\
(1, 2) & 1 & 1 & 1 \\
(2, 2) & 2 & 2 & 2 \\
(3, 2) & 3 & 3 & 3 \\
(1, 1) & 4 & 0.5 & 0.5 \\
(2, 1) & 5 & 1 & 1 \\
(3, 1) & 6 & 1.5 & 1.5 \\
\end{array}
\]
7.3 Least Squares Approximation

Suppose we have a set of data \((x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)\) where \(x_0, x_1, \ldots, x_n\) are distinct. In section 3.1 we found a unique interpolating polynomial \(P_n\) of degree at most \(n\) such that \(P_n(x_i) = y_i, \ i = 0, 1, \ldots, n\). Now suppose there are errors in the data. Then it is not a good idea to approximate the data by \(P_n\) that passes through each point \((x_i, y_i)\). Rather we find a polynomial \(P_k\) of degree \(k < n\) which does not go 'far' from each point \((x_i, y_i)\):

If the data is approximated by \(P_k(x) = c_0 + c_1 x + \cdots + c_k x^k\), then we find such \(c_0, c_1, \ldots, c_k\) that minimizes the sum of squared errors:

\[
E = \sum_{i=0}^{n} (y_i - P_k(x_i))^2 = \sum_{i=0}^{n} y_i^2 - 2 \sum_{i=0}^{n} y_i P_k(x_i) + \sum_{i=0}^{n} (P_k(x_i))^2.
\]

Putting \(E\) in terms of \(c_o, c_1, \ldots, c_k\),

\[
E = \sum_{i=0}^{n} y_i^2 - 2 \sum_{i=0}^{n} y_i \left( \sum_{j=0}^{k} c_j x_i^j \right) + \sum_{i=0}^{n} \left( \sum_{j=0}^{k} c_j x_i^j \right)^2
\]

\[
= \sum_{i=0}^{n} y_i^2 - 2 \sum_{j=0}^{k} c_j \left( \sum_{i=0}^{n} y_i x_i^j \right) + \sum_{j=0}^{k} c_j \sum_{m=0}^{k} c_m \left( \sum_{i=0}^{n} x_i^{j+m} \right).
\]

To minimize \(E\), we set \(\frac{\partial E}{\partial c_j} = 0, j = 0, \ldots, k\), which gives

\[
\frac{\partial E}{\partial c_j} = -2 \sum_{i=0}^{n} y_i x_i^j + 2 \sum_{m=0}^{k} c_m \left( \sum_{i=0}^{n} x_i^{j+m} \right) = 0 \implies \sum_{m=0}^{k} c_m \left( \sum_{i=0}^{n} x_i^{j+m} \right) = \sum_{i=0}^{n} y_i x_i^j.
\]

This gives a system of \(k + 1\) equations in \(k + 1\) variables \(c_o, c_1, \ldots, c_k\):

\[
A \vec{c} = \vec{b}, \text{ where } A = \begin{bmatrix}
\sum_{i=0}^{n} x_i^0 & \sum_{i=0}^{n} x_i^1 & \cdots & \sum_{i=0}^{n} x_i^k \\
\sum_{i=0}^{n} x_i^1 & \sum_{i=0}^{n} x_i^2 & \cdots & \sum_{i=0}^{n} x_i^{k+1} \\
\vdots & \vdots & \ddots & \vdots \\
\sum_{i=0}^{n} x_i^k & \sum_{i=0}^{n} x_i^{k+1} & \cdots & \sum_{i=0}^{n} x_i^{2k}
\end{bmatrix}, \vec{c} = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_k \end{bmatrix}, \text{ and } \vec{b} = \begin{bmatrix} \sum_{i=0}^{n} y_i x_i^0 \\ \sum_{i=0}^{n} y_i x_i^1 \\ \vdots \\ \sum_{i=0}^{n} y_i x_i^k \end{bmatrix}.
\]

It can be shown that \(A\) is invertible when \(x_0, x_1, \ldots, x_n\) are distinct. Thus we have a unique solution \(\vec{c} = A^{-1} \vec{b}\) for \(c_0, c_1, \ldots, c_k\) giving a unique \(P_k(x) = c_0 + c_1 x + \cdots + c_k x^k\) that minimizes the least squares error \(E = \sum_{i=0}^{n} (y_i - P_k(x_i))^2\).

Example. Find the least squares polynomials of degree 1 and 2 for the following data:

<table>
<thead>
<tr>
<th>x</th>
<th>0</th>
<th>0.5</th>
<th>1</th>
<th>1.5</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>1.35</td>
<td>1.52</td>
<td>1.78</td>
<td>2.41</td>
</tr>
</tbody>
</table>
**Solution.** To find $P_1(x) = c_0 + c_1 x$, we solve the following system:

\[
\begin{align*}
    c_0 \sum_{i=0}^{4} x_i^0 &+ c_1 \sum_{i=0}^{4} x_i^1 = \sum_{i=0}^{4} y_i x_i^0 \\
    c_0 \sum_{i=0}^{4} x_i^1 &+ c_1 \sum_{i=0}^{4} x_i^2 = \sum_{i=0}^{4} y_i x_i^1
\end{align*}
\]

Numerically solving the system

\[
\begin{align*}
    5c_0 &+ 5c_1 = 8.06 \\
    5c_0 &+ 7.5c_1 = 9.685
\end{align*}
\]

we get $(c_0, c_1) = (0.962, 0.65)$ giving $P_1(x) = 0.962 + 0.65x$.

Similarly to find $P_2(x) = c_0 + c_1 x + c_2 x^2$, we solve the following system:

\[
\begin{align*}
    c_0 \sum_{i=0}^{4} x_i^0 &+ c_1 \sum_{i=0}^{4} x_i^1 + c_2 \sum_{i=0}^{4} x_i^2 = \sum_{i=0}^{4} y_i x_i^0 \\
    c_0 \sum_{i=0}^{4} x_i^1 &+ c_1 \sum_{i=0}^{4} x_i^2 + c_2 \sum_{i=0}^{4} x_i^3 = \sum_{i=0}^{4} y_i x_i^1 \\
    c_0 \sum_{i=0}^{4} x_i^2 &+ c_1 \sum_{i=0}^{4} x_i^3 + c_2 \sum_{i=0}^{4} x_i^4 = \sum_{i=0}^{4} y_i x_i^2
\end{align*}
\]

Numerically solving the system, we get $(c_0, c_1, c_2) = (1.055, 0.2786, 0.1857)$ giving $P_2(x) = 1.055 + 0.2786x + 0.1857x^2$.

In statistics simple linear regression finds a degree one polynomial to predict a $y$ value for a given $x$ value from a data set with errors.