Emory University
MATH 315 Numerical Analysis
Learning Notes

Jiuru Lyu

November 25, 2023

Contents

1 Floating Point Numbers **3**
- 1.1 Binary Representation . . . 3
- 1.2 Integers in Computers . . . 3
- 1.3 Representation of Floating Point Numbers . . . 4
- 1.4 Errors . . . 7

2 Solutions of Linear Systems **11**
- 2.1 Simply Solved Linear Systems . . . 11
- 2.2 GEPP and Matrix Factorization . . . 15
- 2.3 Measuring Accuracy of Solutions . . . 18

3 Curve Fitting **25**
- 3.1 Polynomial Interpolation . . . 25
- 3.2 Error in Polynomial Interpolation . . . 29
- 3.3 Least Square . . . 34

4 Differentiation and Integration **39**
- 4.1 Review - Taylor Series . . . 39
- 4.2 Differentiation . . . 39
- 4.3 Integration . . . 40
- 4.4 Error in Integration . . . 44
- 4.5 Adaptive Integration and More . . . 49

5 Root Finding **51**
- 5.1 Fixed Point Iteration (FPI) . . . 51
- 5.2 Convergence Iteration Analysis . . . 52

5.3 Root Finding Methods . . . 53
5.4 Calculating Square Roots . . . 56
5.5 Roots of Polynomials . . . 57

6 Univariate Minimization **61**
6.1 Find Minima Without Calculus . . . 61

1 Floating Point Numbers

1.1 Binary Representation

Definition 1.1.1 (Binary). 0 and 1; on and off.

Example 1.1.2 Represent Numbers in Base-2

Consider $13 = 1(10) + 3(1) = 1(10) + 3(10^0)$ in base-10. It can be converted into base-2 by decomposing 13 as $1(2^3) + 1(2^2) + 0(2^1) + 1(2^0)$.

Example 1.1.3 Fractions in Base-2

$$\frac{7}{16} = \frac{1}{16}(7) = \left(2^{-4}\right)\left(2^2 + 2^1 + 2^0\right) = 2^{-2} + 2^{-3} + 2^{-4}.$$

Example 1.1.4 Repeating Fractions in Base-2

$$\begin{aligned} \frac{1}{5} &= \frac{1}{8} + \varepsilon_1 \quad \Longrightarrow \quad \varepsilon_1 = \frac{1}{5} - \frac{1}{8} = \frac{8-5}{(5 \times 8)} = \frac{3}{40} \\ \varepsilon_1 &= \frac{3}{3(16)} + \varepsilon_2 \quad \Longrightarrow \quad \cdots \end{aligned}$$

Repeating the steps above, we would finally get

$$\frac{1}{5} = \frac{1}{8} + \frac{1}{16} + \frac{1}{128} + \frac{1}{256} + \cdots$$

Theorem 1.1.5

Let $n \in \mathbb{Z}$ and $n \geq 1$, then

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

1.2 Integers in Computers

Definition 1.2.1 (Storing Integers). `unit8` stands for unsigned integers and `int8` stands for signed integers.

Remark. 1.1 *The 8 here represents 8 bits. It is a measure of how much storage (how many 0s or 1s).*

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
unsigned:	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
signed:	-2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Example 1.2.2

$$\texttt{unit8}(13) = 00001101$$

Since $-13 = 1(-2^7) + 1(2^6) + 1(2^5) + 1(2^4) + 0(2^3) + 0(2^2) + 1(2^1) + 1(2^0)$, we have

$$\texttt{int8}(-13) = 11110011$$

Remark. 1.2 *Largest and Smallest Integers:*

$$\begin{aligned}
\texttt{uint8}(x_L) &= 11111111 &&\Longrightarrow x_L = 2^7 + 2^6 + \cdots + 2^0 = 2^8 - 1 = 255\\
\texttt{uint8}(x_S) &= 00000000 &&\Longrightarrow x_S = 0(2^7) + 0(2^6) + \cdots + 0(2^0) = 0\\
\texttt{int8}(x_L) &= 01111111 &&\Longrightarrow x_L = 0(-2^7) + 2^6 + \cdots + 2^0 = 2^7 - 1 = 127\\
\texttt{int8}(x_S) &= 100000000 &&\Longrightarrow x_S = 1(-2^7) + 0(2^6) + \cdots + 0(2^0) = -128
\end{aligned}$$

1.3 Representation of Floating Point Numbers

Definition 1.3.1 (Normalized Scientific Notation). Only 1 digit (non-zero) to the left of the decimal point.

Example 1.3.2

$$\begin{aligned}
&123.456 \times 10^7\\
&12.3456 \times 10^8\\
&1.23456 \times 10^9 \to \text{normalized}
\end{aligned}$$

Definition 1.3.3 (Anatomy of Floating Point Numbers). A floating point number, $\texttt{float}(x)$, consists of three parts: $s(x)$ (sign bit), $e(x)$ (exponent bits), and $f(x)$ (fraction bits).
Definition 1.3.4 (Precision). Precision is defined by the number of bits per part:

	$s(x)$	$e(x)$	$f(x)$	total
double precision (DP)	1	11	52	64
single precision (SP)	1	8	23	32
half precision (HP)	1	5	10	16

Remark. 1.3 *The less bits the float point number has, the less storage it requires and faster computation it performs, but more error introduces.*

Definition 1.3.5 (Floating Point Number).

$$\texttt{float}(x) = (-1)^{s(x)}\left(1 + \frac{f(x)}{2^{\text{\# of fraction bits}}}\right)2^{E(x)}, \tag{1}$$

where $E(x)$ is called the *unbiased exponent* because it is centered about 0 and is calculated through the $e(x)$, the *biased exponent* because it can only be non-negative integers, by the following formula:

$$E(x) = e(x) - \left(2^{\text{\# of exponent bits}-1} - 1\right).$$

Remark. 1.4 *Eq. (1) is in normalized scientific notation because the largest number $f(x)$ can represent is $2^{\text{\# of fraction bits}} - 1$. Hence,*

$$1 + \frac{f(x)}{2^{\text{\# of fraction bits}}} < 2,$$

and thus there will be only 1 digit in front of the decimal point.

Example 1.3.6 Formula for a Floating Point Number in Double Precision (DP)

$$\texttt{float}_{\text{DP}}(x) = (-1)^{s(x)}\left(1 + \frac{f(x)}{2^{52}}\right)2^{e(x)-1023}.$$

Example 1.3.7 Converting DP into Decimal

Suppose a DP floating number is stored as $s(x) = 0$, $e(x) = 10000000011$, and $f(x) = 0100100\cdots0$. Find its representation in decimal base-10.

Solution 1.

$e(x) = 10000000011 = 2^{10} + 2^1 + 2^0$ and $f(x) = 0100100\cdots0 = 2^{50} + 2^{47}$. Then, the unbiased exponent $E(x) = e(x) - 1023 = 2^{10} + 2^1 + 2^0 - (2^{10} - 1) = 4$. So,

$$\begin{aligned}
\texttt{float}_{\text{DP}}(x) &= (-1)^{s(x)} + \left(1 + \frac{f(x)}{2^{52}}\right)2^{E(x)} \\
&= (-1)^0\left(1 + \frac{2^{50} + 2^{47}}{2^{52}}\right)2^4 \\
&= \left(1 + 2^{-2} + 2^{-5}\right)2^4 \\
&= 2^4 + 2^2 + 2^{-1} \\
&= 16 + 4 + 0.5 = 20.5
\end{aligned}$$

□

Example 1.3.8 Converting Value to DP

Suppose a number in base-10 is -10.75. Find its representation of floating point number under DP.

Solution 2.

We have

$$\begin{aligned}
\texttt{value}(x) = -10.75 &= (-1)(10+0.75) \\
&= (-1)\big(2^3+2^1+2^{-1}+2^{-2}\big) \\
&= (-1)\big(1+2^{-2}+2^{-4}+2^{-5}\big)2^3 \quad \big[\text{In normalized scientific notation}\big] \\
&= (-1)^1\left(1+\frac{2^{50}+2^{48}+2^{47}}{2^{52}}\right)2^{1026-1023} \\
&= (-1)^1\left(1+\frac{2^{50}+2^{48}+2^{47}}{2^{52}}\right)2^{2^{10}+2^1-1023}
\end{aligned}$$

So, we have $s(x)=1$, $e(X)=10000000010$, and $f(x)=010110\cdots 0$. $\square$

Theorem 1.3.9 Some Special Rules

1. The formula
$$\texttt{value}(x) = (-1)^{s(x)} + \left(1+\frac{f(x)}{2^{52}}\right)2^{e(x)-1023}$$
only holds when $0 < e(x) < 2^{11}-1$ or $00\cdots01 < e(x) < 11\cdots10$.

2. If $e(x) = 11\cdots1$, then it encodes special numbers.

3. If $e(x) = 00\cdots0$:
   - If $f(x) = 00\cdots0$, then $\texttt{value}(x) = 0$.
   - If $f(x) > 0$, it encodes a *denormalized floating point number*:
   $$\texttt{value}(x) = (-1)^{s(x)}\left(0+\frac{f(x)}{2^{52}}\right)2^{-1022}.$$
   This denormalized floating point number is more precise when describing really small things.

Definition 1.3.10 (Machine Epsilon/$\varepsilon_{\mathbf{WP}}$). Let "WP" stands for the working precision (DP/SP/HP/etc.). The *machine epsilon*, denoted as ε_{WP}, is the gap between 1 and the next largest floating point number. Equivalently, it can be viewed as the smallest possible non-zero value of $\frac{f(x)}{2^{\text{number of fraction bits}}}$. So, $\varepsilon_{\text{DP}} = 2^{-52}$, $\varepsilon_{\text{SP}} = 2^{-23}$, and $\varepsilon_{\text{HP}} = 2^{-10}$.

Definition 1.3.11 (Special Numbers).

1. ± 0: when $s(x) = \pm 1$ and $e(x) = f(x) = 0$.

2. $\pm$`Inf`

3. `NaN`: not-a-number

Definition 1.3.12 (Floating Point Arithmetic).

1. The set of real numbers, $\mathbb{R}$, is closed under arithmetic operations.

2. The set of all WP floating point numbers, however, is not closed under arithmetic operations. For example, $\texttt{float}_{\text{DP}}(x) = \texttt{float}_{\text{DP}}(y) = 2^{52} + 1$, but $xy = 2^{104} + \varepsilon$ cannot be represented using DP.

3. Suppose x and y are floating point numbers, then $x \oplus y = \texttt{float}(x + y)$ and $x \otimes y = \texttt{float}(xy)$. Consider `float` as a rounding process, we can also define subtraction and division of floating point numbers.

Example 1.3.13

Assume we are only allowed three significant digits (in Base-10) in a computer. Suppose $x = 1.23 \times 10^4$ and $y = 6.54 \times 10^3$. Find $x \oplus y = \texttt{float}(x + y)$.

Solution 3.

$$
\begin{aligned}
x \oplus y &= \texttt{float}(x + y) \\
&= \texttt{float}(1.23 \times 10^4 + 6.54 \times 10^3) \\
&= \texttt{float}(1.23 \times 10^4 + 0.654 \times 10^3) \\
&= \texttt{float}(1.884 \times 10^4) \\
&= 1.88 \times 10^4.
\end{aligned}
$$

□

1.4 Errors

Definition 1.4.1 (Errors We May See).

1. *Overflow*: The exponent is too large. This means $|x|$ is large and the computer will represent it as $\pm$`Inf`. Note: In DP, $x_{\text{large}} = (2 - 2^{-52}) \times 2^{1023} \approx 1.798 \times 10^{308}$. This number is referred as `realmax` in MATLAB.

2. *Underflow*: Large negative exponent. This means $|x|$ is tiny and the computer will represent it as ± 0. Note: In SP, $x_{\text{small}} \approx 2.225 \times 10^{-53}$ and is referred as `realmin` in MATLAB.

3. *Roundoff error*: cutoff or round at some point.

Note that sometimes we are encounter the catastrophic cancellation, meaning the subtraction leads to our loss of significance or information. In this case, it is different from underflow error or roundoff error.

Example 1.4.2 Catastrophic Cancellation/Loss of Significance Due to Subtraction

$$\begin{aligned} x &= 3.141592920353983 \approx \frac{355}{113} && \text{16 digits} \\ y &= 3.141592653589794 \approx \pi && \text{16 digits} \\ x - y &= 0.000000266764189 && \text{9 digits} \end{aligned}$$

Definition 1.4.3 (Relative Error). Let $z \in \mathbb{R}$. The relative error between $\texttt{float}(z)$ and z is denoted as μ and

$$\mu = \frac{\texttt{float}(z) - z}{z}$$
$$\texttt{float}(z) = z(1+\mu),$$

where we know

$$|\mu| \leq \frac{\varepsilon_{\text{WP}}}{2}.$$

Example 1.4.4 Propagation of Errors

There are two major sources of errors: storing number and arithmetics.

Consider a computer only allow 3 significant figures. Then $\varepsilon_{\text{WP}} = 0.01$.

Consider $x = \frac{1}{3}$, $y = \frac{8}{7}$, and $x + y = \frac{31}{21}$. Then,

$$\texttt{float}(x) = 0.333 = 3.33 \times 10^{-1} = x(1+\mu_x).$$

Solving for μ_x:

$$\begin{aligned} \frac{333}{1000} &= \frac{1}{3}(1+\mu_x) \\ \mu_x &= \frac{999}{1000} - 1 = \frac{-1}{1000} = -0.001 \end{aligned}$$

Note that $|\mu_x| = 0.01 < \frac{\varepsilon_{\text{WP}}}{2}$. Similarly, we can solve $\texttt{float}(y) = 1.14 \times 10^0 = y(1+\mu_y)$ for $|\mu_y| = 0.0025$. Now, consider the floating point addition

$$\begin{aligned} x \oplus y &= \texttt{float}(\texttt{float}(x) + \texttt{float}(y)) \\ &= \texttt{float}(3.33 \times 10^{-1} + 1.14 \times 10^0) \\ &= \texttt{float}(1.473 \times 10^0) \\ &= 1.47 \times 10^0. \end{aligned}$$

Also, solve $x \oplus y = (x+y)(1+\mu_a)$ for $|\mu_a| = 0.0042$. Note that

$$|\mu_x| + |\mu_y| = 0.0035 < 0.0042 = |\mu_a|.$$

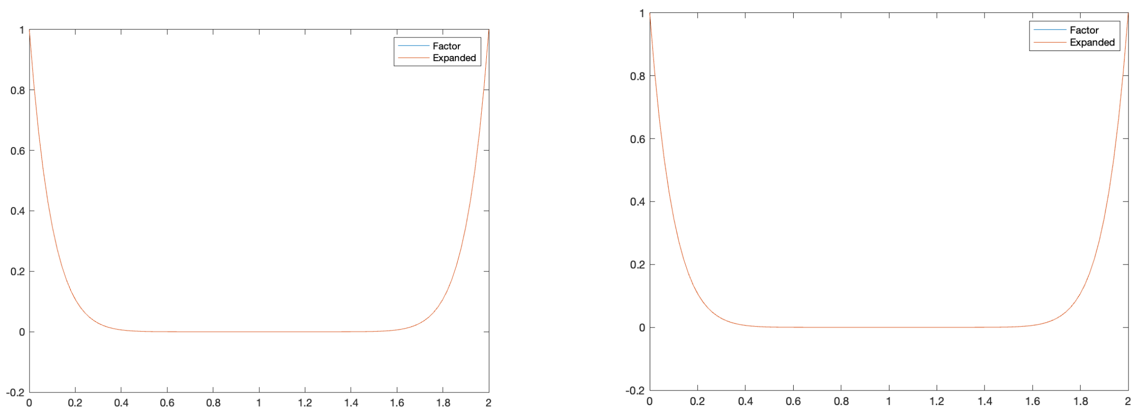
This is called the propagation of error.

Example 1.4.5 Plotting Exponentials Using Factored and Expanded Forms

Consider $p(x) = (1-x)^{10}$ and its expanded form. Plot them to see which is better.

Example 1.4.5

```
%% Defining the Functions
p_1 = @(x) (1-x).^10;
p_2 = @(x) x.^10-10*x.^9+45*x.^8-120*x.^7+210*x.^6-252*x.^5+...
        210*x.^4-120*x.^3+45*x.^2-10*x+1;
%% Ploting the Functions
x = linspace(0, 2, 100);
plot(x, p_1(x))
hold on
plot(x, p_2(x))
legend("Factor", "Expanded")
%% Zooming In
y = linspace(0.99, 1.01, 100);
hold off
plot(y, p_1(y))
hold on
plot(y, p_2(y))
legend("Factor", "Expanded")
```



(a) Plotting Functions (b) Zooming In

It seems that the two functions are the same (Fig 1(a)); however if we zooming in (Fig 1(b)), the expanded version introduces more error than the factored version because the expanded version requires more arithmetical operations in it.

2 Solutions of Linear Systems

Remark. 2.1 *Assumption throughout this chapter:* $\mathbf{A}$ *is a square* $n \times n$ *matrix.*

2.1 Simply Solved Linear Systems

Definition 2.1.1 (Linear System).

- Equation form: x_i are variables (what we solve for) and a_{ij} are coefficients:

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

  This system has n equations and n variables.

- Matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \implies \mathbf{A}x = b,$$

  where $\mathbf{A}$ is the coefficient matrix, a $n \times n$ matrix, x is the unknown, the solution vector with length n, and b is the right hand side, vector with length n.

Theorem 2.1.2 Number of Solutions to a Linear System
A linear system $\mathbf{A}x = b$ could have the following numbers of solutions:

- One unique solution: $\mathbf{A}x = b$ is nonsingular; $\mathbf{A}$ is invertible.
- No solutions: $\mathbf{A}x = b$ is singular.
- Infinite many solutions: $\mathbf{A}x = b$ is singular.

Theorem 2.1.3 Matrix-Vector Multiplication
Let $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$.

- View 1: Row-wise. Let $y = \mathbf{A}x$, then $y_i = \sum_{j=1}^{n} a_{ij}x_j$ as the i^{th} row of y.
- View 2: Column-wise. $\mathbf{A}x$ is a linear combination of columns of $\mathbf{A}$. So, $y = \sum_{j=1}^{n} x_j \vec{\mathbf{a_j}}$, where we regard $\mathbf{A}$ as $\begin{bmatrix} \vec{\mathbf{a_1}} & \vec{\mathbf{a_2}} & \cdots & \vec{\mathbf{a_n}} \end{bmatrix}$

Row-Wise Vector Multiplication

```
y = zeros(n, 1);
for i = 1:n % loop over rows
    for j = 1:n % loop over sum
        y(i) = y(i) + A(i,j) + x(j);
    end
end
```

Row-Wise Vector Multiplication (Vectorization)

```
y = zeros(n, 1);
for i = 1:n % loop over rows
    y(i) = A(i,:) * x(i); % vectorization
end
```

Column-Wise Vector Multiplication

```
y = zeros(n, 1);
for j = 1:n % loop over columns
    y = y + x(j) * A(:, j);
end
```

Definition 2.1.4 (Important Part of a Matrix).

- Diagonal Part
- Strictly Upper Triangular Part
- Strictly Lower Triangular Part

Theorem 2.1.5 Solving Diagonal Matrix
Given

$$\begin{bmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix},$$

we have

$$a_{11}x_1 = b_1; \quad a_{22}x_2 = b_2; \quad \cdots \quad a_{nn}x_n = b_n$$

So we have

$$x_i = \frac{b_i}{a_{ii}},$$

only if $a_{ii} \neq 0$.

Remark. 2.2 $a_{ii} \neq 0$ *holes if* $\mathbf{A}$ *is invertible.*

Remark. 2.3 *A Diagonal matrix is also a lower triangular matrix or an upper triangular matrix.*

Solving Diagonal Matrix

```
x = zeros(n, 1);
for i = 1:n
    x(i) = b(i) / A(i,i); % overflow and underflow
end
```

Theorem 2.1.6 Solving Lower Triangular Systems

Given

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & & a_{nn} \end{bmatrix} \begin{bmatrix} x_2 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ v_n \end{bmatrix},$$

we have

$$\begin{aligned} a_{11}x_1 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

We can use the Forward Substitution to solve:

$$x_i = \frac{b_1 - a_{i1}x_1 - a_{i2}x_2 - \cdots - a_{i(i-1)}x_{i-1}}{a_{ii}.}$$

Algorithm 1: Row-Oriented Forward Substitution

Input: matrix A $= [a_{ij}]$; vector $b = [b_i]$
Output: solution vector $x = [x_i]$

begin
  for *i* = *1* **to** *n* **do** `// loop over rows`
    for *j* = *1* **to** *i-1* **do** `// loop over columns`
      $b_i := b_i - a_{ij}x_j$;
    $x_i := b_i/a_{ii}$;

Example 2.1.7

Given

$$\begin{bmatrix} -5 & & \\ 3 & 3 & \\ 2 & -5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix}.$$

Use column-wise forward substitution to solve this system.

Solution 1.

In column-wise:

$$x_1 \begin{bmatrix} -5 \\ 3 \\ 2 \end{bmatrix} + x_2 \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix}.$$

1. Step 1: Solve for $x_1 = -10/-5 = 2$.
2. Step 2: Plug $x_1 = 2$ into the equation:

$$x_2 \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} + x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \\ 21 \end{bmatrix} - (2) \begin{bmatrix} -5 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 17 \end{bmatrix}.$$

3. Step 3: Solve for $x_2 = -3/3 = -1$.
4. Step 4: Plug $x_2 = -1$ into the equation:

$$x_3 \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ 17 \end{bmatrix} - (-1) \begin{bmatrix} 0 \\ 3 \\ -5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 12 \end{bmatrix}.$$

5. Step 5: Solve for $x_3 = 12/4 = 3$.

□

Algorithm 2: Column-Oriented Forward Substitution

Input: matrix A $= [a_{ij}]$; vector $b = [b_i]$
Output: solution vector $x = [x_i]$

begin
  for *j* = *1* **to** *n* **do**
    $x_j := b_j / a_{jj}$;
    for *i* = *j*+*1* **to** *n* **do**
      $b_i := b_i - a_{ij} x_j$;

Theorem 2.1.8 Computational Cost of Forward Substitution
Number of floating point operations ($+$, $-$, $\times$, $/$) in row i is 1 division, $(i-1)$ multiplications, and $(i-1)$ subtractions. So, Number of floating points operations, or flops, of the algorithm is

$$
\begin{aligned}
\text{flops} = \sum_{i=1}^{n}(1+i-1+i-1) &= \sum_{i=1}^{n}(2i-1) \\
&= 2\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 \\
&= 2\left[\frac{(n+1)(n)}{2}\right] - n \\
&= n^2
\end{aligned}
$$

Remark. 2.4 *It should be the same number of flops if we do column-oriented forward substitution.*

Remark. 2.5 *Solving upper triangular system using backward substitution.*

2.2 GEPP and Matrix Factorization

Theorem 2.2.1 Gaussian Elimination
In Gaussian Elimination, we are allowed to

1. Swap rows (exchange, pivot)
2. Add multiple of one row to another
3. Multiply row by non-zero scalar.

Remark. 2.6 *We require the equation with the largest coefficient in magnitude at the top when doing the Gaussian Elimination. This is because we want to divide by large numbers instead of smaller ones (which will cause errors).*

Algorithm 3: General Structure of GEPP

```
begin
    for all stages do
        pivot;
        eliminate;
```

At stage k, eliminate x_k, from rows $k+1$ to n

```
for i = k+1:n
    m(i,k) = a(i,k) / a(k,k); % find the multiplier
    a(i,k) = 0;
    for j = k+1:n
        a(i,j) = a(i,j) - m(i,k) * a(k,j); % could use vectorization
    end
    b(i) = b(i) - m(i,k) * b(k);
end
```

Pivoting at stage k: find the coefficient with the largest magnitude

```
%% The code tells us which row has the pivot.
p = k;
for i = k+1:n
    if abs(a(p,k)) < abs(a(i,k))
        p = i;
    end
end
%% Swap rows in A and b
A([p,k],:) = A([k,p],:);
b(p) = b(k);
```

Theorem 2.2.2 Cost of GEPP

At stage k, we only focus on rows k through n and columns k through n. We have $(n-k)$ divisions for multipliers. For every multiplier we have $(n-k)$ multiplications, which are then used $(n-k)$ times to change each row. So, we have $(n-k)(n-k)$ multiplications in total. Subtractions come with multiplications, so we also have $(n-k)(n-k)$ subtractions.

Theorem 2.2.3 Another Perspective on GEPP

The process of GEPP can be written as matrix multiplication $\mathbf{EA}$, where $\mathbf{E}$ is an elementary matrix and act on the rows of $\mathbf{A}$.

Theorem 2.2.4 Matrix Factorizations: $\mathbf{PA} = \mathbf{LU}$

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

where $\mathbf{U}$ is upper-triangular, $\mathbf{L}$ is lower-triangular, and $\mathbf{P}$ is the pivot or permutation matrix.

This factorization comes from GEPP. Almost all matrices $\mathbf{A}$ have $\mathbf{PA} = \mathbf{LU}$ unless we have a column of all zeros in $\mathbf{A}$.

Theorem 2.2.5 Solving $\mathbf{A}x = b$ with $\mathbf{PA} = \mathbf{LU}$
Given $\mathbf{A}x + b$ and pre-computed $\mathbf{PA} = \mathbf{LU}$:

$$\begin{aligned}
\mathbf{PA} &= \mathbf{P}b \\
\mathbf{LU} &= \mathbf{P}b \\
\mathbf{U}x &= \mathbf{L}^{-1}\mathbf{P}b && \text{using forward substitution} \\
x &= \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}b && \text{using backward substitution}
\end{aligned}$$

This process need around $\mathcal{O}(n^2)$ operations. $\mathcal{O}(\cdot)$ is the big-O notation, meaning the number is dominated by n^2.

$\mathbf{PA} = \mathbf{LU}$ in MATLAB

```
[L, U, P] = lu(A);
% P could be omitted sometimes.
```

Theorem 2.2.6 Cholesky Factorization
If $\mathbf{A}$ is symmetric ($\mathbf{A} = \mathbf{A}^T$) and positive definite (all eigenvalues are positive), then $\mathbf{A} = \mathbf{R}^T\mathbf{R}$, where $\mathbf{R}$ is upper-triangular and $\mathbf{R}^T$ is lower-triangular. This factorization is 2 time less expensive than GEPP.

Remark. 2.7 *Choleksy Factorization is just the* $\mathbf{PA} = \mathbf{LU}$ *factorization for SPDs (symmetric positive definite matrices).*

Theorem 2.2.7 Other Matrix Factorization

1. QR decomposition:
$$\mathbf{A} = \mathbf{QR},$$
where $\mathbf{R}$ is upper-triangular and $\mathbf{Q}$ is orthogonal such that $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}^T\mathbf{Q} = \mathbf{I}$.
$\mathbf{Q}$ is really easy to be inverted and comes from the Gram-Schmidt process.
This composition is a bit more expensive than $\mathbf{PA} = \mathbf{LU}$.

2. Singular Value Decomposition (SVD):
$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T,$$
where $\mathbf{U}$ and $\mathbf{V}^T$ are orthogonal and $\boldsymbol{\Sigma}$ is diagonal. This factorization is also more expensive than $\mathbf{PA} = \mathbf{LU}$ decomposition.

2.3 Measuring Accuracy of Solutions

Definition 2.3.1 (Vector Norms). A vector norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ that satisfies

- **Positive Definiteness**: $\|x\| \geq 0 \quad \forall x \in \mathbb{R}^n$ and $\|x\| = 0$ if and only if $x = 0$.
- **Positive Homogeneity**: $\|cx\| = |c|\|x\| \quad \forall x \in \mathbb{R}^n$ and $c \in \mathbb{R}$.
- **Triangular Inequality**: $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbb{R}^n$.

Definition 2.3.2 (Common Definitions of Norm).

- Pythagorean Distance: $\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$.
- Taxicab/Manhattan Distance: $\|x\|_1 = |x_1| + |x_2| + \cdots + |x_n|$.
- Infinity Norm: $\|x\|_\infty = \max_{i=1,\ldots,n} |x_i|$.

Proof 1. In this prove, we want to show the $1-$norm is a proper norm.

- **Positive Definiteness**: Note that $\|x\|_1 = |x_1 + + \cdots + ()|x_n| \geq 0$ since each $|x_j| \geq 0$. If one $x_i \neq 0$, $|x_i| \geq 0$, then $\|x\|_1 > 0$. So, if $\|x\|_1 = 0$, it must be $x = 0$. $\square$
- **Positive Homogeneity**:

$$\begin{aligned}\|cx\|_1 &= |cx_1| + |cx_2| + \cdots + |cx_n| \\ &= |c||x_1| + |c||x_2| + \cdots + |c||x_n| \\ &= |c|(|x_1| + |x_2| + \cdots + |x_n|) \\ &= |c|\|x\|_1. \quad \square\end{aligned}$$

- **Triangle Inequality**:

$$\begin{aligned}\|x + y\|_1 &= |x_1 + y_1| + |x_2 + y_2| + \cdots + |x_n + y_n| \\ &\leq |x_1| + |y_1| + |x_2| + |y_2| + \cdots + |x_n| + |y_n| \\ &= (|x_1| + |x_2| + \cdots + |x_n|) + (|y_1| + |y_2| + \cdots + |y_n|) \\ &= \|x\|_1 + \|y\|_1.\end{aligned}$$

■

Definition 2.3.3 (Matrix Norm). A matrix norm is a function $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ *s.t.*

- **Positive Definiteness**: $\|\mathbf{A}\| \geq 0$ and $\|\mathbf{A}\| = 0$ if and only if $\mathbf{A} = 0$.
- **Positive Homogeneity**: $\|c\mathbf{A}\| = |c|\|\mathbf{A}\|$.
- **Triangle Inequality**: $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$.

Definition 2.3.4 (Some Matrix Norms).

1. **Frobenius Norm**:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}^2}.$$

2. **Induced Matrix Norm**: Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, $x \in \mathbb{R}^{n\times 1}$, and $p = 1, 2, \infty, \cdots$, then

$$\|\mathbf{A}\|_p = \max_{x\neq 0} \frac{\|\mathbf{A}x\|_p}{\|x\|_p} = \max_{\|x\|_p=1} \|\mathbf{A}x\|_p.$$

Remark. 2.8 *Induced norm intuition: how much does $\|x\|_p$ change when we apply $\mathbf{A}$?*

Theorem 2.3.5 Induced Matrix Norms with different p's

- $\|\mathbf{A}\|_2 = \sigma_1$, the largest singular value. That is, if $\mathbf{A} = \mathbf{U\Sigma V}^T$, then $\|\mathbf{A}\|_2$ is the largest entry in $\mathbf{\Sigma}$.
- $\|\mathbf{A}\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|$, the maximum column sum.
- $\|\mathbf{A}\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}|$, the maximum row sum.

Proof 2. Let's show that $\|\mathbf{A}\|_1$ is the maximum column sum. We will (1) show $\|\mathbf{A}\|_1 \leq$ the maximum column sum, and (2) find one case when we attain the upper bound. Given $\|x\|_1 = 1$, then

$$\begin{aligned}
\|\mathbf{A}x\|_1 = \sum_{i=1}^{n} \left| \underbrace{\sum_{j=1}^{n} a_{ij}x_j}_{i-\text{th entry of}\mathbf{A}x} \right| &\leq \sum_{i=1}^{n}\sum_{j=1}^{n} |a_{ij}||x_j| \\
&= \sum_{j=1}^{n} |x_j| \left(\sum_{i=1}^{n} |a_{ij}| \right) \\
&\leq \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|, \text{ when } x \text{ has exactly 1 entry equal to } 1.
\end{aligned}$$

When $x = e_{j^*}$ be a standard basis vector with 1 in j^* position, where j^* is the column of $\mathbf{A}$ with maximum column sum, we have

$$\|\mathbf{A}e_{j^*}\|_1 = \|j - \text{th column of } \mathbf{A}\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}|.$$

■

Example 2.3.6

Given that $\mathbf{A} = \begin{bmatrix} -1 & 2 \\ -12 & 9 \end{bmatrix}$, find $\|\mathbf{A}\|_1$ and $\|\mathbf{A}\|_\infty$.

$$\|\mathbf{A}\|_1 = \text{maximum column sum} = \max_{j=1,\ldots,n} \|\mathbf{A}(: .j)\|_1 = 13.$$

$$\|\mathbf{A}\|_\infty = \text{maximum row sum} = 21.$$

Theorem 2.3.7 Submultiplicativity of Induced Norm

$$\|\mathbf{A}x\|_p \leq \|\mathbf{A}\|_p \|x\|_p.$$

Proof 3. By definition, we know $\|\mathbf{A}\|_p = \max_{x \neq 0} \frac{\|\mathbf{A}x\|_p}{\|x\|_p}$. Then,

$$\|\mathbf{A}\|_p \geq \frac{\|\mathbf{A}x\|_p}{\|x\|_p}$$
$$\|\mathbf{A}x\|_p \leq \|\mathbf{A}\|_p \|x\|_p.$$

■

Corollary 2.3.8 $\|\mathbf{AB}\|_p \leq \|\mathbf{A}\|_p \|\mathbf{B}\|_p$.

Definition 2.3.9 (Measuring Erros). Suppose x is the true solution, and $\widehat{x}$ is the approximate solution. Then

$$\textit{Error} = \|\widehat{x} - x\|$$
$$\textit{Relative Error} = \frac{\|\widehat{x} - x\|}{\|x\|}, \quad x \neq 0.$$

Remark. 2.9 *In practice, we do not know x, the true solution. So this measurement cannot be used.*

Definition 2.3.10 (Residual). We know $\mathbf{A}$, b, $\widehat{x}$, and we want to solve $\mathbf{A}x = b$. So, the

$$\textit{Residual} = \mathbf{A}\widehat{x} - b$$
$$\textit{Residual Norm} = \|\mathbf{A}\widehat{x} - b\|$$
$$\textit{Relative Residual Norm} = \frac{\|\mathbf{A}\widehat{x} - b\|}{\|b\|}$$

Example 2.3.11

Let $\mathbf{A} = \begin{bmatrix} 0.835 & 0.667 \\ 0.333 & 0.266 \end{bmatrix}$, $b = \begin{bmatrix} 0.168 \\ 0.067 \end{bmatrix}$. Let $x = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$ be the exact solution to the system $\mathbf{A}x = b$ and $\widehat{x} = \begin{bmatrix} 267 \\ -334 \end{bmatrix}$, a bad computation of the solution. Then,

$$\frac{\|b - \mathbf{A}\widehat{x}\|_2}{\|x\|_2} \approx 0.006$$
$$\frac{\|x - \widehat{x}\|_2}{\|x\|_2} \approx \mathcal{O}(10^2)$$

Remark. 2.10 *The residual norm is not always a good estimate of the relative error.*

Definition 2.3.12 (Ill-Conditioned, Well-Conditioned). If the system is linearly dependent, we call the system *ill-conditioned.* If the system is linearly independent, we call it *well-conditioned.*

Definition 2.3.13 (Condition Numbers). The condition number of $\mathbf{A}$ is $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$. Note that $\kappa(\mathbf{A}) \geq 1$ and $\kappa(\mathbf{I}) = 1$. If $\kappa(\mathbf{A})$ is large, then $\mathbf{A}$ is ill-conditioned. If $\kappa(\mathbf{A})$ is close to 1, then $\mathbf{A}$ is well-conditioned.

Remark. 2.11 *Some intuition on $\kappa(\mathbf{A})$:*

- $\|\mathbf{A}\|$*: how much $\mathbf{A}$ moves x: $\mathbf{A}x = b$.*
- $\|\mathbf{A}^{-1}\|$*: how much $\mathbf{A}^{-1}$ moves b: $x = \mathbf{A}^{-1}b$.*

So, if $\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|$ is close to 1, the moves balance each other. If $\kappa(\mathbf{A})$ is large, then we move the vectors a lot.

Theorem 2.3.14 Upper Bound for Relative Error

$$\underbrace{\frac{\|x - \widehat{x}\|}{\|x\|}}_{\text{Relative Error}} \leq \kappa(\mathbf{A}) \cdot \underbrace{\frac{\|b - \mathbf{A}\widehat{x}\|}{\|b\|}}_{\text{Relative Residual}} = \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|b - \mathbf{A}\widehat{x}\|}{\|b\|}.$$

Proof 4. We want to use residual norm to compare $\|x - \widehat{x}\|$ and $\|x\|$. Suppose x is the true solution: $b = \mathbf{A}x$. Then,

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

So,

$$\frac{1}{\|x\|} \leq \frac{\|\mathbf{A}\|}{\|b\|} \tag{2}$$

Consider the residual: $r = b - \mathbf{A}\widehat{x} = \mathbf{A}x - \mathbf{A}\widehat{x} = \mathbf{A}(x - \widehat{x})$. So, $x - \widehat{x} = \mathbf{A}^{-1}r$. Therefore,

$$\|x - \widehat{x}\| = \|\mathbf{A}^{-1}r\| \leq \|\mathbf{A}^{-1}\|\|r\| \tag{3}$$

Putting Eq. (2) and Eq. (3) together, we have

$$\|x - \widehat{x}\| \cdot \frac{1}{\|x\|} \leq \|\mathbf{A}^{-1}\|\|r\| \cdot \frac{\|\mathbf{A}\|}{\|b\|}$$

Re-arrange the inequality, we have

$$\frac{\|x - \widehat{x}\|}{\|x\|} \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|b - \mathbf{A}\widehat{x}\|}{\|b\|}.$$

■

Remark. 2.12 *Since norms measure how far two things are apart from each other,* $\|x - \widehat{x}\| = \|\widehat{x} - x\|$ *and* $\|b - \mathbf{A}\widehat{x}\| = \|\mathbf{A}\widehat{x} - b\|$.

Corollary 2.3.15 If $\kappa(\mathbf{A}) \approx 1$, then a small residual implies that $\widehat{x}$ is a good approximation to the true solution x. If $\kappa(\mathbf{A})$ is large, then we still don't know if $\widehat{x}$ is a good approximation to the true solution.

Example 2.3.16

Given $\mathbf{A}_1 = \begin{bmatrix} 1 & 10 \\ 0 & 1 \end{bmatrix}$ and $\mathbf{A}_2 = \begin{bmatrix} 1 & 10^6 \\ 0 & 1 \end{bmatrix}$. Which matrix will have a better approximation to the true solution?

Solution 5.

$$\kappa_1(\mathbf{A}_1) = \|\mathbf{A}_1\|_1\|\mathbf{A}_1^{-1}\|_1.$$

Since $\det(\mathbf{A}_1) = 1 - 0 = 1$, we know $\mathbf{A}_1^{-1} = \dfrac{1}{\det(\mathbf{A}_1)}\begin{bmatrix} 1 & -10 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & -10 \\ 0 & 1 \end{bmatrix}$. So,

$$\kappa_1(\mathbf{A}_1) = \|\mathbf{A}_1\|_1\|\mathbf{A}_1^{-1}\|_1 = (11)(11) = 121.$$

Similarly,

$$\kappa_2(\mathbf{A}_2) = \|\mathbf{A}_2\|_1\|\mathbf{A}_2^{-1}\|_1 = \left(1 + 10^6\right)\left(1 + 10^6\right) = \mathcal{O}(10^12).$$

Since $\kappa_2(\mathbf{A}_2) \gg \kappa_1(\mathbf{A}_1)$, $\mathbf{A}_1$ will yield a more accurate approximation. □

Remark. 2.13 *Think* $\kappa(\mathbf{A})$ *as an indicator for how much movement of* x *will there be if we apply* $\mathbf{A}$ *on* x.

Claim. 2.1 *Conditioning is inherent to the problem. So, no algorithms can improve conditioning.*

Definition 2.3.17 (Algorithm Stability/Backward Stability). When we solve $\mathbf{A}x = b$, we will have some algorithm $\widehat{x} = \text{algorithm}(\mathbf{A}, b)$. Imagine we run the algorithm in reverse (backwards). We should obtain $\widehat{\mathbf{A}}$ and $\widehat{b}$ *s.t.* $\widehat{\mathbf{A}}\widehat{x} = \widehat{b}$ in exact arithmetic. An algorithm is *backward stable* if $\|\mathbf{A} - \widehat{\mathbf{A}}\|$ and $\|b - \widehat{b}\|$ are small.

Remark. 2.14 *Algorithm stability has nothing to do with conditioning.*

Example 2.3.18

Given $\mathbf{A} = \begin{bmatrix} \alpha & 1 \\ 1 & 2 \end{bmatrix}$. We know that solving $\mathbf{A}x = b$ using Gaussian Elimination without pivoting, we could get a solution far from true. So, Gaussian Elimination without pivoting is not backward stable. In contrast, GEPP is a backward stable algorithm.

Example 2.3.19 Is Multiplication Backward Stable?

Define

$$x \otimes y = \texttt{float}(\texttt{float}(x) \times \texttt{float}(y))$$

on a computer with 3 significant digits. Suppose $x = \dfrac{1}{3}$ and $y = \dfrac{1}{2}$. Then,

$$x \otimes y = \texttt{float}((0.333)(0.500) = 0.167; \quad \varepsilon_{\text{WP}} = 1.01 - 1.00 = 10^{-2}$$

Take $\widehat{x} = 0.334$ and $\widehat{y} = 0.500$, we get $\widehat{x}\widehat{y} = 0.167$. Since $\|x - \widehat{x}\| \approx 0.001 \leq \dfrac{1}{2}\varepsilon_{\text{WP}}$, we say multiplication is backward stable. Similarly, we could show all floating point operations are backward stable, in fact.

Example 2.3.20

Prove that $\|\mathbf{Q}x\|_2 = \|x\|_2$ for any $x \in \mathbb{R}^n$ if $\mathbf{Q}$ is orthogonal.

Proof 6. Since $\mathbf{Q}$ is orthogonal, $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$. Note that the $2-$norm:

$$\|x\|_2^2 = x_1^2 + x_2^2 + \cdots + x_n^2 = x^T x.$$

Then,

$$\|\mathbf{Q}x\|_2^2 = (\mathbf{Q}x)^T(\mathbf{Q}x) = x^T\mathbf{Q}^T\mathbf{Q}x = x^T x = \|x\|_2^2.$$

So,

$$\|\mathbf{Q}x\|_2 = \|x\|_2.$$

■

Extension. 2.1 *What is* $\|\mathbf{Q}\|_2$*? What is* $\kappa(\mathbf{Q})$*?*

Solution 7.

$$\|\mathbf{Q}\|_2 = \max_{\|x\|_2=1} \frac{\|\mathbf{Q}x\|_2}{\|x\|_2} = \max_{\|x\|_2=1} \frac{\|x\|_2}{\|x\|_2} = \max_{\|x\|_2=1} 1 = 1.$$

$$\kappa(\mathbf{Q}) = \|\mathbf{Q}\|_2\|\mathbf{Q}^{-1}\|_2 = \|\mathbf{Q}\|_2\|\mathbf{Q}^T\|_2 = 1 \cdot 1 = 1.$$

□

3 Curve Fitting

3.1 Polynomial Interpolation

Definition 3.1.1 (Interpolation). A function $p(x)$ interpolates data $\{(x_i, f_i)\}_{i=0}^N$ if $p(x_i) = f_i$ for $i = 0, \ldots, N$.

Remark. 3.1 *Uniqueness of the interpolating polynomial.*

Definition 3.1.2 (Polynomial). A polynomial $p_k(x)$ is of *degree* k if there are constants $c_0, \ldots, c_k$ *s.t.*

$$p_k(x) = c_0 + c_1 x + \cdots + c_k x^k.$$

A polynomial $p_k(x)$ is in *exact degree* k if $c_k \neq 0$.

Theorem 3.1.3 Steps for Polynomial Interpolation

1. Create a problem.
   - Find some data.
   - Design the problem
2. Choose a degree (based on the number of interpolation points)
3. Determine the coefficients. $\rightarrow$ Construct a polynomial
   - Choose a polynomial basis
   - Solve a linear system.
4. Draw the curve: evaluating at lots of points. $\rightarrow$ Evaluate a polynomial

---

Algorithm 4: Constructing a Polynomial Interpolant

Input: data, $\{(x_i, f_i)\}_{i=0}^N$; polynomial basis: $\{q_j(x)\}_{i=0}^N$

Output: $c_0, \ldots, c_M$ *s.t.* $\sum_{j=0}^{M} c_j q_j(x_i) = f_i$ for $i = 0, \ldots, N$

1 Solve for $c_0, \ldots, c_M$:

$$\begin{cases} c_0 q_0(x_0) + c_1 q_1(x_0) + \cdots + c_M q_M(x_0) = f_0 \\ \vdots \\ c_0 q_0(x_N) + c_1 q_1(x_N) + \cdots + c_M q_M(x_N) = f_0 \end{cases}$$

$$\begin{bmatrix} q_0(x_0) & \cdots & q_M(x_0) \\ \vdots & \ddots & \vdots \\ q_0(x_N) & \cdots & q_M(x_N) \end{bmatrix} \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = \begin{bmatrix} f_0 \\ \vdots \\ f_N \end{bmatrix}$$

Theorem 3.1.4 Polynomial Uniqueness
When the nodes $\{(x_i, f_i)\}_{i=0}^N$ are distinct, there is a unique polynomial, the interpolating polynomial $p_N(x)$ of degree N that interpolates the data. That is,

$$\begin{bmatrix} q_0(x_0) & \cdots & q_M(x_0) \\ \vdots & \ddots & \vdots \\ q_0(x_N) & \cdots & q_M(x_N) \end{bmatrix} \in \mathbb{R}^{n\times n}$$

Definition 3.1.5 (Power Series).

$$p_N(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_N x^N$$

The Vandermonde Matrix is defined as

$$\begin{bmatrix} 1 & x_0 & \cdots & x_0^N \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & \cdots & x_N^N \end{bmatrix}.$$

- Pros: easy to understand and implement
- Cons: ill-conditioning, near singularity of the Vandermonde matrix, when $|x_i - x_j|$ is small.

Example 3.1.6 Power Series
Interpolate the points $\{(-1, 0), (0, 1), (1, 3)\}$.
Solution 1.
Suppose $p_2(x) = c_0 + c_1 x + c_2 x^2$. Then,

$$\begin{aligned} (-1, 0) :\ & p_2(-1) = c_0 + c_1(-1) + c_2(-1)^2 = 0 \\ (0, 1) :\ & p_2(0) = c_0 + c_1(0) + c_2(0^2) = 1 \\ (1, 3) :\ & p_2(1) = c_0 + c_1(1) + c_2(1)^2 = 3. \end{aligned}$$

In matrix form:

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \implies \begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3/2 \\ 1/2 \end{bmatrix}.$$

□

Definition 3.1.7 (Newton Form).

$$p_N(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1) + \cdots + b_N(x - x_0)(x - x_1)\cdots(x - x_{N-1}).$$

- Pros: we are having a lower triangular system:

$$\begin{aligned} p_N(x_0) &= b_0 \\ p_N(x_1) &= b_0 + b_1(x - x_0) \\ p_N(x_2) &= b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1). \end{aligned}$$

Example 3.1.8 Newton Form

Interpolate the points $\{(-1, 0), (0, 1), (1, 3)\}$.

Solution 2.

$$\begin{aligned} p_N(x) &= b_0 + b_1(x - (-1)) + b_2(x - (-1))(x - 0) \\ p_N(-1) &= b_0 \\ p_N(0) &= b_0 + b_1 \\ p_N(1) &= b_1 + 2b_1 + 2b_2 \end{aligned}$$

In matrix form:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \xrightarrow[\text{Substitution}]{\text{Forward}} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1/2 \end{bmatrix}.$$

□

Definition 3.1.9 (Lagrange Polynomials).

- Coefficient = function values f_i. $\implies$ we don't need to solve anything
-

$$\ell(x) : \begin{cases} 1 & \text{at } x = x_i \\ 0 & \text{at} x = x_j, \ j \neq i \end{cases}$$

Now, we want to construct a polynomial that has roots at all the nodes:

$$\omega(x) = (x - x_0)(x - x_1) \cdots (x - x_N).$$

Then, if we assume we have distinct noes, we have

$$\ell_0(x) = \frac{(x - x_1)(x - x_2) \cdots (x - x_N)}{(x_0 - x_1)(x_0 - x_2) \cdots (x_0 - x_N)}$$

$$\ell_3(x) = \frac{(x - x_0)(x - x_1)(x - x_2)(x - x_4) \cdots (x - x_N)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)(x_3 - x_4) \cdots (x_3 - x_N)}$$

Generalizing, we have

$$\ell_k(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_N)}{\text{numerator evaluated at } x = x_k} = \prod_{j=0,\ j\neq k}^{N} = \frac{x-x_j}{x_k-x_j}.$$

- $p_N(x) = \sum_{i=0}^{N} f_i \cdot \ell_i(x)$.
- Pros: No solving. Great for theory.
- Cons: constructing $\ell_i(x)$ is tricky.

Example 3.1.10 Lagrange Polynomials

Interpolate the points $\{(-1,0),(0,1),(1,3)\}$.

Solution 3.

$$\ell_0(x) = \frac{(x-0)(x-1)}{(-1-0)(-1-1)}; \quad \ell_1(x) = \frac{(x-(-1))(x-1)}{(0-(-1))(0-1)}; \quad \ell_2(x) = \frac{(x-(-1))(x-0)}{(1-(-1))(1-0)}$$

So,

$$p_N(x) = f_0\ell_0(x) + f_1\ell_1(x) + f_2\ell_2(x).$$

□

Definition 3.1.11 (Chebyshev Polynomial).

- Only works for domain $[-1,1]$, but easy to extend to $[a,b]$.
- $T_j(x) = \cos(j \cdot \arccos(x)), \quad j = 0,\ldots,N$. Therefore, $T_0(x) = \cos(0 \cdot \arccos(x)) = 1$ and $T_1 = \cos(1 \cdot \arccos(x)) = x$. By trigonometric identities, we can show

$$T_{j+1} = 2xT_j(x) - T_{j-1}(x).$$

Remark. 3.2 *Domain is limited to* $[-1,1]$ *because* $\arccos(x)$ *can only take* $x \in [-1,1]$.

- We solve for $d_0,\ldots,d_N$, where

$$p_N(x) = d_0T_0(x) + d_1T_1(x) + d_2T_2(x) + \cdots + d_NT_N(x)$$

- Pros: Nice numerical properties due to oscillation.
- Cons: Highly non-intuitive.

Example 3.1.12 Chebyshev Polynomial

Interpolate the points $\{(-1,0),(0,1),(1,3)\}$.

Solution 4.

Assume $p_2(x) = d_0T_0 + d_1T_1(x) + d_2T_2(x)$, where $T_0(x) = 1$, $T_1(x) = x$, $T_2(x) = 2x^2 - 1$. Then,

$$\begin{aligned} p_2(-1) &= d_0 + d_1(-1) + d_2(1) \\ p_2(0) &= d_0 + d_1(0) + d_2(-1) \\ p_2(1) &= d_0 + d_!(1) + d_2(1) \end{aligned}$$

In matrix form, we have

$$\begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 3 \end{bmatrix} \implies \begin{bmatrix} d_0 \\ d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} 5/4 \\ 3/2 \\ 1/4 \end{bmatrix}$$

□

3.2 Error in Polynomial Interpolation

Remark. 3.3 *Set-up:*

- $f(x)$, *the unknown function*
- $\{(x_i, f_i)\}_{i=0}^N$, *data, where* $f_i = f(x_i)$.
- $p_N(x)$, *degree* N *interpolant*
- *Our goal is to find* error $= f(x) - p_N(x)$. *Note that at* $x = x_i$, $f(x_i) - p_N(x_i) = 0$ *for* $i = 1, \ldots, N$. *So,* x_i *are roots of* $f(x_i) - p_N(x_i)$. *Recall that* x_i *are also roots of* $\omega(x) = (x - x_0)(x - x_1) \cdots (x - x_N)$. *We wonder if we can build a connection between them.*

Theorem 3.2.1

$\exists\, \xi_x$, a point on $[a, b]$ depends on x such that

$$f(x) - p_N(x) = \frac{\omega(x)}{(N+1)!} f^{(N+1)}(\xi_x).$$

Example 3.2.2

Suppose $f(x) = x^2$ on $[0, 1]$. We have data $\{(0,0),(1,1)\}$ and interpolant $p_1(x) = x$.

Then,

$$f(x) - p_1(x) = x^2 - x$$

and

$$\omega(x) = (x-0)(x-1) = x^2 - x.$$

Therefore,

$$\text{LHS} = \frac{\omega(x)}{(N+1)!} f^{(N+1)}(\xi_x) = \frac{x^2 - x}{(1+1)!} f''(\xi_x) = \frac{x^2 - x}{2!}(2) = x^2 - x = \text{RHS}$$

Example 3.2.3

Suppose $f(x) = x^3$ on $[0,1]$. We have data $\{(0,0),(1,1)\}$ and interpolant $p_1(x) = x$. Then,

$$f(x) - p_1(x) = x^3 - x$$

and

$$\omega(x) = (x-0)(x-1) = x^2 - x.$$

Therefore,

$$\text{RHS} = \frac{\omega(x)}{(N+1)!} f^{(N+1)}(\xi_x) = \frac{x^2 - x}{(1+1)!} f''(\xi_x) = \frac{x^2 - x}{2}(6\xi_x) = x^2 - x.$$

Question: Can we find a ξ_x where $\frac{x^2 - x}{2}(6\xi_x) = x^2 - x$.

Answer: Say we want to evaluate error at $x = \frac{1}{2}$, we can certainly find some $\xi_x = \xi_{\frac{1}{2}}$ such that

$$\left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right) = \frac{(1/2)^2 - (1/2)}{2}(6\xi_x).$$

Remark. 3.4 *Some intuition on why this equation holds.*

- *When is $f(x) - p_N(x) = 0$? At $x_0, \ldots, x_N$ due to interpolation.*
- *When is $\omega(x) = 0$? At $x_0, \ldots, x_N$ due to construction.*
- *So, $f(x) - p_N(x)$ has roots $x_0, \ldots, x_N$, which means we can "factor" out the roost using $\omega(x)$: $f(x) - p_N(x) = \omega(x)g(x)$.*
- *What is ξ_x? It is found through the Rolle's Theorem. a special case for the Mean Value Theorem: If $f(a) = f(b) = 0$, then $\exists \xi \in [a,b]$* s.t. *$f'(\xi) = 0$.*

Theorem 3.2.4 Make Error Small

$$\max_{x\in[a,b]} |f(x) - p_N(x)| \leq \underbrace{\max_{x\in[a,b]} |\omega(x)|}_{\text{make this small!}} \cdot \frac{\overbrace{\max_{z\in[a,b]} \left|f^{(N+1)}(z)\right|}^{\text{typically unknown}}}{(N+1)!}$$

Theorem 3.2.5 Make $|\omega(x)|$ small

$$\omega(x) = (x - x_0)(x - x_1)\cdots(x - x_N)$$

Choose x_i so that usually x is "close" to some x_i: Chebyshev points, roots of Chebyshev polynomials:

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \quad i = 1, \dots, n$$

Theorem 3.2.6

$$|\omega(x)| \leq |b - a|^{N+1}$$

Remark. 3.5 *This inequality is true because $\omega(x)$ can be regarded as distances from x to x_i's. Hence, another approach to make $|\omega(x)|$ smaller is to make the interval $[a, b]$ smaller.*

Definition 3.2.7 (Linear Splines).

- Data: $\{(x_i, f_i)\}_{i=1}^N$
- Ordering: $a = x_0 < x_1 < x_2 < \cdots < x_N = b$

  Remark. 3.6 *We do not require equally spaced points here.*

- Linear Splines:

$$S_{1,N}(x) = \begin{cases} f_0 \cdot \dfrac{x - x_1}{x_0 - x_1} + f_1 \cdot \dfrac{x - x_0}{x_1 - x_0}, & x \in [x_0, x_1] \\ f_1 \cdot \dfrac{x - x_2}{x_1 - x_2} + f_2 \cdot \dfrac{x - x_1}{x_2 - x_1}, & x \in [x_1, x_2] \\ \vdots & \\ f_{N-1} \cdot \dfrac{x - x_N}{x_{N-1} - x_N} + f_N \cdot \dfrac{x - x_{N-1}}{x_N - x_{N-1}}, & x \in [x_{N-1}, x_N] \end{cases},$$

where 1 indicates the degree of each piece (1 for linear) and N is the number of intervals ($N + 1$ points create N intervals)

Example 3.2.8

Given $\{(-1,0),(0,1),(1,3)\}$. Construct $S_{1,2}(x)$.

Solution 1.

$$S_{1,2} = \begin{cases} 0 \cdot \dfrac{x-0}{(-1)-0} + 1 \cdot \dfrac{x-(-1)}{0-(-1)} = x+1 & x \in [-1,0] \\ 1 \cdot \dfrac{x-1}{0-1} + 3 \cdot \dfrac{x-0}{1-0} = 1-x+3x = 2x+1 & x \in [0,1] \end{cases}.$$

□

Definition 3.2.9 (Linear B-Splines). We define the linear B-splines basis as follows:

$$L_0(x) = \begin{cases} \dfrac{x-x_1}{x_0-x_1} & x \in [x_0,x_1] \\ 0 & o/w \text{ (otherwise)} \end{cases}$$

$$L_i(x) = \begin{cases} \dfrac{x-x_{i-1}}{x_i-x_{i-1}} & x \in [x_{i-1},x_i] \text{ (left of } x_i) \\ \dfrac{x-x_{i+1}}{x_i-x_{i+1}} & x \in [x_i,x_{i+1}] \text{ (right of } x_i) \\ 0, & o/w \end{cases}, \quad i = 1,\ldots,N-1.$$

$$L_N(x) = \begin{cases} \dfrac{x-x_{N-1}}{x_N-x_{N-1}} & x \in [x_{N-1},x_N] \\ 0 & o/w \end{cases}$$

Therefore, we can write the linear splines as

$$S_{1,N}(x) = f_0L_0(x) + f_1L_1(x) + \cdots + f_NL_N(x).$$

Definition 3.2.10 (Cubic Splines).

$$S_{3,N}(x) = \begin{cases} p_1(x) = a_{1,0} + a_{1,1}x + a_{1,2}x^2 + a_{1,3}x^3 & x \in [x_0,x_1] \\ p_2(x) = a_{2,0} + a_{2,1}x + a_{2,2}x^2 + a_{2,3}x^3 & x \in [x_1,x_2] \\ \vdots & \\ p_N(x) = a_{N,0} + a_{N,1}x + a_{N,2}x^2 + a_{N,3}x^3 & x \in [x_{N-1},x_N] \end{cases},$$

where $a_{i,j}$ are unknowns. We have in total $4N$ unknowns, so we need $4N$ equations to solve them. We want the splines to interpolate and continuous. So, for $i = 1,\ldots,N$ we have

$$p_i(x_{i-1}) = f_{i-1} \quad \text{and} \quad p_i(x_i) = f_i,$$

which gives us in total $2N$ equations. To ensure continuous, we want the derivatives to be continuous, so we have

$$p_i'(x_i) = p_{i+1}'(x_i) \quad \text{for } i = 1, \ldots, N-1,$$

which gives us another $N-1$ equations. To further ensure continuous and smoothy, we need continuous second derivatives, so we have

$$p_i''(x_i) = p_{i+1}''(x_i) \quad \text{for } i = 1, \ldots, N-1,$$

which also gives us $N-1$ equations. Now, we have in total $4N-2$ equations, and we cannot take the 3^{rd} derivatives because after taking the 3^{rd} derivatives, there will be only constants left. Hence, we still need 2 more equations, and there are several different approaches:

- Natural boundary conditions:

$$p_1''(x_0) = 0; \qquad p_N''(x_N) = 0.$$

- Second derivative conditions:

$$p_1''(x_0) = f''(x_0); \qquad p_N''(x_N) = f''(x_N)$$

  This condition is not always helpful because we don't always know the information on the original function.

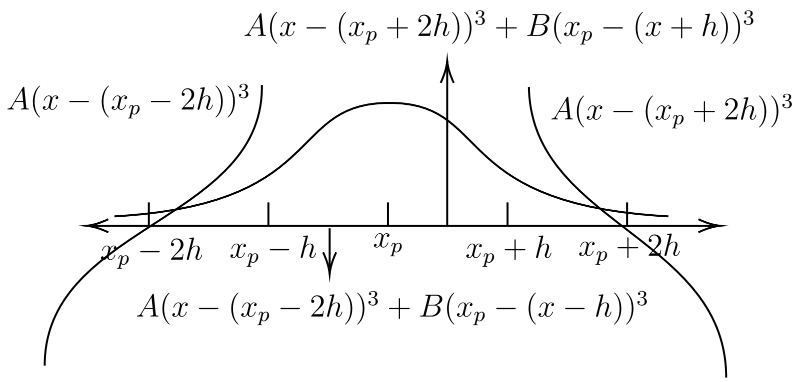
- First derivative conditions:

$$p_1'(x_0) = f'(x_0); \qquad p_N'(x_N) = f'(x_N).$$

- Not-a-knot condition:

$$p_1'''(x_1) = p_2'''(x_1); \qquad p_{N-1}'''(x_{N-1}) = p_N'''(x_{N-1})$$

Definition 3.2.11 (Cubic B-Splines). Assume we have equally spaced points x_0, $x_1 = x_0 + h$, $x_2 = x_1 + h, \cdots$. We want to find $B_p(x)$ center at x_p:



We require $B_p(x_p) = 1$, so we have

$$A(x - (x_p - 2h))^3 + B(x - (x_p - h))^3 = 1.$$

We require continuity, so

$$A(x - (x_p - 2h))^3 B(x - (x - h))^3 = A(x - (x_p + 2h))^3 B(x - (x + h))^3.$$

Solving the system, we will have

$$B_p(x) = \begin{cases} 0 & x < x_p - 2h \\ \frac{1}{4h^3}(x - (x_p - 2h))^3 & x_p - 2h \leq x < x_p - h \\ \frac{1}{4h^3}(x - (x_p - 2h))^3 - \frac{1}{h^3}(x - (x_p - h))^3 & x_p - h \leq x < x_p \\ -\frac{1}{4h^3}(x - (x_p + 2h))^3 + \frac{1}{h^3}(x - (x_p + h))^3 & x_p \leq x < x_p + h \\ -\frac{1}{4h^3}(x - (x_p + 2h))^3 & x_p + h \leq x + x_p + 2h \\ 0 & x_p + 2h \leq x \end{cases}$$

Then, we have

$$S_N(x) = \sum_{i=-1}^{N+1} a_i \cdot B_i(x),$$

where

$$B_p(x_{p-1}) = \frac{1}{4}; \quad B_p(x_p) = 1; \quad B_p(x_{p+1}) = \frac{1}{4}$$

3.3 Least Square

Definition 3.3.1 (Least Square Polynomial). Let $q_M(x)$ be a polynomial of degree M. We want to measure the error between $q_M(x)$ and $f(x)$ with data $\{(x_i, f_i)\}_{i=0}^N$. We define

$$\sigma(q_M(x)) \equiv \sum_{r=0}^{N} \{q_M(x_r) - f_r\}^2$$

The *least square polynomial* of degree M, $p_M(x)$, satisfies

$$\sigma(p_M) \leq \sigma(q_M)$$

for any other degree M polynomial $q_M(x)$.

Example 3.3.2 Find the Best Fit Line

i	x_i	f_i
0	1	2
1	3	4
2	4	3
3	5	1

Solution 1.
Suppose $p_1(x) = a_0 + a_1 x$. Then,

$$\sigma(p_1) = \sum_{r=0}^{3} \{a_0 + a_1 x_r - f_r\}^2$$

To minimize $\sigma(p_1)$, we set $\dfrac{\partial \sigma}{\partial a_0} = \dfrac{\partial \sigma}{\partial a_0} = 0$:

$$\frac{\partial \sigma(p_1)}{\partial a_0} = \sum_{r=0}^{3} 2(a_0 + a_1 x_r - f_r) = 0$$
$$\frac{\partial \sigma(p_1)}{\partial a_0} = \sum_{r=0}^{2} 2x_r(a_0 + a_1 x_r - f_r) = 0$$

Plugging in data, we have

$$\frac{\partial \sigma(p_1)}{\partial a_0} = 2(4a_0 + 13a_1 - 10) = 0; \quad \frac{\partial \sigma(p_1)}{\partial a_0} = 2(13a_0 + 51a_1 - 31) = 0$$

Solve the following system:

$$\begin{bmatrix} 4 & 13 \\ 13 & 51 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 10 \\ 31 \end{bmatrix} \implies \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 107/35 \\ -6/35 \end{bmatrix}.$$

□

Definition 3.3.3 (Best Fit Polynomial).

$$q_M(x) = a_0\varphi_o(x) + a_1\varphi_1(x) + \cdots + a_M\varphi_M(x).$$

We want $q_M(x)$ to fit data $\{(x_i, f_i)\}_{i=0}^{N}$ as best as we can, so we compute the error

$$\sigma(q_M) = \sum_{r=0}^{N} \{q_M(x_r) - f_r\}^2 = \sum_{r=0}^{N} \{a_0\varphi_o(x_r) + a_1\varphi_1(x_r) + \cdots + a_M\varphi_M(x_r) - f_r\}^2.$$

In matrix form (vector 2-norm), we have

$$\sigma(q_M) = \|\mathbf{V}a - f\|_2^2.$$

That is,

$$\sigma(q_M) = \left\| \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \cdots & \varphi_M(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \cdots & \varphi_M(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_N) & \varphi_1(x_N) & \cdots & \varphi_M(x_N) \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_M \end{bmatrix} - \begin{bmatrix} f_0 \\ f_1 \\ \vdots \\ f_N \end{bmatrix} \right\|_2^2$$

To find a p_M minimizes $\|\mathbf{V}a - f\|_2^2$, we solve the normal equations, that is

$$\mathbf{V}^T\mathbf{V}a = \mathbf{V}^T f.$$

Solving the normal equation is equivalent to solving the least square problems.

Example 3.3.4 Basis

For $\varphi_i(x) = x^j$, the power series, the big matrix $\mathbf{V}$ becomes

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^M \\ 1 & x_1 & x_1^2 & \cdots & x_1^M \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \cdots & x_N^M \end{bmatrix}$$

This is consistent with the Vandermonde matrix. The matrix might not necessarily be a squared matrix.

Example 3.3.5 Solving the Normal Equation

Find the best fit equation $p_2(x) = a_1 + a_1 x + a_2 x^2$.

i	x_i	f_i
0	-2	6
1	-1	3
2	0	1
3	1	3
4	2	6

Solution 2.

Construct $\mathbf{V}$ (power series):

$$\mathbf{V} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}; \qquad f = \begin{bmatrix} 6 \\ 3 \\ 1 \\ 3 \\ 6 \end{bmatrix}$$

Construct $\mathbf{V}^T\mathbf{V}$ and $\mathbf{V}^T f$:

$$\mathbf{V}^T\mathbf{V} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 10 & -18 \\ 10 & -18 & 34 \end{bmatrix}; \qquad \mathbf{V}^T f = \begin{bmatrix} 19 \\ 0 \\ 54 \end{bmatrix}$$

Remark. 3.7 *In most cases of this class,* $\mathbf{V}^T\mathbf{V}$ *will be symmetric, positive, definite (having positive eigenvalues), so* $\mathbf{V}^T\mathbf{V}$ *is invertible*

Solve $\mathbf{V}^T\mathbf{V}a = \mathbf{V}^T f$: `MATLAB: (V'*V)\(V'*f); % worse conditioning`

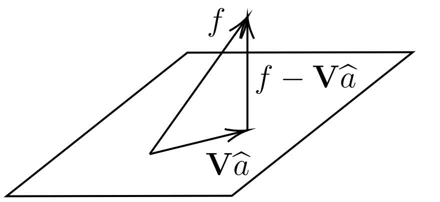
$$a \approx \begin{bmatrix} 1.5143 \\ 0 \\ 1.1429 \end{bmatrix}.$$

□

Remark. 3.8 *Important Points.*

- $\mathbf{V}$ *is an* $(N+1) \times (M+1)$ *matrix, and* $M < N$.
- *If* $M < N$ *and* x_i *are distinct,* $\mathbf{V}^T\mathbf{V}$ *will be SPD, and* p_M *is unique.*
- *If* $M = N$, *we are back to polynomial interpolation.*
- *In practice, least square polynomial is more useful than interpolation for data analysis.*

Remark. 3.9 *Why does the normal equation work? - Linear Algebra.*



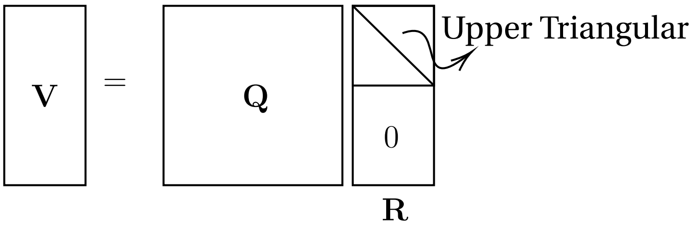
$\mathrm{col}(\mathbf{V}) = \{\mathbf{V}y \mid y \in \mathbb{R}^{m+1}\}$

Our goal: $\min\limits_a \|\mathbf{V}a - f\|_2^2$. *We want a nontrivial* $\widehat{a}$ *such that*

$$\begin{aligned} (\mathbf{V}\widehat{a})^T - (f - \mathbf{V}\widehat{a}) &= 0 \\ \widehat{a}^T\big(\mathbf{V}^T(f - \mathbf{V}\widehat{a})\big) &= 0 \\ \mathbf{V}^T f - \mathbf{V}^T\mathbf{V}\widehat{a} &= 0 \qquad\qquad \textit{normal equation} \end{aligned}$$

Definition 3.3.6 (Solving Least Squares with Matrix Factorization). We will use QR factorization.

- $\mathbf{V} = \mathbf{QR}$, where Q is orthogonal and $\mathbf{R}$ is upper triangular.



- Substitute:

$$\begin{aligned}
\|\mathbf{V}a - f\|_2^2 &= \|\mathbf{QR}a - f\|_2^2 \\
&= \|\underbrace{\mathbf{Q}^T(\mathbf{Q}}_{\mathbf{I}}\mathbf{R}a - f)\|_2^2 \qquad 2 - \textit{norms invariant to orthogonal matrices} \\
&= \|\mathbf{R}a - \mathbf{Q}^T f\|_2^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_M \\ \mathbf{0} \end{bmatrix} a - \begin{bmatrix} b \\ c \end{bmatrix} \right\|_2^2 \qquad \textit{parition } \mathbf{Q}^T f \textit{ into } 2 \textit{ parts} \\
&= \|\mathbf{R}_M a - b\|_2^2 - \|c\|_2^2.
\end{aligned}$$

- Solve $\mathbf{R}_M a = b$ using backward substitution, where b is the first M row of $\mathbf{Q}^T f$.

4 Differentiation and Integration

4.1 Review - Taylor Series

Definition 4.1.1 (Taylor Series). Let $f(x)$ be a smooth function (*we can take derivatives*). We can approximate $f(x)$ about $x = a$ using Taylor series:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \mathcal{O}\big(|x-a|^3\big),$$

where $\mathcal{O}\big(|x-a|^3\big)$ denotes higher order terms that depend on $|x-a|^3$ or higher orders.
Definition 4.1.2 (Taylor Series). To approximate $f(x+h)$ for $h > 0$ about x, we have

$$f(x+h) = f(x) + f'(x) \underbrace{h}_{(x+h-x)} + \frac{1}{2}f''(x)h^2 + \mathcal{O}(h^2).$$

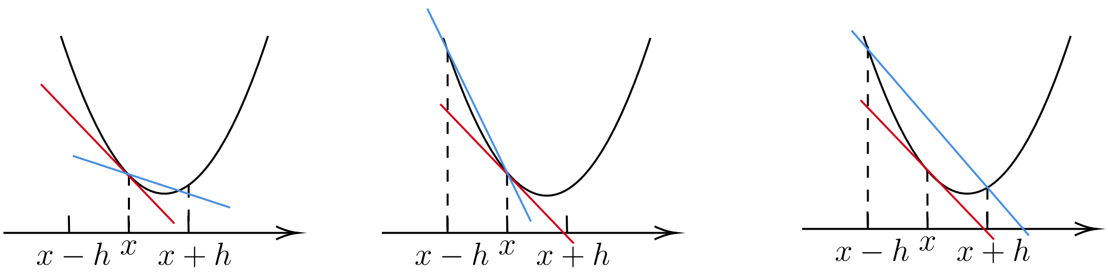
4.2 Differentiation

Definition 4.2.1 (Differentiation).

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

Definition 4.2.2 (Finite Difference Approxiation).

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



Forward Difference Backward Difference Central Difference

Example 4.2.3 Finite Difference

Suppose $f(x) = x^4$. We want to approximate $f'(1)$. By hand, we know $f'(x) = 4x^3$, so $f'(1) = 4$. However, numerically, we have the first order approximation ($\mathcal{O}(h)$) is

$$f'(1) \approx \frac{f(1+h)-f(1)}{h}$$

and the second order approximation ($\mathcal{O}(h^2)$) is

$$f'(1) \approx \frac{f(1+h)-f(1-h)}{2h}$$

Note that in the first order approximation, every time we divide h by 10, we are 10 times more accurate. In the second order approximation, every time we divide h by 10, we will be 100 times more accurate.

Remark. 4.1 *Why Central Differencing is in Second Order?*
By Taylor series, we know

$$f(x+h) = f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \mathcal{O}(h^3)$$
$$f(x-h) = f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 + \mathcal{O}(h^3).$$

Then,

$$\begin{aligned} f(x+h) - f(x-h) &= f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + \mathcal{O}(h^3) \\ &\quad - \left[f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 + \mathcal{O}(h^3)\right] \\ &= 2f'(x)h + \mathcal{O}(h^3) \\ f'(x) &= \frac{f(x+h) - f(x-h)}{2h} + \mathcal{O}(h^2) && \textit{Divide by } h \\ &\approx \frac{f(x+h) - f(x-h)}{2h} \end{aligned}$$

4.3 Integration

Definition 4.3.1 (Definite Integral). Suppose f is a function defined over the interval $[a, b]$. Then, the functional $I(x)$ is the area under the curve, and

$$I(f) = \int_a^b f(x)\,\mathrm{d}x.$$

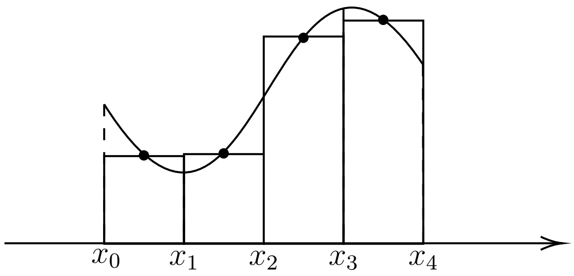
Definition 4.3.2 (Functional). A functional is a mapping from functions to real numbers.

Theorem 4.3.3 Steps to approximate $I(f)$

- Partition the Interval: $a \leq x_0 < x_1 < x_2 < \cdots < x_N \leq b$.
- Evaluate f at points x_i, we get $f(x_i)$ and the data $\{(x_i, f_i)\}_{i=0}^N$.
- Integrate using Riemanian Sums

Definition 4.3.4 (Midpoint Rule).

$$\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x \approx \int_{x_{i-1}}^{x_i} f\left(\frac{x_{i-1}+x_i}{2}\right)\mathrm{d}x = (x_i - x_{i-1})f\left(\frac{x_{i-1}-x_i}{2}\right).$$



Definition 4.3.5 (Composite Midpoint Rule).

$$
\begin{aligned}
I(x) = \int_a^b f(x)\,\mathrm{d}x &= \int_{x_0}^{x_1} f(x)\,\mathrm{d}x + \int_{x_1}^{x_2} f(x)\,\mathrm{d}x + \cdots + \int_{x_{N-1}}^{x_N} f(x)\,\mathrm{d}x \\
&\approx (x_1 - x_0) f\left(\frac{x_0 + x_1}{2}\right) + (x_2 - x_1) f\left(\frac{x_1 + x_2}{2}\right) + \cdots + (x_N - x_{N-1}) f\left(\frac{x_{N-1} + x_N}{2}\right)
\end{aligned}
$$

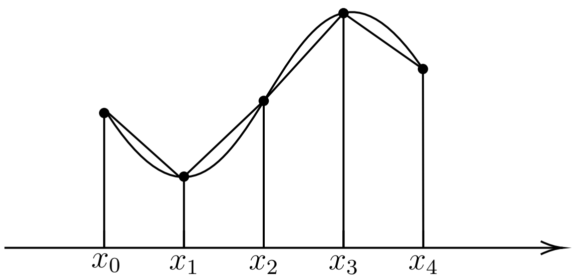
If we have equally spaced points: $x_i = a + ih$ for $i = 0, \ldots, N$, and $h = \dfrac{b-a}{N}$, then

$$
I(x) = h \cdot \sum_{i=1}^{N} f\left(x_{\frac{i-1}{2}}\right) \equiv R_{CM}(f, h).
$$

If $h \to 0$, the approximation gets better (in theory). In practice, numerical considerations come into play.

Definition 4.3.6 (Trapezoidal Rule).

$$
\int_{x_{i-1}}^{x_i} f(x)\,\mathrm{d}x \approx (x_i - x_{i-1}) \left(\frac{f(x_{i-1}) + f(x_i)}{2} \right).
$$



Definition 4.3.7 (Composite Trapezoidal Rule).

$$R_{CT}(f,h) \equiv h \sum_{i=1}^{N} \left(\frac{f(x_{i-1}) + f(x_i))}{2} \right)$$

Definition 4.3.8 (Quadrature Rule). The Quadrature question concerns "Can we find a square that has the same area as a circle?" In the integration problem, the Quadrature Rules states that given points $x_0 < x_1 < \cdots < x_N$ and weights $w_0, w_1, \ldots, w_N$, then

$$R(f) \equiv \sum_{i=0}^{N} w_i f(x_i).$$

Theorem 4.3.9 Properties of $R(f)$

- Linear functional:

$$R(\alpha f + \beta g) = \alpha R(f) + \beta R(g).$$

- Alternative perspective:

$$R(f) = I(q),$$

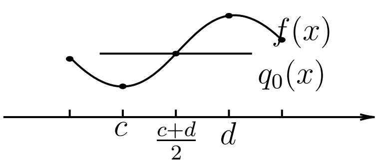
  where q is an approximation (q has a polynomial interpolant) to f.

$$\begin{aligned} I(q) &= I(f + (q - f)) \\ &= I(f) + \underbrace{I(q-f)}_{\text{error of integration}} \end{aligned}$$

  Then, the error of integration depends on how well q approximates f.

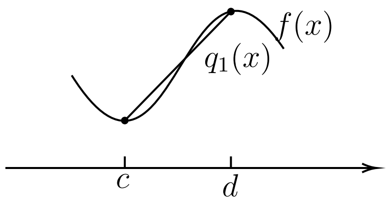
Example 4.3.10 Midpoint Rule Revisit

Use degree 0 approximation of $f(x)$, $q_0(x)$, and use $x = \dfrac{c+d}{2}$ on $[c, d]$. Then, we have exactly the midpoint rule:



Example 4.3.11 Trapezoidal Rule Revisit

Use degree 1 approximation of $f(x)$, $q_1(x)$, with $q_1(c) = f(c)$ and $q_1(d) = f(d)$:



Use Lagrange polynomials to build $q_1(x)$:

$$\ell_1(x) = \frac{x-d}{c-d}, \quad \ell_2 = \frac{x-c}{d-c}, \quad q_1(x) = f(c)\ell_0(x) + f(d)\ell_1(x).$$

Then,

$$\begin{aligned} I(f) &\approx I(q_1) \\ &= f(c)I(\ell_0) + f(d)I(\ell_1) && Compare: \; w_0 f(x_0) + w_1 f(x_1) \\ &= f(c)\left(\frac{d-c}{2}\right) + f(d)\left(\frac{d-c}{2}\right) \\ &= (d-c)\left(\frac{f(c)+f(d)}{2}\right), \end{aligned}$$

which is exactly the trapezoidal rule.

Definition 4.3.12 (Interpolatory Quadrature). Given data $\{(x_i, f_i)\}_{i=0}^N$, ordering $x_0 < x_1 < \cdots < x_N$, and the interpolating polynomial $q_N(x)$. We can always write

$$R(f) = q_N(x) = \sum_{i=0}^{N} f(x_i)\ell_i(x),$$

where $\ell_i(x)$'s are Lagrange polynomials:

$$\begin{aligned} I(f) \approx R(f) &= I(q_N) \\ &= I\left(\sum_{i=0}^{N} \underbrace{f(x_i)}_{\text{constant}} \overbrace{\ell_i(x)}^{\text{variable}}\right) \\ &= \sum_{i=0}^{N} f(x_i) \underbrace{I(\ell_i)}_{w_i} && Integration \; is \; a \; linear \; functional \end{aligned}$$

So, $w_i = I(\ell_i) = \int_a^b \ell_i(x)\,\mathrm{d}x$.

Definition 4.3.13 (Newton-Cotes Rules). We are working with interpolatory quadrature with equally spaced points. That is, $x_1 = a + ih$ and $h = (b-a)/N$. We have two types of

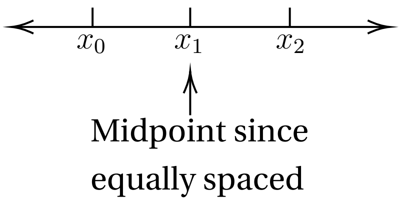
Newton-Cotes Rules:

- Closed $(N+1)$-point Newton-Cotes: We will use all points $x_0, \ldots, x_N$, including end-points.
- Open $(N-1)$-point Newton-Cotes: Use only interior points: $x_1, \ldots, x_{N-1}$.

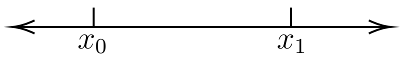
Example 4.3.14 Midpoint Rule and Trapezoidal Rule as Newton-Cotes

The midpoint rule is equivalent to an open 1-point Newton Cotes. On the other hand, the trapezoidal rule is equivalent to a closed 2-point Newton-Cotes.

Midpoint Rule



Trapezoidal Rule



Theorem 4.3.15 Simpson's Rule

The Simpsons's Rule is a closed 3-point Newton-Cotes. The Simpson's Rule states that

$$I(f) = \int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Proof 1.

$$\begin{aligned}
w_0 &= \frac{b-a}{6} = \int_a^b \ell_0(x)\,\mathrm{d}x = \int_a^b \frac{\left(x - \frac{a+b}{2}\right)(x-b)}{\left(a - \frac{a+b}{2}\right)(a-b)}\,\mathrm{d}x \\
w_1 &= \frac{4(b-a)}{6} = \int_a^b \ell_1(x)\,\mathrm{d}x \\
w_2 &= \frac{b-a}{6} = \int_a^b \ell_2(x)\,\mathrm{d}x
\end{aligned}$$

∎

4.4 Error in Integration

Theorem 4.4.1 Integral Mean Value Theorem

Suppose $g(x)$ and $w(x)$ are continuous on $[a,b]$, and $w(x)$ is non-negative on (a,b). Then, for some point $\eta \in (a,b)$, we get

$$\int_a^b \underbrace{w(x)}_{\text{weight}} g(x)\,\mathrm{d}x = \left\{\int_a^b w(x)\,\mathrm{d}x\right\} g(\eta).$$

Proof 1. Since $g(x)$ is continuous on $[a, b]$, by the Extreme Value Theorem, $g(x)$ attains a maximum and minimum on $[a, b]$. Let $m = \min_{x \in [a,b]} g(x)$ and $M = \max_{x \in [a,b]} g(x)$. Then, since $w(x) \geq 0 \quad \forall x \in (a, b)$, we have

$$m \int_a^b w(x)\,\mathrm{d}x \leq \int_a^b w(x)g(x)\,\mathrm{d}x \leq M \int_a^b w(x)\,\mathrm{d}x.$$

Since $g(x)$ is continuous on $[a, b]$, by the Intermediate Value Theorem, $g(x)$ attains every value between m and M. In other words, for any y *s.t.* $m \leq y \leq M$, $\exists \eta$ *s.t.* $g(\eta) = y$. Therefore, $\exists \eta \in (a, b)$ *s.t.* the statement holds. ■

Theorem 4.4.2 Error in Trapezoidal Rule

$$I(f) - R_T(f) = -\frac{(d-c)^3}{12} f''(\eta) \quad \text{for some } \eta \in (a, b)$$

Proof 2. The error in trapezoidal rule is given by

$$\begin{aligned} I(f) - R_T(f) &= I(f) - I(q_1) && q_1 \textit{ is a linear approximation} \\ &= I(f - q_1) \\ &= \int_c^d \{\text{error in linear interpolation}\}\,\mathrm{d}x \end{aligned}$$

So, error in integration is the integral of error in interpolation. Recall

$$f(x) - q_1(x) = \frac{\omega(x)}{2!} f''(\xi_x), \text{ where } \omega(x) = (x-c)(x-d).$$

Then,

$$\begin{aligned} I(f) - R_T(f) &= \int_c^d \underbrace{\frac{(x-c)(x-d)}{2}}_{w(x)} \underbrace{f''(\xi_x)}_{g(x)}\,\mathrm{d}x \\ &= -\int_c^d \frac{(x-c)(d-x)}{2} f''(\xi_x)\,\mathrm{d}x \\ &= -\left\{\int_c^d \frac{(x-c)(d-x)}{2}\,\mathrm{d}x\right\} f''(\eta) \\ &= -\frac{(d-c)^3}{2} f''(\eta). \end{aligned}$$

■

Corollary 4.4.3 If $f''(\eta) \geq 0$, then we always overestimate the integral since the error is negative. If $f''(\eta) < 0$, then we always underestimate the integral since error is positive.

Theorem 4.4.4 Errors in Composite Trapozoidal Rule

$$I(f) - R_{CT}(f,h) = -\frac{h^3}{12} N f''(\eta) = -\frac{h^2}{12} \underbrace{(b-a)}_{=Nh} f''(\eta), \quad N = \text{ \# of intervals}$$

Theorem 4.4.5 Errors in Midpoint Rule and Composite Midpoint Rule

$$I(f) - R_M(f) = \frac{(d-c)^3}{24} f''(\eta);$$

$$I(f) - R_{CM}(f) = \frac{h^3}{24} N f''(\eta) = \frac{h^2}{24}(b-a) f''(\eta).$$

Definition 4.4.6 (Degree of Precision/DOP). We say $R(f)$ has $\mathrm{DOP} = m$ if

$$R(p_k) = I(p_k) \quad \text{for } k = 0, 1, \dots, m,$$

and

$$R(p_{m+1}) \neq I(p_{m+1}).$$

Theorem 4.4.7 DOP Test
$R(f)$ has a $\mathrm{DOP} = m$ if $R(x^k) = I(x^k)$ for $k = 0, 1, \dots, m$ and $R(x^{m+1}) \neq I(x^{m+1})$.

Example 4.4.8 DOP of Midpoint Rule, Trapezoidal Rule, and Simpson's Rule

- Midpoint rule has $\mathrm{DOP} = 1$.

  Proof 3.

k	degree k	$I(x^k)$	$R(x^k)$
0	$p_0(x) = 1$	$\int_a^b 1 \, \mathrm{d}x = b - a$	$(b-a)(1)$
1	$p_1(x) = x$	$\int_a^b x \, \mathrm{d}x = \frac{1}{2}(b^2 - a^2)$	$(b-a)\left(\frac{a+b}{2}\right) = \frac{1}{2}(b^2 - a^2)$
2	$p_2(x) = x^2$	$\int_a^b x^2 \, \mathrm{d}x = \frac{1}{3}(b^3 - a^3)$	$(b-a)\left(\frac{a+b}{2}\right)^2 = \frac{1}{4}(a+b)^2(b-a)$

  So, the Midpoint Rule has a $\mathrm{DOP} = 1$. ■

  Remark. 4.2 *Super-convergence in Midpoint Rule: we use a degree* 0 *polynomial to interpolate our data, but we end up with a* $\mathrm{DOP} = 1 > 0$.

- Trapezoidal Rule has a $\mathrm{DOP} = 1$.
- Simpson's Rule has a $\mathrm{DOP} = 3$. (Super-convergence)

Remark. 4.3 *Super-convergence generally means we use a bad interpolation but get an accurate approximation of integral. (Degree of interpolation* $< \text{DOP}$*)*

Theorem 4.4.9 Method of Underdetermined Coefficient
Goal: maximize DOP by choosing weights (given $x_0, \ldots, x_N$ on canonical interval $[-1, 1]$). When $k = 0$, we have

$$\int_{-1}^{1} 1 \, \mathrm{d}x = \sum_{i=0}^{N} w_i(1) \quad \text{DOP} \geq 0$$

When $k = 1$, we have

$$\int_{-1}^{1} x \, \mathrm{d}x = \sum_{i=0}^{N} w_i x_i \quad \text{DOP} \geq 1$$

$$\vdots$$

When $k = m$, we have

$$\int_{-1}^{1} x^m \, \mathrm{d}x = \sum_{i=0}^{N} w_i x_i^m \quad \text{DOP} \geq m$$

If the next equation is not satisfied, $\text{DOP} =,$. That is,

$$\int_{-1}^{1} x^{m+1} \, \mathrm{d}x \neq \sum_{i=0}^{N} w_i x_i^{m+1}$$

We now have $m+1$ equations and $N+1$ unknowns $(w_0, \ldots, w_N)$. To find the weights, we solve a linear system.

Example 4.4.10 Method of Underdetermined Coefficients
Suppose $x_0 = -1,\ x_1 = 0,\ x_2 = +1$. Develop a $R(f) = w_0 f(-1) + w_1 f(0) + w_2 f(+1)$.
Solution 4.

k	degree k	$I(x^k)$	$R(x^k)$
0	$p_0(x) = 1$	$\int_{-1}^{1} 1 \, \mathrm{d}x = 2$	$w_0 + w_1 + w_2$
1	$p_1(x) = x$	$\int_{-1}^{1} x \, \mathrm{d}x = 0$	$-w_0 + w_2$
2	$p_2(x) = x^2$	$\int_{-1}^{1} x^2 \, \mathrm{d}x = \frac{2}{3}$	$w_0 + w_2$
3	$p_3(x) = x^3$	$\int_{-1}^{1} x^3 \, \mathrm{d}x = 0$	$-w_0 + w_2$
4	$p_4(x) = x^4$	$\int_{-1}^{1} x^4 \, \mathrm{d}x = \frac{2}{5}$	$w_0 + w_2$

So, we can form the following system of equations:

$$\begin{cases} w_0 + w_1 + w_2 = 2 \\ -w_0 + w_2 = 0 \\ w_0 + w_2 = 2/3 \end{cases} \implies \begin{cases} w_0 = 1/3 \\ w_1 = 4/3 \\ w_2 = 1/3 \end{cases}$$

Therefore, we have a $\mathrm{DOP} = 3$, which is a case of superconvergence.

$$R(f) = \frac{(+1)-(-1)}{6}\Big[f(-1) + 4f(0) + f(1)\Big] \longrightarrow \text{Simpson's Rule}$$

□

Theorem 4.4.11 Peano's Theorem
Suppose

- $R(f)$ with $\mathrm{DOP} = m$, and
- integrand $f(x)$ and its first $m+1$ derivatives exist and are continuous on $[a, b]$.

Then, there exists a function $K(x)$, the Peano's Kernel, for which

$$I(f) - R(f) = \int_a^b K(x) f^{(m+1)}(x)\,\mathrm{d}x$$

Corollary 4.4.12 If $K(x)$ does not change sign on $[a, b]$, then by the Integral Mean Value Theorem, we get

$$I(f) - R(f) = \kappa f^{(m+1)}(\eta),$$

for some $\eta \in (a, b)$ and κ is a number called the *Peano's constant.*

Remark. 4.4 *Midpoint Rule, Trapezoidal Rule, Simpson's Rule satisfy the condition that* $K(x)$ *does not change sign on* $[a, b]$.

Example 4.4.13 Peano's Kernel
On $[-1, 1]$, the Simpson's Rule gives us

$$R(f) = \frac{1}{3}\Big[f(-1) + 4f(0) + f(1)\Big],$$

with $\mathrm{DOP} = 3$. Then, by Peano's Theorem,

$$I(f) - R(f) = \int_{-1}^{1} K(x) f^{(4)}(x)\,\mathrm{d}x = \kappa f^{(4)}(\eta).$$

Estimate the Peano's constant κ.

Solution 5.

Choose $f(x) = x^4$, so we have $f^{(4)}(x) = 24$. Then

$$I(x^4) - R(x^4) = \kappa f^{(4)}(\eta) \implies \frac{2}{5} - \frac{2}{3} = 24\kappa \implies \kappa = -\frac{1}{90}$$

□

4.5 Adaptive Integration and More

Theorem 4.5.1 Characteristics of Adaptive Integration

- Fewer function evaluations
- Control of error
- Goal: estimate $I(f,T) \equiv \int_T f(x)\,\mathrm{d}x$, where $T = [a,b]$.

Notation 4.5.2. T^*: subinterval of T. $R_1(f,T^*)$ and $R_2(f,T^*)$ are both quadrature rules, but R_2 is more accurate (with higher DOP/composite method).

Definition 4.5.3 (Etimate Error).

$$\begin{aligned} E(f,T^*) &= |I(f,T^*) - R_1(f,T^*)| \\ &\approx |R_2(f,T^*) - R_1(f,T^*)| \end{aligned}$$

Algorithm 5: Adaptive Integration

1. Partition T into subintervals T_i, where $i = 1, \ldots, n$;
2. Compute $R(f,T_i)$ and $E(f,T_i)$ for $i = 1, \ldots, n$;
3. Find the subinterval with largest error and bisect. Say T_{i^*} is the interval, then we turn it into $T_{i^*}^{\text{left}}$ and $T_{i^*}^{\text{right}}$;
4. Repeat. Stop when we get $\sum_{i=1}^{n} E(f,T_i) \leq \underbrace{\text{tolerance}}_{\text{to be chosen}}$;

Definition 4.5.4 (Gauss Rules).

- Main idea: maximize DOP by choosing weights w_i and nodes x_i.
- Assumption: canonical interval $[-1,1]$.
- Properties:
  - positive weights: $w_i > 0$

- interior nodes: $x_i \in (-1, 1)$ for $i = 0, \ldots, N$
- symmetry
- interlacing: when comparing N-point and $(N+1)$-point, those nodes will appear in the interweaving/interlacing order. *This guarantees good sampling of integration intervals*
- interpolatory
- DOP of $(N+1)$-point Gauss Quadrature is $2N+1$.

Remark. 4.5 *There are other rules such as Lobatto Rules or Gauss-Kronrod Rules. They are also widely used in practice.*

Theorem 4.5.5 Transformation from Canonical Interval

$$\int_{g(c)}^{g(d)} f(x)\,\mathrm{d}x = \int_c^d f(g(t))g'(t)\,\mathrm{d}t,$$

where $f(g(t))$ represents evaluating f on $[a, b]$, and $g'(t)$ shows how much $[a, b]$ is stretched/shrunk from $[-1, 1]$. In our case, $c = -1,\ d = +1$, and $g : [-1, 1] \to [a, b]$. We will use a linear map that maps -1 to a and $+1$ to b:

$$g(t) = \frac{(b-a)}{2}t + \frac{(b+a)}{2}$$

So,

$$\begin{aligned} I(f) \equiv \int_a^b f(x)\,\mathrm{d}x &= \underbrace{\frac{b-a}{2}}_{g'(t)} \int_{-1}^{1} f\left(\frac{b-a}{2}t + \frac{b+a}{2}\right)\mathrm{d}t \\ &\approx \frac{b-a}{2}\sum_{i=0}^{N} w_i^* f\left(\frac{b-a}{2}t_i^* + \frac{b+a}{2}\right) \\ &= \sum_{i=0}^{N} w_i f(x_i), \end{aligned}$$

where $w_i = \dfrac{b-a}{2}w_i^*$ and $x_i = \dfrac{b-a}{2}t_i^* + \dfrac{b+a}{2}$.

5 Root Finding

5.1 Fixed Point Iteration (FPI)

Definition 5.1.1 (Root). A *root* of a function f is a point x_* where $f(x_*) = 0$.
Definition 5.1.2 (Simple Root). A point x_* is a *simple root* of a function f if the following hold:

- x_* is a root, i.e., $f(x_*) = 0$,
- $f'(x)$ exists everywhere on an open interval containing x_*, and
- $f'(x_*) \neq 0$.

Remark. 5.1 *In other words, we corss the x-axis "like a line."*

Definition 5.1.3 (Fixed Point). A point x_* is a *fixed point* of function g if $x_* = g(x_*)$.

Remark. 5.2 *It is those points where $g(x)$ intersects $y = x$.*

Theorem 5.1.4 Fixed Point and Root
If x_* is a fixed point of g, then x_* is a root of $f(x) = x - g(x)$. Conversely, if x_* is a root of $f(x)$, then x_* is a fixed point of $g(x) = x - f(x)$.

Example 5.1.5
Given $g(x) = 5 - \dfrac{3}{x+2}$. Find fix points of this function.
Solution 1.

$$x = g(x) \implies x = 5 - \frac{3}{x+2} \implies x^2 - 3x + 7 = 0 \implies x_* = \frac{3 \pm \sqrt{37}}{2}$$

□

Algorithm 6: Fixed Point Iteration (FPI)

Goal: Find x_* such that $x_* = g(x_*)$;
Start with a guess x_0. Repeat the following step:

$$x_{n+1} = g(x_n), \quad n = 0, 1, \ldots,$$

where g is the FPI function;

Remark. 5.3 *The result of FPI depends significantly on our initial guess!*

Example 5.1.6

Find fixed points of $g(x) = e^{-x}$.

Guess: $x_0 = 0.5$. Then,

$$x_1 = g(x_0) = g(0.5) \approx 0.6065; \quad x_2 = g(x_1) = g(0.6065) \approx 0.5452; \ldots$$

Remark. 5.4 *Consider* $h(x) = -\ln(x)$. x_* *of* $g(x)$ *is also a fixed point of* $h(x)$*:*

$$x_* = g(x_*) = e^{x_*} \implies \ln(x_*) = -x_* \implies x_* = -\ln(x_*) = h(x_*).$$

However, using the guess $x_0 = 0.5$ *to solve for a fixed point of* $h(x)$ *would not result in a converging fixed point.*

5.2 Convergence Iteration Analysis

Definition 5.2.1 (Error). The *err*r of FPI is defined as

$$e_n \coloneqq x_n - x_*$$

Theorem 5.2.2

If FPI converges, then $\{|e_n|\}_{n=0}^{\infty}$ decreases monotonically eventually.

Remark. 5.5 *We will assume* $g'(x)$ *is continuous whenever we need.*

Theorem 5.2.3 Mean Value Theorem, MVT

By the mean value theorem, we can find the relationship between e_{n+1} and e_n.

$$\begin{aligned} e_{n+1} &= x_{n+1} - x_* \\ &= g(x_n) - g(x_*) \\ &= g'(\eta_n)(x_n - x_*) \\ e_{n+1} &= g'(\eta_n)e_n \end{aligned}$$

Theorem 5.2.4 Convergence Analysis

We have $|e_{n+1}| < |e_n| \iff |g'(\eta_n)| \le G < 1$.

Theorem 5.2.5 Tangent Line Condition

If $|g'(x)| \le G < 1 \quad \forall x \in I = [x_* - r, x_* + r]$, then FPI converges for every starting guess $x_i \in I$.

Corollary 5.2.6 If $g'(x_*) > 0$, convergence is on-sided (= the sign of error does not change). If $g'(x_*) < 0$, convergence is two-sided (= the sign of error changes).

Theorem 5.2.7
The smaller $|g'(x_*)|$, the faster we converge.

Theorem 5.2.8 Error Approximation
When x_n is close enough to x_*,

$$e_{n+1} \approx g'(x_*)e_n$$

Definition 5.2.9 (Linear, Superlinear). Convergence rate is *linear* if $g'(x_*) \neq 0$. Convergence rate is *superlinear* if $g'(x_*) = 0$.

Definition 5.2.10 (Convergence Rate). If $\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = K$, where $K > 0$, then the *convergence rate* is *order-*p. If $p = 1$ and $K < 1$, then convergence is *linear*. If $p = 2$, then we have *quadratic convergence*.

5.3 Root Finding Methods

Algorithm 7: Newton Iteration

1. To find a simple root $f(x) = 0$, we can use the FPI function

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

   and apply the fixed point iteration algorithm;

2. So, in each iteration, we have

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

Proof 1. This comes from Taylor Series:

$$f(z) = f(x) + (z-x)f'(x) + (z-x)^2\frac{f''(x)}{2} + \cdots$$
$$f(z) \approx f(x) + (z-x)f'(x).$$

Find the root of linear approximation (solving for z):

$$0 = f(x) + (z-x)f'(x)$$
$$z = \frac{xf'(x) - f(x)}{f'(x)} = x - \frac{f(x)}{f'(x)}$$

In practice, we have

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

■

Theorem 5.3.1 Convergence of the Newton Methods
We have superlinear convergence when we are close near the root with the Newton's Method.

Proof 2.

$$e_n = x_n - x_*; \quad e_{n+1} = g'(\eta_n)e_n.$$

Derivative of the Newton's Method FPI function:

$$\begin{aligned} g'(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left[x - \frac{f(x)}{f'(x)}\right] \\ &= 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{f'(x)^2} \\ &= 1 - 1 + \frac{f(x)f''(x)}{[f'(x)]^2} \\ g'(x) &= \frac{f(x)f''(x)}{[f'(x)]^2}. \end{aligned}$$

When we are near x_*, we approximate

$$\begin{aligned} e_{n+1} &\approx g'(x_*)e_n \\ g'(x_*) &= \frac{f(x_*)f''(x_*)}{[f'(x_*)]^2} = 0. \end{aligned}$$

So, we have superlinear convergence when we are close near the root. ■

Definition 5.3.2 (Rate of Convergence). If $\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = K$, K is a constant $\in (0, \infty)$, then the method has convergence rate p.

Example 5.3.3

- $e_{n+1} = \frac{1}{2}e_n, \quad p = 1$
- $e_{n+1} = e_n^2, \quad p = 2$
- $e_{n+1} = \frac{1}{3}e_n^3, \quad p = 3$

As p increases, the method converges faster.

Remark. 5.6 *p has to be greater than 1. For example, $e_{n+1} = e_n^{1/2}$ will give us a divergent method.*

Theorem 5.3.4 Rate of Convergence of Newton's Method
The Newton's Method has a rate of $p = 2$. It is converging quadratically.

Proof 3. Note that

$$\begin{aligned} e_{n+1} &= x_{n+1} - x_* \\ &= g(x_n) - g(x_*) \\ &= \left(g(x_*) + \underbrace{(x_n - x_*)}_{e_n} \underbrace{g'(x_*)}_{0} + (x_n - x_*)^2 \frac{g''(x_*)}{2} + \cdots \right) - g(x_*) \\ &= e_n^2 \cdot \frac{g''(x_*)}{2} \end{aligned}$$

So, the Newton's Method has a rate of $p = 2$. It is converging quadratically. ■

Remark. 5.7 *Drawback of Newton's Method: we need information of the derivative.*

Algorithm 8: Secant Method

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

where m_n is the slope of the secant line, computed by

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

This method does not require information of the derivative;

The order of convergence of the secant method is $p = \dfrac{1+\sqrt{5}}{2} = \varphi$, the golden ratio;

Remark. 5.8 *Drawback of Newton's Method and Secant Method:they are very sensitive to our initial guess. Bad initial guesses might even lead to divergence.*

Algorithm 9: Bisection Method

Choose a bracket $[a, b]$ (an interval) such that $f(a)$ and $f(b)$ are different signs. That is, $f(a)f(b) < 0$ (As f is continuous, by the intermediate value theorem, f must cross 0 in this interval);

Compute the midpoint $m = \dfrac{a+b}{2}$ and $f(m)$;

Form a new bracket: If $f(a)f(m) < 0$, then the new bracket is $[a, m]$. Otherwise, $f(m)f(b) < 0$, then the new bracket is $[m, b]$;

Repeat;

The bisection method is converging linearly;

Algorithm 10: Quadratic Inverse Interpolation

Assume f is invertible. Obviously, $f(x) = y \iff x = f^{-1}(y)$. Then, if x_* is a root of f, then $f(x_*) = 0 \iff x_* = f^{-1}(0)$;

Given there points: x_{n-2}, x_{n-1}, x_n;

Compute the corresponding y values: $y_{n-2} = f(x_{n-2})$, $y_{n-1} = f(x_{n-1})$, $y_n = f(x_n)$;

Interpolate $\{(y_{n-2}, \underbrace{f^{-1}(y_{n-2})}_{x_{n-2}}), (y_{n-1}, \underbrace{f^{-1}(y_{n-1})}_{x_{n-1}}), (y_n, \underbrace{f^{-1}(y_n)}_{x_n})\}$:

$$\begin{aligned} p_2(y) = \frac{(y-y_{n-1})(y-y_{n-2})}{(y_n-y_{n-1})(y_n-y_{n-2})} f^{-1}(y_n) &+ \frac{(y-y_n)(y-y_{n-2})}{(y_{n-1}-y_n)(y_{n-1}-y_{n-2})} f^{-1}(y_{n-1}) \\ &+ \frac{(y-y_n)(y-y_{n-1})}{(y_{n-2}-y_n)(y_{n-2}-y_{n-1})} f^{-1}(y_{n-2}) \end{aligned}$$

So, $x_{n+1} = p_2(0)$;

Interpolate $\{(y_{n-1}, f^{-1}(y_{n-1})), (y_n, f^{-1}(y_n)), (y_{n+1}, f^{-1}(y_{n+1}))\}$, and repeat.

Theorem 5.3.5

The error terms of quadratic inverse interpolation satisfy

$$e_{n+1} = Ke_n^{1.44}.$$

So, quadratic inverse interpolation is converging superlinearly.

5.4 Calculating Square Roots

Theorem 5.4.1

To compute $\sqrt{a}$ for $a > 0$, we are essentially finding the roots of the function given by $f(x) = x^2 - a$, using the Newton's Method:

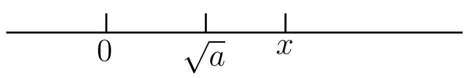
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^2 - a}{2x_n} = \frac{x_n + \frac{a}{x_n}}{2}$$

The second expression is better as we have less operations involved and dividing by 2 is easier.

Lemma 5.4.2 Consider $x_0 > 0$, then $e_1 = \dfrac{e_0^2}{2x_0} > 0$. So, $x_1 - \sqrt{a} > 0$ and we have one-sided convergence. Therefore, we have $e_n \geq 0$ for all $n \geq 1$. Therefore,

$$0 \leq \frac{e_n}{x_n} = \frac{x_n - \sqrt{a}}{x_n - 0} < 1$$

Let's consider the inequality using the number line and distance.



Theorem 5.4.3 Error Analysis
We will always converge when finding the square roots.

Proof 1. Consider the error terms:

$$\begin{aligned} e_n &= x_n - \sqrt{a} \\ e_{n+1} &= x_{n+1} - \sqrt{a} \\ &= \frac{\left(x_n + \frac{a}{x_n}\right)}{2} - \sqrt{a} = \frac{(x_n - \sqrt{a})^2}{2x_n} = \frac{e_n^2}{2x_n}. \end{aligned}$$

By Lemma 5.4.2, we have

$$e_{n+1} = \frac{e_n^2}{2x_n} = \left(\frac{e_n}{x_n}\right)\left(\frac{e_n}{2}\right) \leq \frac{e_n}{2}.$$

So, we will always converge. ■

Algorithm 11: A Good Starting Guess for Calculating Square Roots

1 Write a in base-2:

$$a = S(a)2^{e(a)},$$

where $S(a)$ is the significant, $1 \leq S(a) < 2$, and $e(a)$ is the exponent, an integer;

2 Compute $\sqrt{a}$:

$$\sqrt{a} = \begin{cases} \sqrt{S(a)} \cdot 2^{\frac{e(a)}{2}} & e(a) \text{ is even} \\ \sqrt{2S(a)} 2^{\frac{e(a)-1}{2}} & e(a) \text{ is odd} \end{cases}$$

So, finding $\sqrt{a}$ depends on finding the root of the significant;

3 Range reduction:

$$1 \leq S(a) < 2 \leq 2S(a) < 4,$$

with this reduced range, we can make good starting guess for calculation of square roots;

4 One additional trick: seeding;

5.5 Roots of Polynomials

Definition 5.5.1 (Polynomial of Degree-n). A polynomial of degree-n is given by

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

Theorem 5.5.2 Fundamental Theorem of Algebra
A polynomial of degree-n ($n \geq 1$) will have n roots by counting multiplicities, some of which could be complex.

Algorithm 12: Roots of Polynomials

Use Newton's Method to find a root. $\longrightarrow$ We need to evaluate $p_n(x)$ and $p_n'(x)$ quickly;
Factor out root. $\longrightarrow$ Synthetic division;
Repeat on remaining polynomial;

Theorem 5.5.3 Horner's Rule of Degree-3 Polynomials
Given a cubic polynomial:

$$\begin{aligned} p_3(x) &= a_3x^3 + a_2x^2 + a_1x + a_0 \\ &= (a_3x^2 + a_2x + a_1)x + a_0 \\ &= ((a_3x + a_2)x + a_1)x + a_0 \end{aligned}$$

We then can define the following temporary polynomials:

$$\begin{array}{c|c} t_3(x) = a_3 & t_3'(x) = 0 \\ t_2(x) = a_2 + xt_3(x) & t_2'(x) = t_3(x) + xt_3'(x) \\ t_1(x) = a_1 + xt_2(x) & t_1'(x) = t_2(x) + xt_2'(x) \\ p_3(x) \equiv t_0(x) = a_0 + xt_1(x) & p_3'(x) \equiv t_0'(x) = t_1(x) + xt_1'(x) \end{array}$$

In this way, our computation of $p_3(x)$ and $p_3'(x)$ invovles less operations of floating point numbers.

Algorithm 13: Horner's Rule in Practice

Input: $[a_n, a_{n-1}, \ldots, a_0]$ coefficients; x value of evaluation
Output: t value of the polynomial at x; t_p value of the derivative at x

```
begin
    Initialize: t_p = 0; t = a_n;
    for i = n − 1 to 0 do
        t_p = t + t_p * x;
        t = a_i + t * x;
```

Theorem 5.5.4 Synthetic Division/Polynomial Long Division
Suppose α is a root such that $p_N(\alpha) = 0$. Generally,

$$\frac{p_N(x)}{(x-\alpha)} = q_{N-1}(x) + \frac{b_0}{(x-\alpha)} \iff p_N(x) = (x-\alpha)q_{N-1}(x) + b_0$$

When α is a root, $b_0 = 0$, so

$$\frac{p_N(x)}{(x-\alpha)} = q_{N-1}(x)$$

Example 5.5.5 Long Division

$$\begin{array}{r@{\,}l} & x+9+\frac{66}{x-7} \\ x-7 & \overline{)\, x^2+2x+3} \\ & \underline{x^2-7x} \\ & 9x+3 \\ & \underline{9x-63} \\ & 66 \end{array}$$

So, $\dfrac{x^2+2x+3}{x-7} = x+9+\dfrac{66}{x-7}$ and $x^2+2x+3 = (x+9)(x-7)+66$.

Theorem 5.5.6 Synthetic Division in Practice
Given a degree-3 polynomial $p_3(x) = (a-\alpha)q_2(x) + b_0$. We can rewrite the equation into

$$a_3x^3 + a_2x^2 + a_1x + a_0 = (x-\alpha)\big(b_3x^2 + b_2x + b_1\big) + b_0$$

$$\implies \begin{cases} a_3 = b_3 \\ a_2 = b_2 - \alpha b_3 \\ a_1 = b_1 - \alpha b_2 \\ a_0 = b_0 - \alpha b_1 \end{cases} \implies \begin{cases} b_3 = a_3 \\ b_2 = a_2 + \alpha b_3 = a_2 + \alpha a_3 \\ b_1 = a_1 + \alpha b_2 = a_1 + \alpha(a_2 + \alpha a_3) \\ b_0 = a_0 + \alpha b_1 = a_0 + \alpha(a_1 + \alpha(a_2 + \alpha a_3)) \end{cases}$$

In summary, we want

- Find root α of $p_3(x)$
- Find the mapping: $[a_3, \ldots, a_0] \to [b_3, \ldots, b_1]$
- Find roots of $q_2(x)$ and repeat.

Theorem 5.5.7 Conditioning of Polynomial Roots
Given the true polynomial $p_n(x) = a_0 + a_1x + \cdots + a_nx^n$ with root z_0 such that $p(z_0) = 0$. We computed a polynomial $P_n(x) = A_0 + A_1x + \cdots + A_nx^n$ with root Z_0 such that $P(Z_0) = 0$ from our algorithm. Then, the error $= z_0 - Z_0$ can be approximated with

$$z_0 - Z_0 \approx \frac{P_n(z_0) - p_n(z_0)}{P_n'(Z_0)} = \frac{\sum\limits_{i=0}^{n}(A_i - a_i)z_0^i}{P_n'(Z_0)}.$$

Proof 1. We can approximate $P_n(z_0)$ using Taylor series, expanded about Z_0:

$$P_n(z_0) \approx \underbrace{P_n(Z_0)}_{0} + (z_0 - Z_0)P_n'(z_0)$$

$$P_n(z_0) - \underbrace{p_n(z_0)}_{0} \approx (z_0 - Z_0)P_n'(Z_0)$$

Therefore, we have

$$\text{error} = z_0 - Z_0 \approx \frac{P_n(z_0) - p_n(z_0)}{P_n'(Z_0)} = \frac{\sum_{i=0}^{n}(A_i - a_i)z_0^i}{P_n'(Z_0)}.$$

■

Remark. 5.9 *Theoretically, $A_i - a_i$ should be small. Therefore, the error becomes large when*

$$P_n'(Z_0)a \approx 0.$$

That means flatter polynomials at the roots will cause large errors.

6 Univariate Minimization

Remark. 6.1 (Goal) *The goal of this chapter is to find a minimum of continuous function f on $[a, b]$. By the Extreme Value Theorem, we will always find such a minimum.*

Remark. 6.2 (A Calculus Perspective) *If f is differentiable, local extrema occur when $f'(x) = 0$. That is a root finding problem! Suppose $x_1, \ldots, x_p$ are where $f'(x_o) = 0$. Then, the global minimization on the interval $[a, b]$ will be*

$$\min_{x\in[a,b]} f(x) = \min\{f(x_i), f(a), f(b)\}.$$

Definition 6.0.1 (Minimum/Minimizer). *Minimum* is the smallest value the function can take on $[a, b]$. *Minimizer* is the input (x) for which the function attains its minimum.

6.1 Find Minima Without Calculus

Definition 6.1.1 (U-Shaped Function). A function f is *U-shaped* on $[a, b]$ if

- f is continuous
- Strictly decreasing on $[a, x_m]$, where x_m is the minimizer.
- Strictly increasing on $[a_m, b]$.

If f is continuously differentiable, then f is *U-shaped* if $f'(x) = \begin{cases} < 0 & x \in [a, x_m] \\ > 0 & x \in [x_m, b] \end{cases}$

Remark. 6.3 *A U-shaped function has not to be differentiable (i.e., smooth)!*

Definition 6.1.2 (Bracket for a Local Minimum). $[a, b]$ is a *bracket for the minimum* if the minimizer $x_m \in [a, b]$.

Definition 6.1.3 (Search Points). Given a bracket $[a, b]$, c, d are *search points* if the have $a < c < d < b$.

Algorithm 14: Bracket Refinement

Input: a bracket $[a, b]$ and two search points c, d.

1 If $f(c) \leq f(d)$, then $[a, d]$ is a bracket for the minimum;

2 If $f(c) \geq f(d)$, then $[c, d]$ is a bracket for the minimum;

Algorithm 15: Golden Section Search

Input: a bracket for the minimum $[a, b]$, $r = \dfrac{3-\sqrt{5}}{2}$.

1 $c \equiv a + r(b-a), \qquad d \equiv b - r(b-a)$;

Remark. 6.4 (Relationship to the Golden Ratio) $\varphi = \frac{1+\sqrt{5}}{2} = \frac{1}{1-r}$.

Remark. 6.5 *Advancement of the Golden Section Search*

- *balance when we refine*
- *each refinement cuts the bracket down by a factor of* $1 - r \approx 0.618$.
- *refinements reuse search points* $\longrightarrow$ *great for computation.*

  (to derive this point, verify that the next search point $c' \equiv c + r(b-c)$ *by substituting* $c = a + r(b-a)$. *Also, keep in mind that* r *is a root of* $x^2 - 3x + 1$. *Simply the previous equation, we will have* $c' = d$ *eventually.)*

Theorem 6.1.4 Condition to Stop the Golden Section Search

1. Function has a similar value at all four points
2. Brackets is small:
$$|b-a| \leq \sqrt{\frac{\varepsilon}{L}},$$
   where ε is the working precision, and l is some constant.

Remark. 6.6 *For example, if we want five digits of precision,* ε *should be around* 10^{-6} *or* 10^{-5}, *but generally, the smaller* ε, *the more iterations we need.*

Proof 1. Here, we offer some intuition of why that condition for small brackets will work.

If f is continuously differentiable, then we have

$$\left|\frac{f(x) - f(x_m)}{x - x_m}\right| \leq L,$$

where l is the Lipschitz constant. This inequality shows that for all $x \in [a,b]$ and $x \neq x_m$, the slope is bounded:

$$|f(x) - f(x_m)| \leq L|x - x_m| \leq L|b-a| \leq \text{some upper bound}$$

Now, let's discuss the role of L, the Lipschitz constant. Assume f is second-order differentiable. We can approximate x_m with x by using the Taylor's series:

$$f(x) \approx f(x_m) + \underbrace{f'(x_m)(x - x_m)}_{0} + \frac{f''(x_m)}{2}(x - x_m)^2$$

$$\underbrace{f(x)}_{\text{computed}} \approx \underbrace{f(x_m)}_{\text{true}} \underbrace{(1+\psi)}_{\text{relative error}}$$

So, we would have

$$\psi \equiv \frac{f''(x_m)}{2f(x_m)}(x-_m)^2$$
$$|\psi| \leq \left|\frac{f''(x_m)}{2f(x_m)}\right|(b-a)^2 = L(b-a)^2$$

Therefore, we know L depends on the curvature of $f''(x_m)$ and the value of $f(x_m)$. ■