廣東工業大學
Guangdong University of Technology

Root finding

GongYi龚怡

2167570874@qq.com

Link

Slides：

https://pan.baidu.com/s/1uqWdtLXJhJ9TvKN1CsK_9w

Access code ：math

Software:

https://pan.baidu.com/s/1iXhXryPJG-YNYF-RedTZ1Q

Access code : 57fs

Linear Function

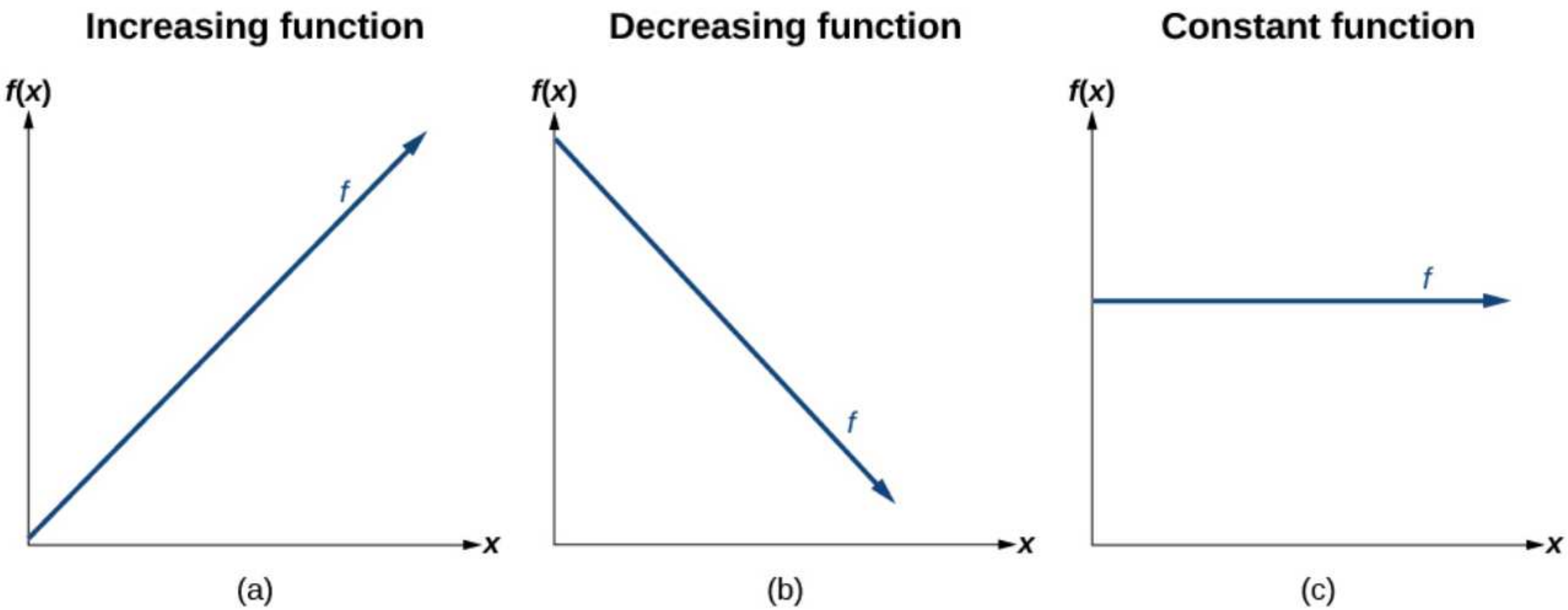
A linear function is a function whose graph is a line.

Linear functions can be written in the slope-intercept form of a line

$$f(x)=mx + b$$

where b is the initial or starting value of the function (when input, $x=0$), and m is the constant rate of change, or slope of the function. The y-intercept is at $(0,b)$.

Linear Function



Linear Function

The slope determines if the function is an increasing linear function, a decreasing linear function, or a constant function.

$f(x) = mx + b$ is an increasing function if $m>0$.

$f(x) = mx + b$ is an decreasing function if $m<0$.

$f(x) = mx + b$ is a constant function if $m=0$.

Linear Function

Standard form

A, **B**, and **C** are integers

$$Ax + By = C$$

Slope-intercept

Slope **m** y-intercept **b**

$$y = mx + b$$

Point-Slope form

Slope **m** Point (x_1, y_1)

$$y - y_1 = m(x - x_1)$$

Linear Function

In general, a linear function is a function of the form

$$ax + by + \ldots + cz + d = 0$$

Non-linear Function

A nonlinear function is a function which is not linear. There are several types of nonlinear functions:

- absolute Value $f(x) = |x|$
- Polynomial Functions $f(x) = a_n x^n + \cdots + a_1 x + a_0, n > 1$
- Rational Functions, $f(x) = \frac{P(x)}{Q(x)}$, $P(x)$ and $Q(x)$ are polynomial functions
- Exponential Functions $f(x) = a^x, a > 0$
- Radioactive Decay $A(t) = A(0)e^{-kt}, t \geq 0$

Linear vs. Non-linear Function

Sr. No	Linear function	Nonlinear function
1	Can be plotted as a straight line with no curves	Do not form a straight line; instead, they always have a curve
2	The degree will always equal 1	The degree will always be greater than 1
3	Always form a straight line in the XY-Cartesian plane, and the line can extend to any direction depending upon the limits or constraints of the equation	Always form a curved graph. The curve of the graph will depend upon the degree of the function. The higher the degree, the higher the curvature.
4	Can be written as *f(x)=mx + b, "m" is the slope, while "b" is the constant value. "x" and "y" are the variables of the equation.*	An example of a nonlinear equations is $ax^2 + bx = c$, the degree is 2, so it is a quadratic equation. If the degree is 2, it will be a cubic equation.
5	Examples of linear functions $3x + y = 4$ $4x + 1 = y$ $2x + 2y = 6$	Examples of nonlinear functions $2x^2 + 6x = 4$ $3x^2 - 6x + 10 = 0$ $3x^3 + 2x^2 + 3x = 4$

Problem

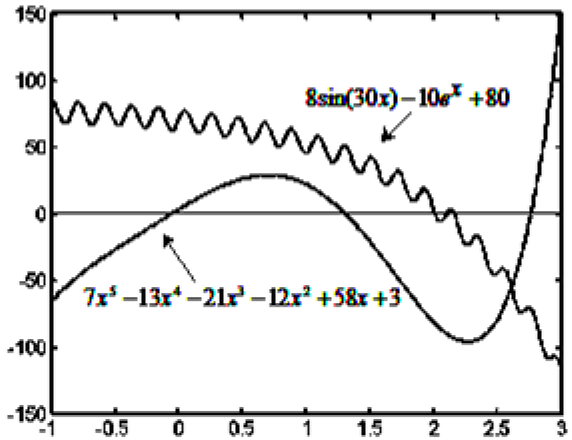
Generally, there are no formulas to find roots for nonlinear functions. Even it is difficult to find that there is a root or not, or the number of roots for transcendental equations,

n^{th}degree polynomial function

$$a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$$

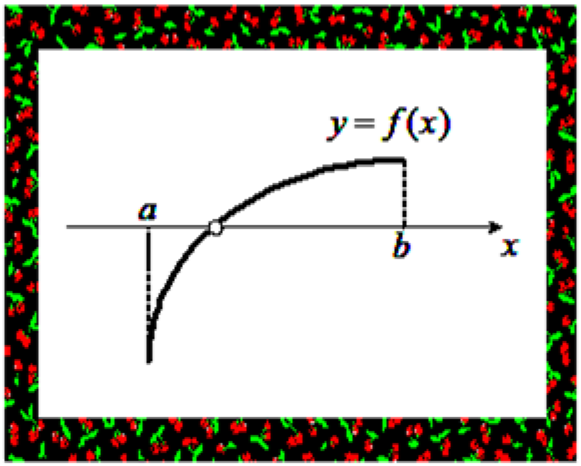
Transcendental equation

$$e^{-x} + \ln x - \sin\frac{\pi x}{2} = 0$$



Then, is there any way to solve nonlinear equations? Or how to make sure the accuracy of the approximate solution

A variety of numerical algorithms, such as the bisection method, fixed-point iteration, Newton's method, and the secant method, are commonly used to solve nonlinear equations



Contents

Numerical solutions of nonlinear equations

- Bisection method
- Newton-Raphson Method

Bisection method (Binary-search method)

Assume $f \in C[a, b]$， with $f(a) \cdot f(b) < 0$， then it follows that there exists a root $\alpha \epsilon (a, b)$.

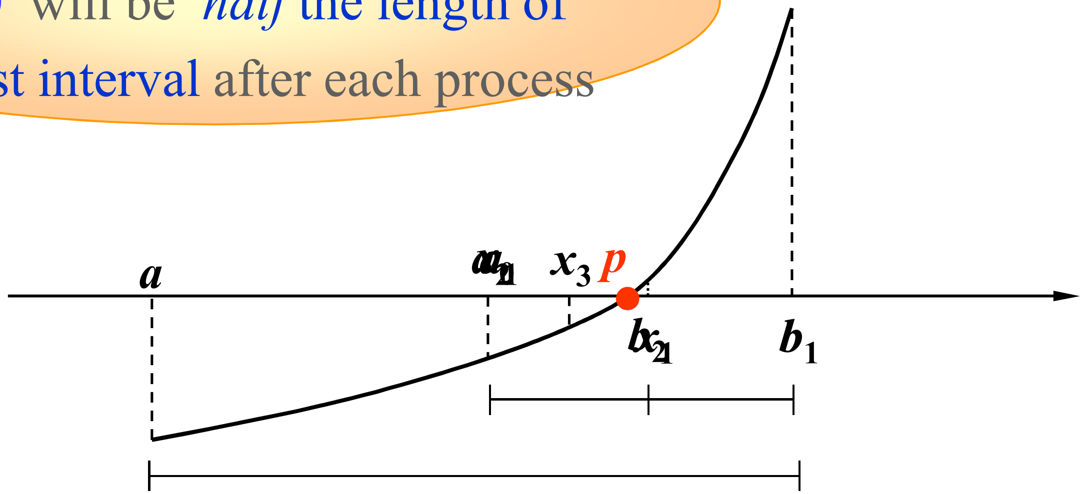
(**Note**: There may be more than one root in the interval.)

Bisection method (Binary-search method)

- Assume $f \in C[a, b]$, with $f(a) \cdot f(b) < 0$, then it follows that there exists a root $\alpha \epsilon (a, b)$.

Interval $[a_n, b_n]$ of solution for $f(x)=0$ will be *half* the length of the last interval after each process

When to stop?



Bisection method (Binary-search method)

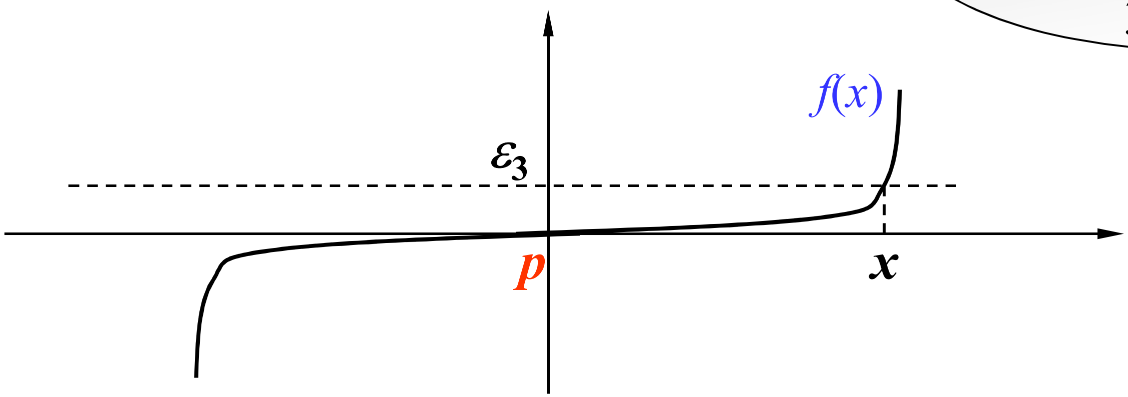
❖ **There are 3 stopping criteria :**

(1) $|x_n - x_{n-1}| < \varepsilon_1$ **Absolute error of two adjacent x**

(2) $\dfrac{|x_n - x_{n-1}|}{|x_n|} < \varepsilon_2, x_n \neq 0$ **Relative error of two adjacent x**

(3) $|f(x_n)| < \varepsilon_3$ **Function value error**

Accuracy of x cannot be guaranteed in the 3rd rule



Bisection method (Binary-search method)

Now let's try to use bisection method to find α. Let c be the midpoint of the interval $[a, b]$, i.e.,

$$c = \frac{1}{2}(a + b)$$

and consider the product $f(a)f(c)$. There are three possibilities:

1. $\boldsymbol{f(a)f(c) < 0}$; this means that a root (there might be more than one) is between a and c, i.e., $\alpha \in [a, c]$.

2. $\boldsymbol{f(a)f(c) = 0}$; if we assume that we already know $f(a) \neq 0$, this means that $f(c) = 0$, thus $\alpha = c$ and we have found a root.

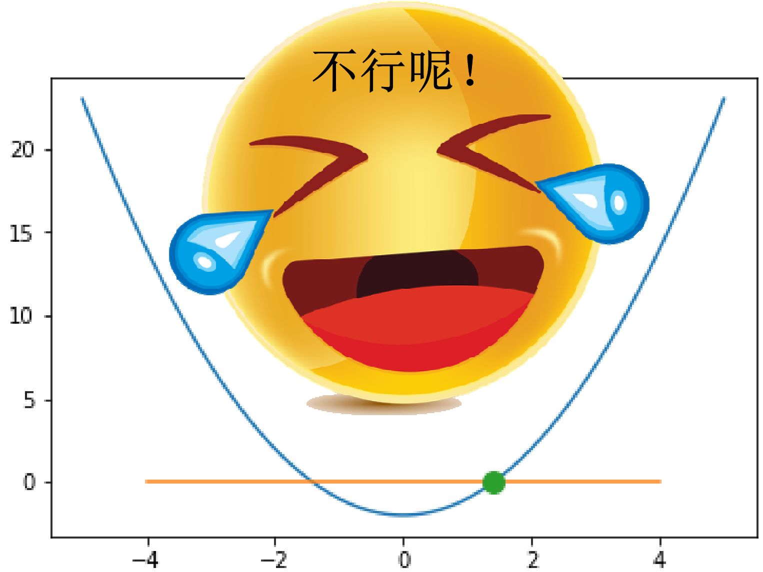
3. $\boldsymbol{f(a)f(c) > 0}$; this means that a root must lie in the other half of the interval, i.e., $\alpha \in [c, b]$.

Bisection method (Binary-search method)

❖ Assume $f \in C[a, b]$， with $f(a) \cdot f(b) < 0$， then it follows that there exists a root $\alpha \epsilon (a, b)$.

■ **Example： Use bisection method to solve $\boldsymbol{f(x) = x^2 - 2 = 0}$** on the interval [1.3, 1.5].

```
iter = 0        #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT:
    xmid = (xlow + xup) / 2       #计算新的中
    iter += 1                     #迭代计数加
    if f(xmid) > 0:               #中点函数值
        xup = xmid                #更新xup
    else:                         #中点函数值
        xlow = xmid               #更新xlow
```



❖ **Question： This program can be used to solve the solution on [-2,0] directly？**

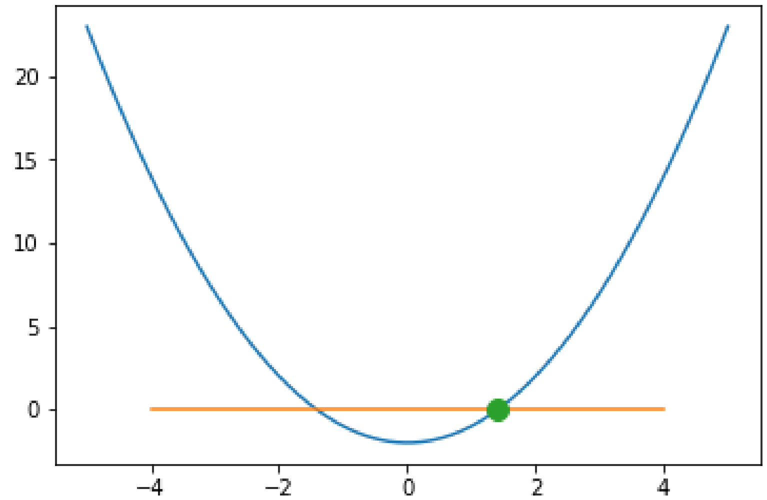
Bisection method (Binary-search method)

❖ Assume $f \in C[a, b]$， with $f(a) \cdot f(b) < 0$， then it follows that there exists a root $\alpha \epsilon (a, b)$.

- **Reason: If *f*(*midpoint*) > 0, [low, up]→ [low, midpoint]. This only holds for solutions on the interval [0, 2]**

But on the interval [-2, 0], the assignment statements must be swapped

```python
iter = 0      #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT:
    xmid = (xlow + xup) / 2     #计算新的中
    iter += 1                   #迭代计数加
    if f(xmid) > 0:             #中点函数值
        xup = xmid              #更新xup
    else:                       #中点函数值
        xlow = xmid             #更新xlow
```



Algorithm 4.1 Bisection Method (Outline Form)

1. Given an initial interval $[a_0, b_0] = [a, b]$, set $k = 0$

and proceed as follows:

2. Compute $c_{k+1} = a_k + \frac{1}{2}(b_k - a_k)$;

3. If $f(c_{k+1})\, f(a_k) < 0$, set $a_{k+1} = a_k$, $b_{k+1} = c_{k+1}$;

4. If $f(c_{k+1})\, f(b_k) < 0$, set $b_{k+1} = b_k$, $a_{k+1} = c_{k+1}$;

5. Update k and go to Step 2.

Bisection method (Binary-search method)

- Assume $f \in C[a, b]$, with $f(a) \cdot f(b) < 0$, then it follows that there exists a root $\alpha \epsilon (a, b)$.

- **Pay attention to the machine accuracy when programming**
  - **Minimize errors**

1. Is there a root localized to an interval $[a, b]$

It is best not to use it directly: $\boldsymbol{f(a) \cdot f(b) < 0}$

It should be used: $\mathbf{sign}(\boldsymbol{f(a)}) \cdot \mathbf{sign}(\boldsymbol{f(b)}) < 0$

Symbolic functions:

$$\mathrm{sign}(x) = \begin{cases} -1, & \text{if} \quad x < 0 \\ 0, & \text{if} \quad x = 0 \\ 1, & \text{if} \quad x > 0 \end{cases}$$

Avoid error caused by Multiplication overflow

Bisection method (Binary-search method)

❖ **Pay attention to the machine accuracy when programming**

- **Minimize errors**

2. Calculate midpoint

It is best not to use it directly: $x_n = \dfrac{a_n + b_n}{2}$

It should be used: $x_n = a_n + \dfrac{b_n - a_n}{2}$

The reason is when both a_n and b_n are close to the accuracy limit of the machine, The resulting value $x_n = \dfrac{a_n + b_n}{2}$ may go beyond the range of [a_n , b_n].

Modified bisection method

```
import numpy as np

a = 2           # f(x) = x*x - a
LIMIT = 1e-20   #终止条件

#方程函数f()定义
def f(x):
    """函数值的计算"""
    return x * x - a
# f()函数结束

#-----主执行部分------
#初始区间设置
xlow = float(input("请输入x值下限:"))
xup = float(input("请输入x值上限:"))
while xup <= xlow:     #要求上限必须比下限值大
    print("请输入比下限%g大的值"%xlow)
    xup = float(input("请输入x值上限:"))

#循环处理
iter = 0      #迭代计数
while (xup - xlow) * (xup - xlow) > LIMIT: #满足终止条件前循环
    xmid = xlow + (xup - xlow) / 2      #计算新的中值点
    iter += 1                           #迭代计数加1
    if np.sign(f(xmid)) * np.sign(f(xlow)) < 0:    #中点与下限对应函数异号
        xup = xmid                      #更新xup
    else:                               #中点函数值为负
        xlow = xmid                     #更新xlow
    print("{:.15g},{:.15g},{:.15g}".format(iter,xlow, xup))
```

All inputs must be validated boundaries

Only by data variablization and process generalization, the program can be expanded

Error analysis:

Bisection error

Theorem 4.1 (Bisection Convergence and Error) Let $[a_0, b_0] = [a, b]$ be the initial interval, with $f(a)f(b) < 0$. Define the approximate root as $x_n = (b_{n-1} + a_{n-1})/2$. Then there exists a root $p \in [a, b]$ such that

$$|x_n - p| \leq \left(\frac{1}{2}\right)^n (b - a).$$

Moreover, to achieve an accuracy of

$$|x_n - p| \leq \varepsilon,$$

It suffices to take

$$n \geq \frac{\log(b-a) - \log \varepsilon}{\log 2}$$

Bisection error

Error analysis:

The 1st iteration: $x_0 = \dfrac{a_0 + b_0}{2}$ **Error:** $|x_0 - p| \le \dfrac{b_0 - a_0}{2}$

The 2nd iteration $x_1 = \dfrac{a_1 + b_1}{2}$ **Error:** $|x_1 - p| \le |x_1 - a_1|$

$$= \frac{a_1 + b_1}{2} - a_1$$

$$= \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2^2}$$

The *n*-th iteration, error of x_n : $|x_n - p| \le \dfrac{b-a}{2^{n+1}}$

Determine iterations ($n+1$) for given ε :

$$\frac{b-a}{2^{n+1}} < \varepsilon \quad \Rightarrow \quad n+1 \ge \left\lceil \frac{\ln(b-a) - \ln\varepsilon}{\ln 2} \right\rceil$$

Example

Example 4.1: use bisection method to find root of equation $7x^5-13x^4-21x^3-12x^2+58x+3=0$ on interval $[1, 2]$, the absolute error of adjacent x is less than $\varepsilon=10^{-5}$, estimate iteration and compare it with actual iteration.

Solution: $f(x)=7x^5-13x^4-21x^3-12x^2+58x+3$

given $f(1)>0$ **and** $f(2)<0$, **so there exists a root of** f ***on the interval*** **[1, 2], Iteration can be calculated**

$$n+1\ge\left\lceil\frac{\ln(b-a)-\ln\varepsilon}{\ln 2}\right\rceil=\left\lceil\frac{\ln(2-1)-\ln 10^{-5}}{\ln 2}\right\rceil=17$$

Example

Example 4.1: use bisection method to find root of equation $7x^5 - 13x^4 - 21x^3 - 12x^2 + 58x + 3 = 0$ **on interval** $[1, 2]$, **the absolute error of adjacent x is less than** $\varepsilon = 10^{-5}$, **estimate iteration and compare it with actual iteration.**

Solution: $f(x) = 7x^5 - 13x^4 - 21x^3 - 12x^2 + 58x + 3$

1st iter: midpoint of [1, 2] $x_0 = \dfrac{(1+2)}{2} = 1.5$, **then** $f(x_0) < 0$

2nd iter: midpoint of [1,1.5] $x_1 = \dfrac{(1+1.5)}{2} = 1.25$, **then** $f(x_1) > 0$

.

.

.

Example

Table 4.1

n	a_n	b_n	x_n	$\lvert f(x_n)\rvert$	$\lvert x_n-x_{n-1}\rvert<\varepsilon$	$\lvert x_n-x_{n-1}\rvert/\lvert x_n\rvert<\varepsilon$
0	1.00000	2.00000	1.50000	20.53130	0.50000	0.33333
1	1.00000	1.50000	1.25000	5.35840	0.25000	0.20000
2	1.25000	1.50000	1.37500	6.59311	0.12500	0.09091
3	1.25000	1.37500	1.31250	0.34135	0.06250	0.04762
4	1.25000	1.31250	1.28125	2.58030	0.03125	0.02439
5	1.28125	1.31250	1.29688	1.13710	0.01563	0.01205
6	1.29688	1.31250	1.30469	0.40224	0.00781	0.00599
7	1.30469	1.31250	1.30859	0.03153	0.00391	0.00299
8	1.30859	1.31250	1.31055	0.15464	0.00195	0.00149
9	1.30859	1.31055	1.30957	0.06149	0.00098	0.00075
10	1.30859	1.30957	1.30908	0.01496	0.00049	0.00037
11	1.30859	1.30908	1.30884	0.00829	0.00024	0.00019
12	1.30884	1.30908	1.30896	0.00334	0.00012	9.32575e-005
13	1.30884	1.30896	1.3089	0.00248	6.10352e-005	4.66309e-005
14	1.30890	1.30896	1.30893	0.00043	3.05176e-005	2.33149e-005
15	1.30890	1.30893	1.30891	0.00102	1.52588e-005	1.16576e-005
16	1.30891	1.30893	1.30892	0.00030	7.62939e-006	5.82876e-006

Table 4.1

n	a_n	b_n	x_n	$\|f(x_n)\|$	$\|x_n-x_{n-1}\|<\varepsilon$	$\|x_n-x_{n-1}\|/\|x_n\|<\varepsilon$
0	1.00000	2.00000	1.50000	20.53130	0.50000	0.33333
1	1.00000	1.50000	1.25000	5.35840	0.25000	0.20000
2	1.25000	1.50000	1.37500	6.59311	0.12500	0.09091
3	1.25000	1.37500	1.31250	0.34135	0.06250	0.04762
4	1.25000	1.31250	1.28125	2.58030	0.03125	0.02439
5	1.28125	1.31250	1.29688	1.13710	0.01563	0.01205
6	1.29688	1.31250	1.30469	0.40224	0.00781	0.00599
7	1.30469	1.31250	1.30859	0.03153	0.00391	0.00299
14	1.30890	1.30896	1.30893	0.00043	3.05176e-005	2.33149e-005
15	1.30890	1.30893	1.30891	0.00102	1.52588e-005	1.16576e-005
16	1.30891	1.30893	$\boxed{1.30892}$	0.00030	$\boxed{7.62939e-006}$	5.82876e-006

Set stopping criterion is $|x_n - x_{n-1}| < \varepsilon$, given $\varepsilon < 10^{-5}$, solution is 1.30892. The actual iteration is 17, consistent with estimates.

Comparing the last three columns of Table 4.1, it shows the differences among three stopping criteria, where the value of $|f(x_n)|$ is not monotonically, while the other two stopping criteria are monotonically decreasing and similar to the change of error

Bisection Method Pros and Cons List

① **Always Convergent.**
② **Easy to Understand.**
③ **It is Fault Free (Generally).**

① **Rate of Convergence is Slow.**
② **Relies on Sign Changes.**
③ **Can't Detect Multiple Roots.**
④ **Requires a Lot of Effort.**

To find the root by bisection, it is best to find the root on the graph of $f(x)$. Or use a program to divide [a, b] into small intervals, and call the bisection procedure for each interval that satisfys $f(a_k)\cdot f(b_k) < 0$ to find multiple roots on the interval [a, b], without requiring $f(a)\cdot f(b) < 0$.

Contents

Numerical solutions of nonlinear equations

- Bisection method
- Newton-Raphson Method

Newton-Raphson Method

Replaced the general function with the simple function (a straight line) — Taylor expansion

Given $\boldsymbol{x_0 \approx p}$, we expand $\boldsymbol{f(x)}$ in a Taylor series about $\boldsymbol{x_n}$:

$$f(x) = f(x_n) + f'(x_n)(x - x_n) + \boxed{\frac{f''(\xi)}{2!}(x - x_n)^2}$$

, where ξ is between x_n and x

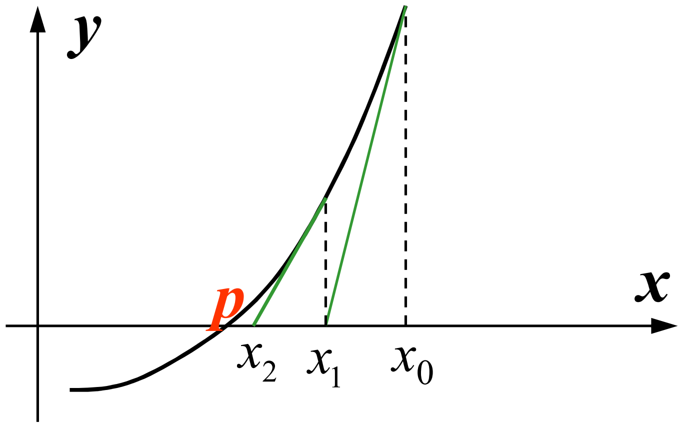
Remove items of order 2 and above (linearization), then:

$$f(x) = f(x_n) + f'(x_n)(x - x_n)$$

Assume $f'(x_n) \neq 0$, **replace** $\boldsymbol{x_{n+1}}$ **by** $\boldsymbol{x}$, **set** $\boldsymbol{f(x) = 0}$ **to get iterative formula**

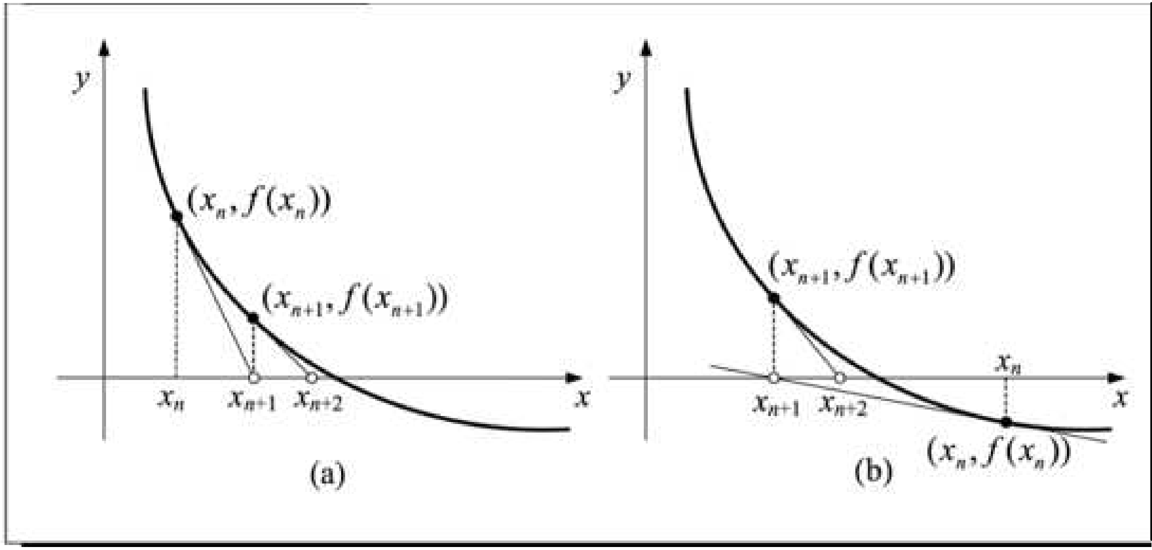
$$x = x_n - \frac{f(x_n)}{f'(x_n)} \quad \Rightarrow \quad x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Geometrical Interpretation of Newton-Raphson Method



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

As long as $f \in C^1$, keep $f'(x_n) \neq 0$, and $\lim_{n \to 0} x_n = p$, then p *is the root of* f.



◆ **The n^th^ iteration**: Drawing a tangent at $[x_n, f(x_n)]$, which cuts the x-axis at x_{n+1}

◆ **The (n+1)^th^ iteration**: Drawing a tangent at $[x_{n+1}, f(x_{n+1})]$, which cuts the x-axis at x_{n+2}

◆ *And so on until the iteration stops*

Sufficient condition for the convergence of the Newton-Raphson method

Theorem4.1

Assume $f \in C^2[a, b]$, that is f has a second-order continuous derivative on $[a, b]$, if

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

There is a root

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

The resulting sequence is monotonically bounded and guarantees convergence

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0)f''(x) > 0$

Then the sequence $\{ x_n \}$ generated by Newton-Rahpson method converges to the unique solution of $f(x)$ on $[a, b]$, and the convergence order is 2.

The effect of the initial guess on convergence in Newton-Raphson method

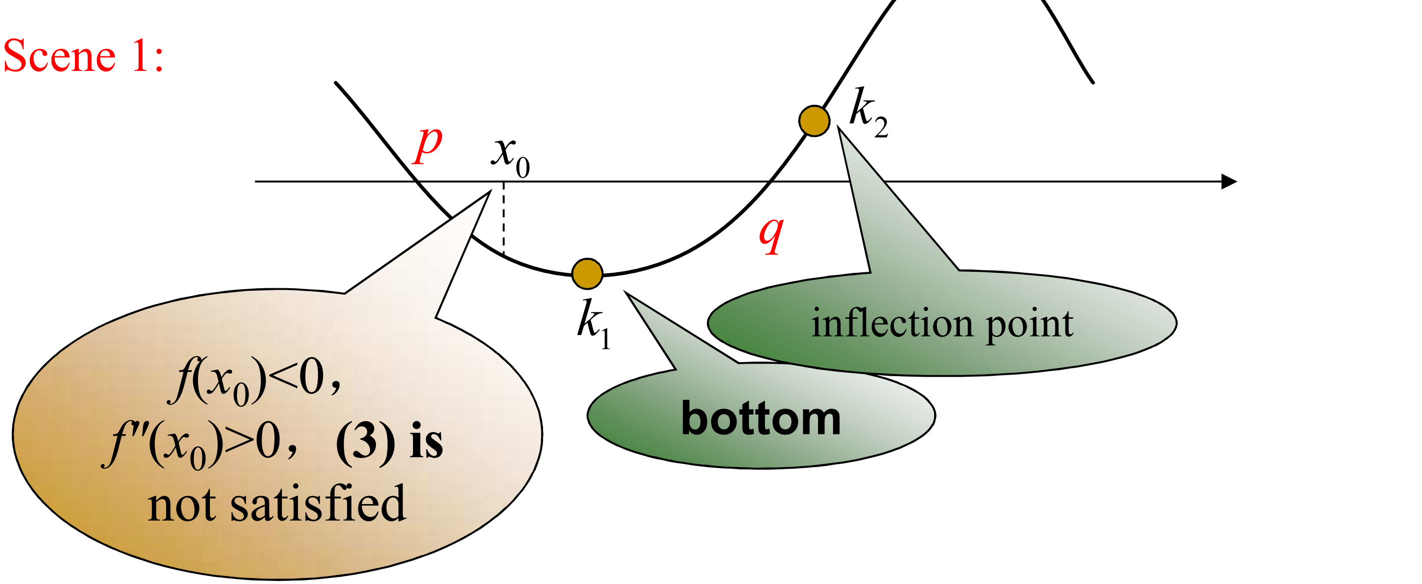
The convergence depends on initial guess of x_0

Theorem 4.1 gives another two conditions :

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

Scene 1:



The final convergence shows that the theorem is sufficient and not necessary

The effect of the initial guess on convergence in Newton-Raphson method

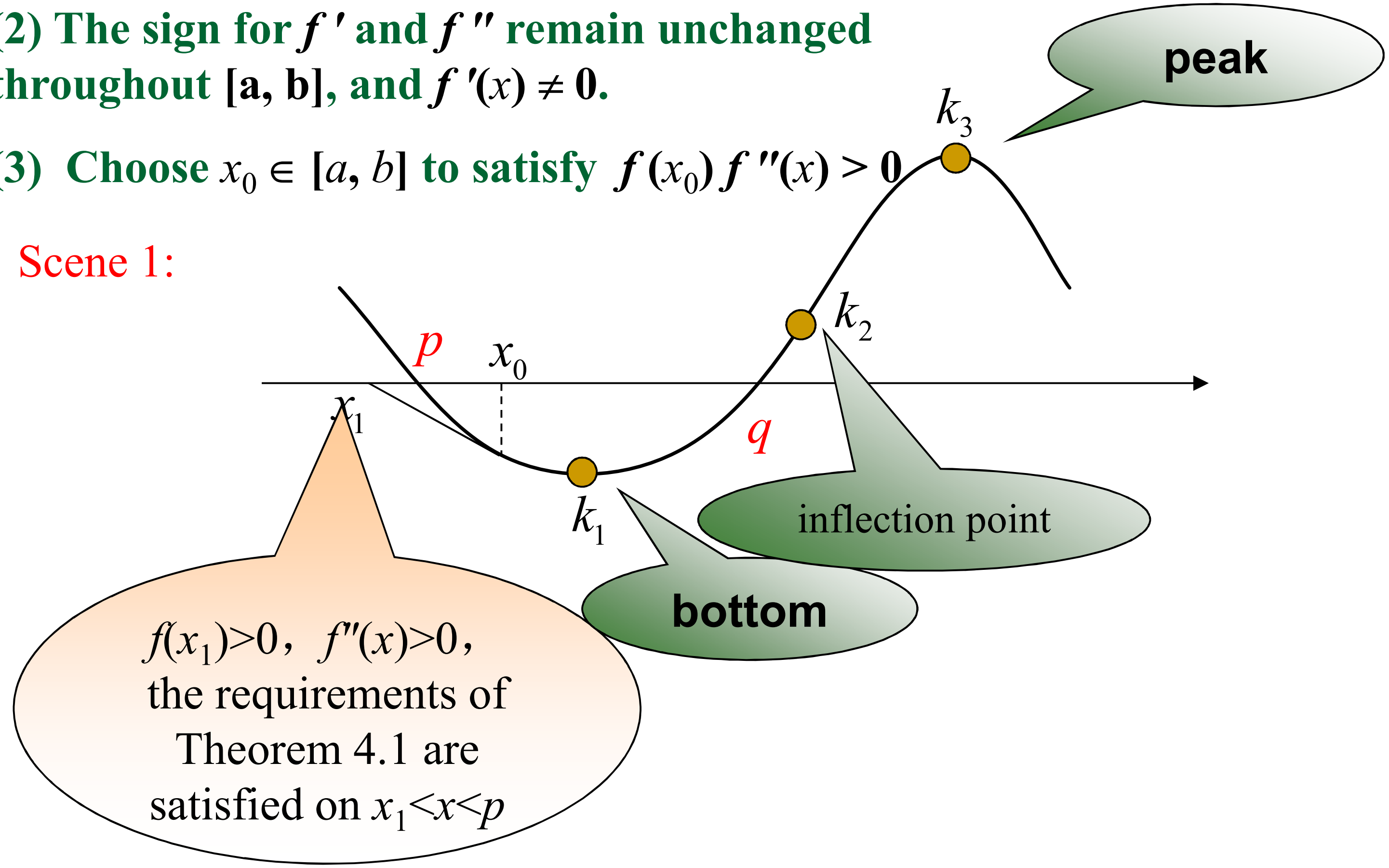
The convergence depends on initial guess of x_0

Theorem 4.1 gives another two conditions :

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

Scene 1:



The effect of the initial guess on convergence in Newton-Raphson method

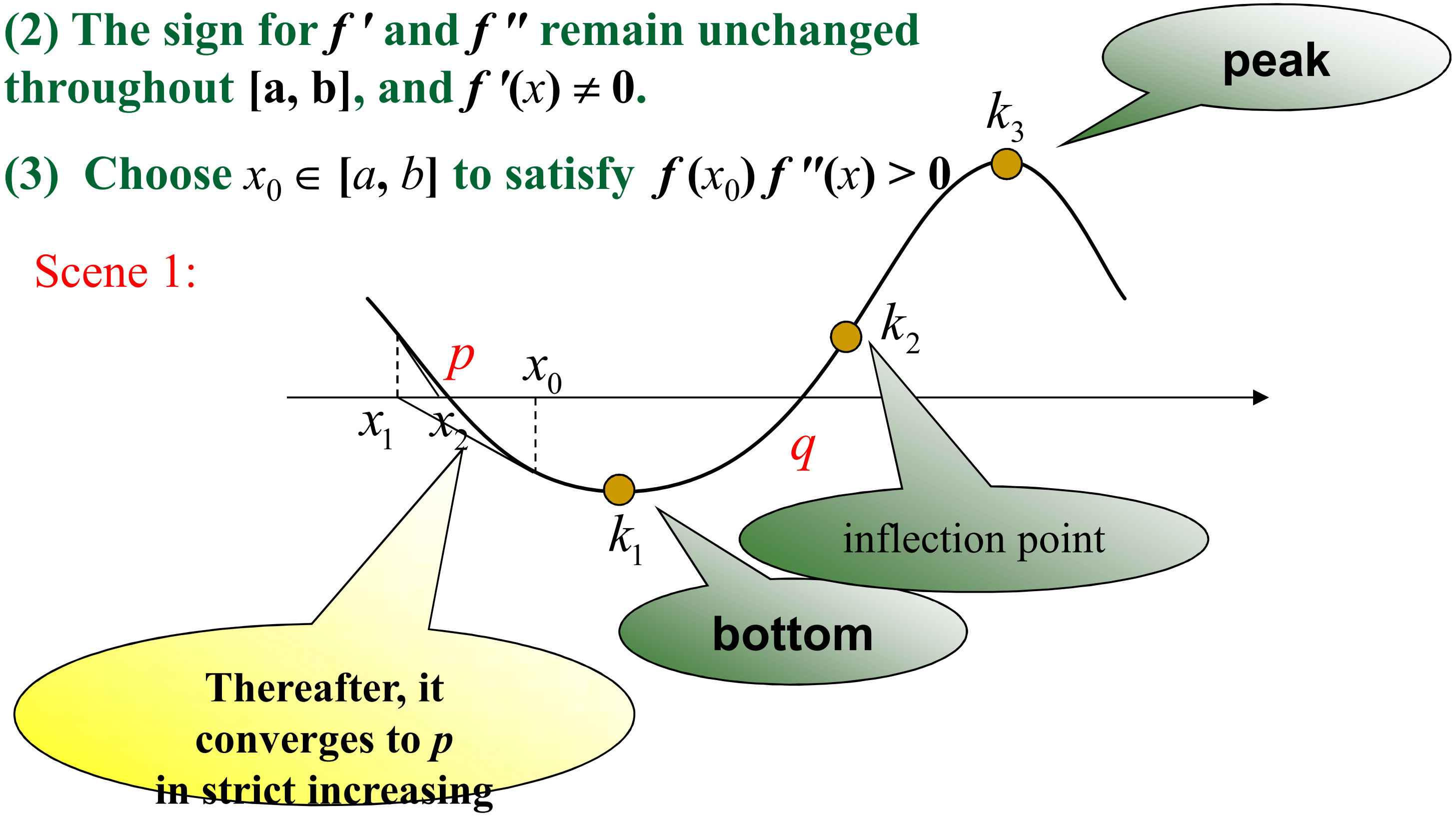
The convergence depends on initial guess of x_0

Theorem 4.1 gives another two conditions :

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

Scene 1:



The final convergence shows that the theorem is sufficient and not necessary

The effect of the initial guess on convergence in Newton-Raphson method

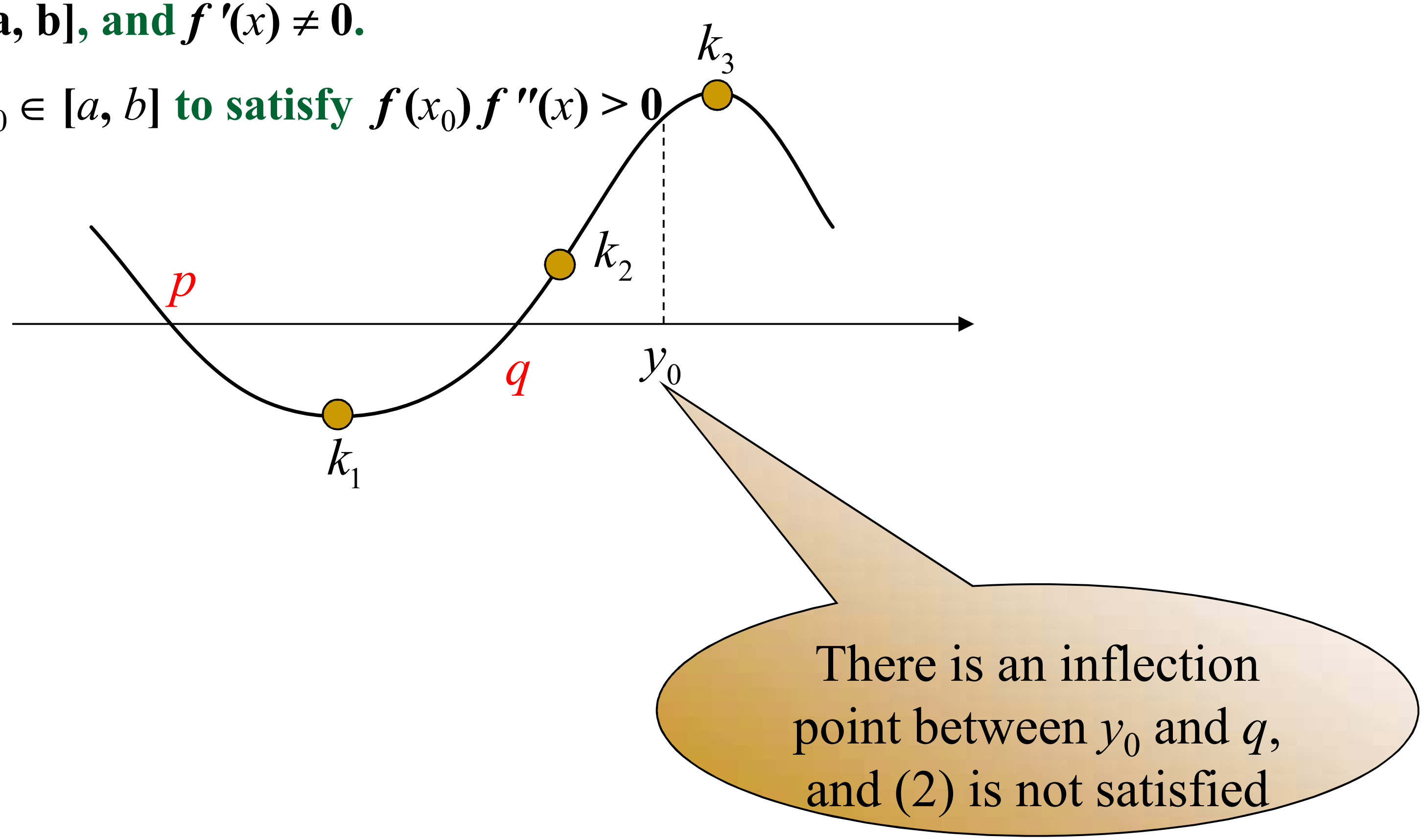
The convergence depends on initial guess of x_0

Theorem 4.1 gives another two conditions :

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

Scene 2:



The effect of the initial guess on convergence in Newton-Raphson method

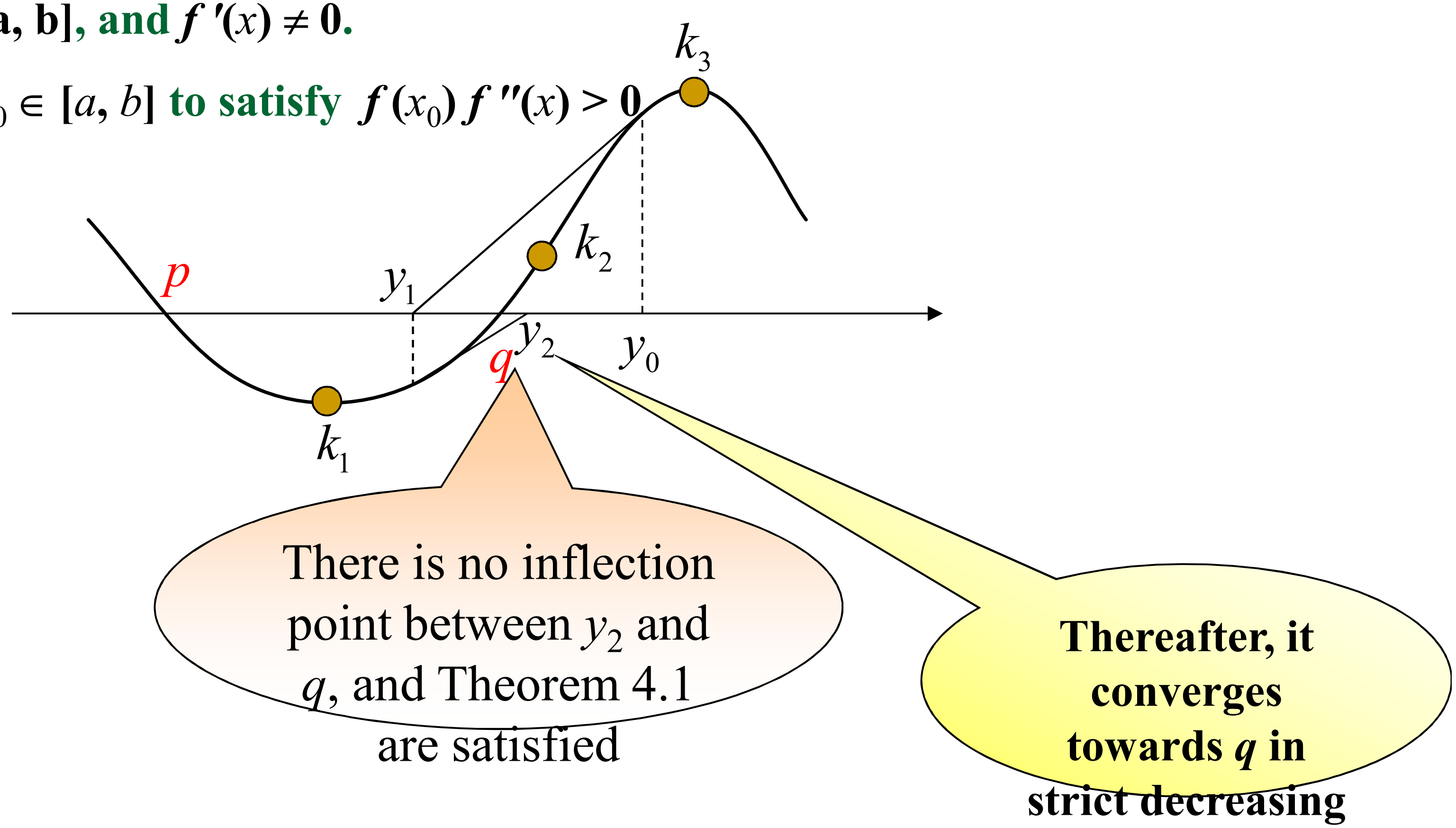
The convergence depends on initial guess of x_0

Theorem 4.1 gives another two conditions :

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

Scene 2:



The effect of the initial guess on convergence in Newton-Raphson method

The convergence depends on initial

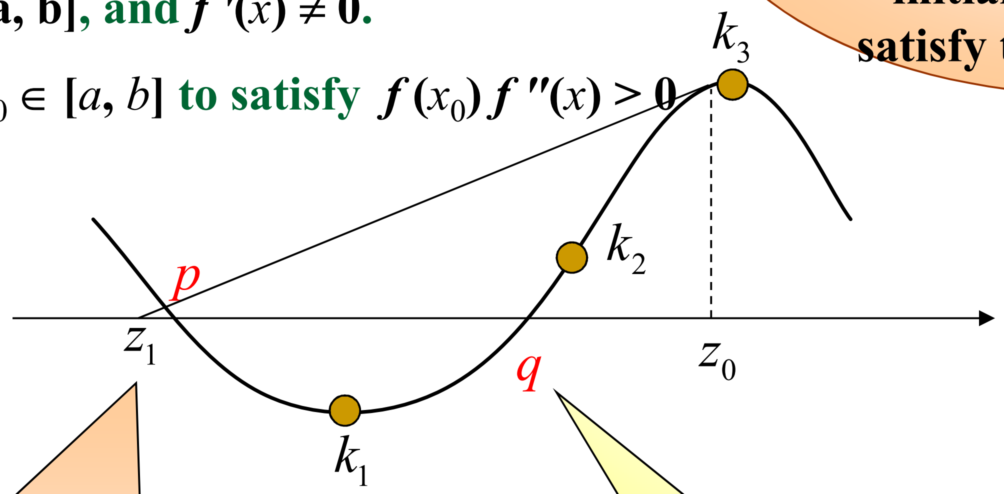
Theorem 4.1 gives another two cond

(2) The sign for f' and f'' remain unchanged throughout [a, b], and $f'(x) \neq 0$.

(3) Choose $x_0 \in [a, b]$ to satisfy $f(x_0) f''(x) > 0$

In order to ensure that the series can converge to the solution on the interval , the initial guess should satisfy the theorem 4.1.

Scene 3:



The first z_1 is far from the nearest root q and closes to another distant root p

Eventually converges, but not to the expectable root q

Example

- **Example 4.2 Use Newton method to solve equations**

$$7x^5-13x^4-21x^3-12x^2+58x+3=0\,,\quad x\in[1,2]$$

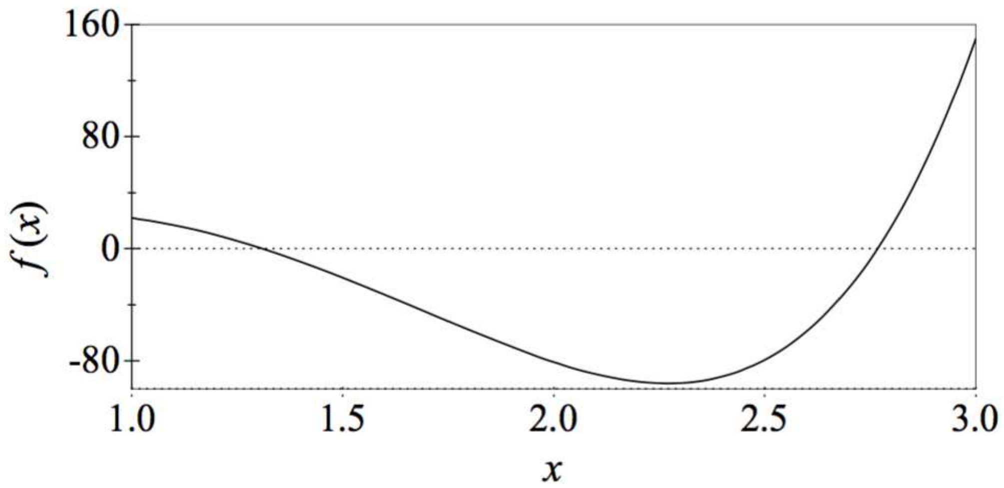
Solution: Let $f(x)=7x^5-13x^4-21x^3-12x^2+58x+3$, we have $f'(x)=35x^4-52x^3-63x^2-24x+58$, since $f(1)>0$ and $f(2)<0$, there is a root on [1, 2]. But $f'(1)f''(1)>0$ and $f'(2)f''(2)<0$, they do not meet the requirements of the theorem , and therefore it cannot be sure that the sequence generated by any initial guess on [1,2] in Newton's method converges to the solution.

Example

- **Example 4.2 Use Newton method to solve equations**

$$7x^5-13x^4-21x^3-12x^2+58x+3=0\,,\quad x\in[1,2]$$

Consider narrowing the definition domain, draw the graph on the interval [1, 3], take the interval [1, 1.5], we know that the solution is still in this region, and $f'f''>0$ on the interval [1,1.5], so the Theorem 4.1 is satisfied. Choose the initial guess $x_0=1.5$, we stop when $|x_n-x_{n-1}|<\varepsilon=10^{-5}$, the process is listed as bellows:



Example

- **Example 4.2 Use Newton method to solve equations**

$$7x^5 - 13x^4 - 21x^3 - 12x^2 + 58x + 3 = 0\,, \quad x \in [1,2]$$

n	$x_0 = 1$	$x_0 = 1.5$	$x_0 = 2$	$x_0 = 2.5$
0	1.00000	1.50000	2.00000	2.50000
1	1.47826	1.32610	1.17347	2.99921
2	1.32319	1.30914	1.32805	2.82321
3	1.30907	1.30892	1.30919	2.7712
4	1.30892	1.30892	1.30892	2.76688
5	1.30892		1.30892	2.76685

The theorem is not satistied on [1,p]

The theorem is not satistied on [p,2]

Converges to another solution outside the domain

Converge to the same solution