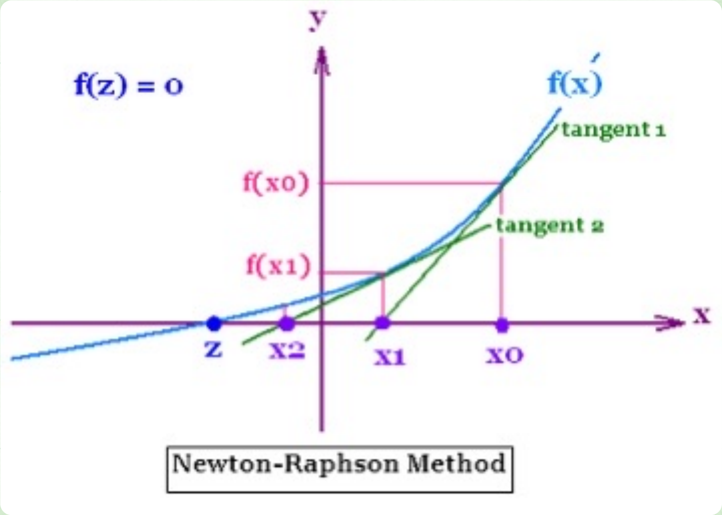
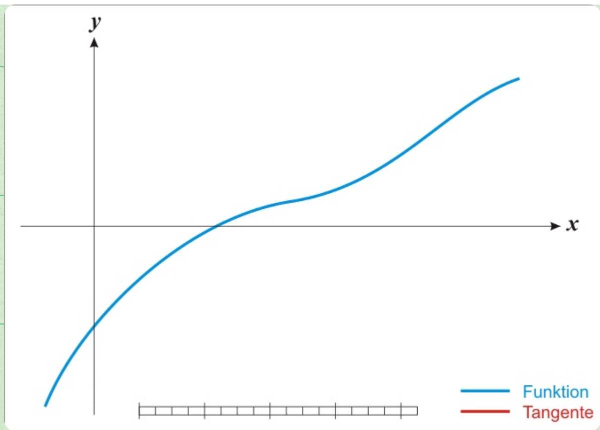
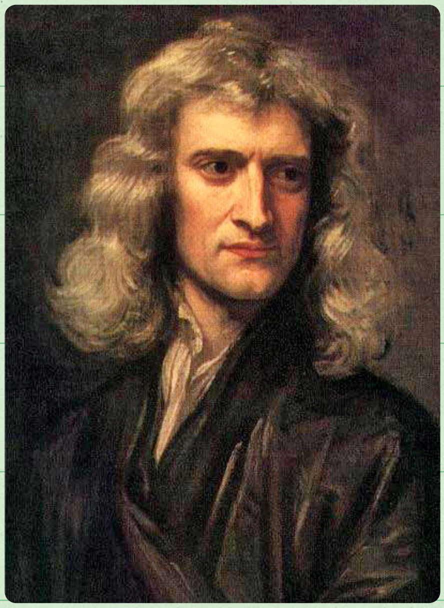
NEWTON - RAPHSON'S METHOD





This method is one of the most powerful and well-known numerical methods for solving a root-finding problem

$$f(x) = 0$$



Issac Newton

(1641 - 1727)

Joseph Raphson

(1648 - 1715)

😌. Suppose $f \in C^2([a,b])$.

- Let $p_0 \in [a,b]$ be an approximation of p s.t.

$$\begin{cases} f'(p_0) \neq 0 \\ |p - p_0| \text{ is small} \end{cases}$$

- Using the 1^{st} Taylor polynomial for $f(x)$ expanded about p_0 & evaluated at $x = p$ we get

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

where $\xi(p)$ is between p and p_0.

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

Solving this for p gives us

$$p \approx p_0 - \frac{f(p_0)}{f'(p_0)} = p_1$$

😉 Oh, I get an idea !

Construct a sequence $\{p_n\}_{n=0}^{\infty}$ by

$$\boxed{p_n = p_{n-1} - \frac{f(p_{n-1})}{f'(p_{n-1})} \quad \text{for } n \geq 1}$$

p_0: initial guess

Newton's Illustration

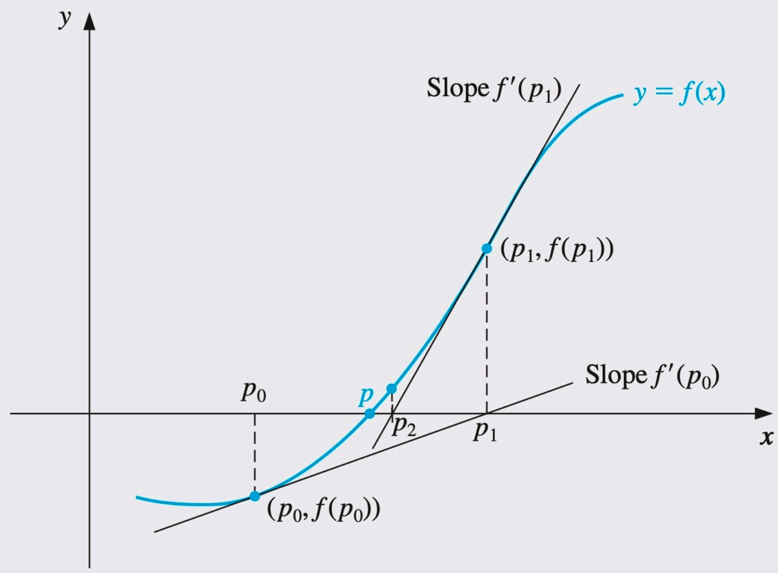


Figure: Figure 2.7

- Start with p_0, p_1 = the x-intercept of tangent line to $y = f(x)$ at $(p_0, f(p_0))$

$\to y = f(p_0) + f'(p_0)(x - p_0)$

Set $x = p_1$ & $y = 0$

we get $0 = f(p_0) + f'(p_0)(p_1 - p_0)$

So $\boxed{p_1 = f(p_0) - \dfrac{f(p_0)}{f'(p_0)}}$

- Similarly, $p_2 =$ the x- intercept of tangent line to $y = f(x)$ at $(p_1, f(p_1))$.

...

Example: Solve $2x^2 - 5x + 3 = 0$ numerically

(Analytical solutions: $x = 1.5$ & $x = 1$).

$$f(x) = 2x^2 - 5x + 3$$

$$f'(x) = 4x - 5$$

The formula that will be "plugged" in the code becomes

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

$$\boxed{x_{new} = x - \frac{2x^2 - 5x + 3}{4x - 5}}$$

```
In [3]: x = 0
for iteration in range (1,101):
    xnew = x - (2*x**2-5*x+3)/(4*x-5)
    if abs(xnew-x) < 0.000001:
        break
    x = xnew
print('The root %0.5f' % xnew)
print('The number of iterations : %d' %itera
```

```
The root 1.00000
The number of iterations : 7
```

```
In [4]: x = 2
for iteration in range (1,101):
    xnew = x - (2*x**2-5*x+3)/(4*x-5)
    if abs(xnew-x) < 0.000001:
        break
    x = xnew
print('The root %0.5f' % xnew)
print('The number of iterations : %d' %iteration)
```

```
The root 1.50000
The number of iterations : 6
```

Compare this with $\underbrace{\text{simple-iteration}}_{\text{fixed-iteration}}$ method.

→ More efficient ! Awesome 😊

◎ Example: Solve $\cos x - x = 0$ numerically.

```
In [5]: from math import *
x = 0
for iteration in range (1,101):
    xnew = x - (cos(x)-x)/(-sin(x)-1)
    if abs(xnew-x) < 0.000001:
        break
    x = xnew
print('The root %0.5f' % xnew)
print('The number of iterations : %d' %iteration)
```

```
The root 0.73909
The number of iterations : 5
```

Try to run the code with different initial guesses. 😊

Convergence Using Newton's Method

Theorem

- Suppose $f \in C^2([a,b])$
- $\exists p \in (a,b)$ s.t $f(p) = 0$ and $f'(p) \neq 0$

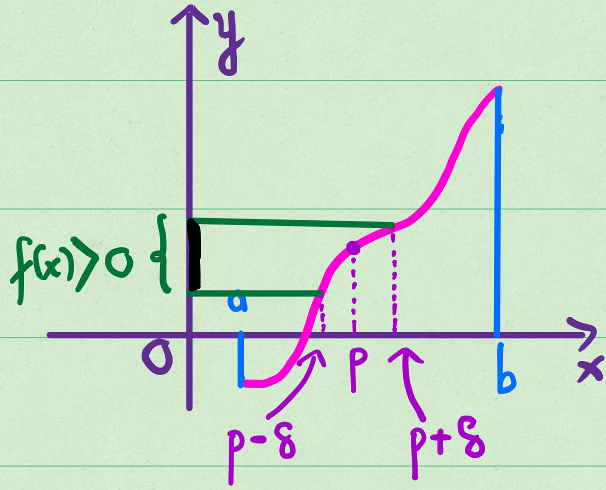
$\Rightarrow \exists \delta > 0$ s.t. Newton's method generates a sequence $\{p_n\}_{n=0}^{\infty}$ converging to p for any initial approximation $p_0 \in [p-\delta, p+\delta]$.

Proof

- Set $g(x) = x - \dfrac{f(x)}{f'(x)}$
- $\{p_n\}_{n=0}^{\infty}$ is then given by

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

Lemma Let $f \in C([a,b])$ and $p \in (a,b)$. If $f(p) \neq 0$, then $\exists \delta > 0$ s.t. $f(x) \neq 0$ for all $x \in [p-\delta, p+\delta] \subset [a,b]$



Assume that $f(p) > 0$.

It follows from the continuity of f at $x = p$ that for $\varepsilon = \frac{1}{2} f(p) > 0$

$$\exists \delta > 0 \text{ s.t. } \underbrace{|f(x) - f(p)| < \varepsilon}_{} \quad \forall x \in [p-\delta, p+\delta]$$

$$\Downarrow$$

If $f(p) = 0$, then $|f(x)| < k$ $\forall x \in [p-\delta, p+\delta]$ for any given $k > 0$

$$f(x) - f(p) > -\varepsilon$$

$$f(x) > f(p) - \varepsilon = \frac{1}{2} f(p) > 0$$ 😊

◎ Goal: Let $k \in (0,1)$, we find an interval $[p-\delta, p+\delta]$ for which

1) $g : [p-\delta, p+\delta] \longrightarrow [p-\delta, p+\delta]$ ✓

2) $|g'(x)| \leq k < 1 \quad \forall x \in (p-\delta, p+\delta)$ ✓

- Since f' is continuous and $f'(p) \neq 0$, we can find $\delta_1 > 0$ s.t. (By the lemma)

$$f'(x) \neq 0 \quad \forall x \in [p-\delta_1, p+\delta_1] \subseteq [a,b]$$

• So g is defined and continuous on $[p-\delta_1, p+\delta_1]$.

Also

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

$\forall x \in [p-\delta_1, p+\delta_1]$

• Because $f(p) = 0$, we get

$$g'(p) = 0$$

• Again as g' is continuous and $0 < k < 1$, it then follows from the lemma that

$\exists \delta > 0$ s.t. $0 < \delta < \delta_1$ and

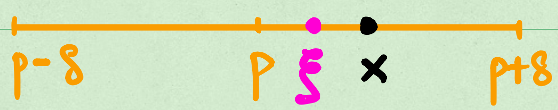
$|g'(x)| \leq k \quad \forall x \in [p-\delta, p+\delta] \subset [a, b]$.

• It remains to verify that

$$g : [p-\delta, p+\delta] \longrightarrow [p-\delta, p+\delta]$$

i.e. pick any $\underbrace{x \in [p-\delta, p+\delta]}_{|x-p| \leq \delta} \overset{\text{argue}}{\Longrightarrow} \underbrace{g(x) \in [p-\delta, p+\delta]}_{|g(x)-p| \leq \delta}$

Employing the Mean Value Theorem gives us some ξ between x and p for which



$$g(x) - g(p) = g'(\xi)(x-p)$$

So $|g(x) - p| = |g'(\xi)||x-p| \leq k|x-p| \leq |x-p| \leq \delta$

which justifies our claim.

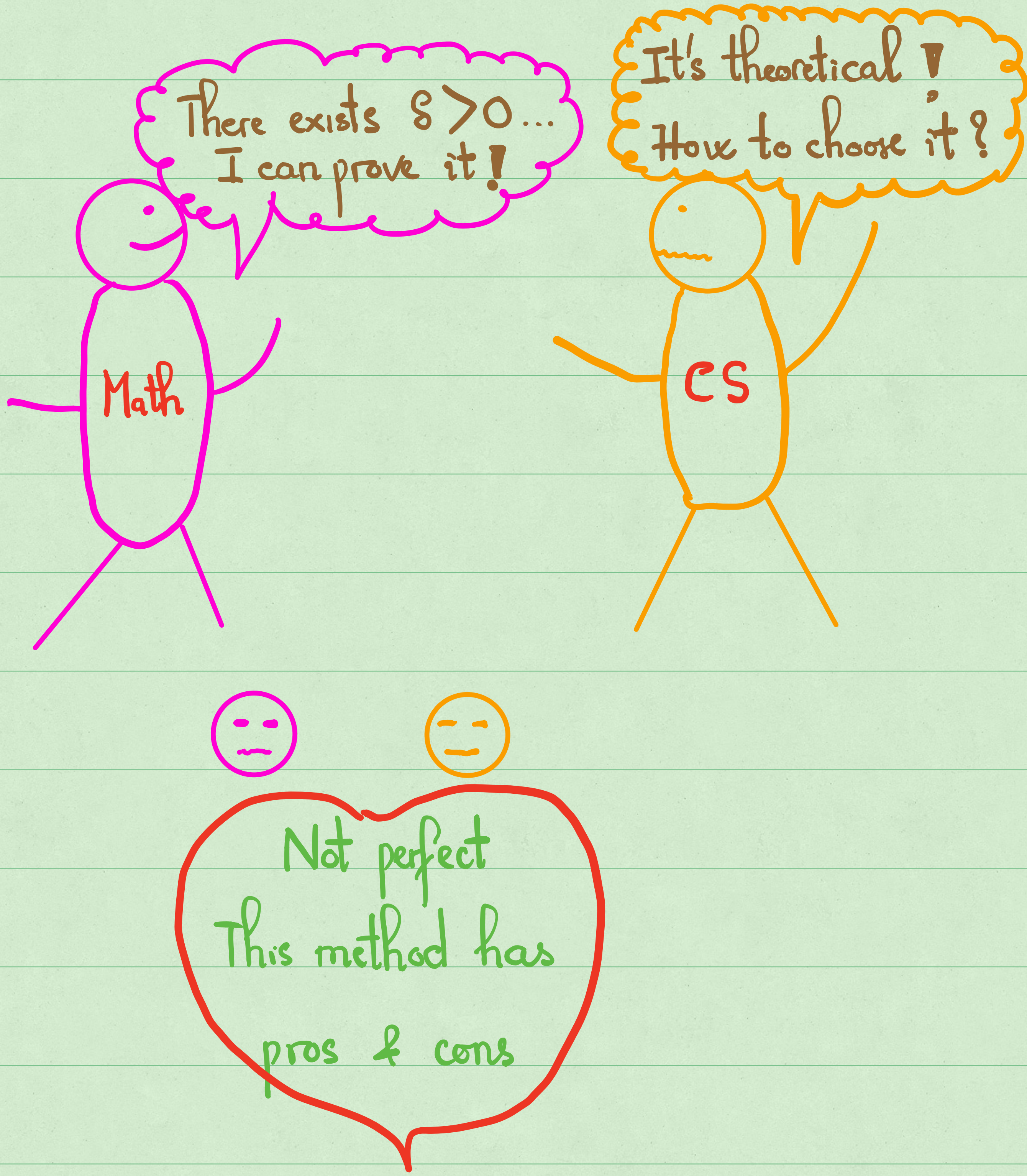
The conclusion of the theorem follows from the fixed-point theorem.

Remark:

- Under some reasonable assumptions, selecting a good initial approximation $\longrightarrow$ makes Newton's method convergent.

- The constant k indicating the speed of convergence of the method decreases to 0 as the procedure continues.

→ This result is very important for the theory of Newton's method, but it is somehow not practical because it does not tell us how to determine δ.



Example: Solve $x^2 + \cos^2 x - 4x = 0$ numerically.

$$f(x) = x^2 + \cos^2 x - 4x$$

$$f'(x) = 2x - 2\cos x \sin x - 4$$

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

$$x_{new} = x - \frac{x^2 + \cos^2 x - 4x}{2x - 2\cos x \sin x - 4}$$

```python
from math import *
x = 0
for iteration in range (1,101):
    xnew = x - (x**2+ cos(x)**2 - 4*x)/(2*(x-cos(x)*sin(x)-2))
    if abs(xnew-x) < 0.000001:
        break
    x = xnew
print('The root %0.5f' % xnew)
print('The number of iterations : %d' %iteration)
```

```
The root 0.25032
The number of iterations : 3
```

Run the code with different initial values

$x = 2, -2, 1, 10$