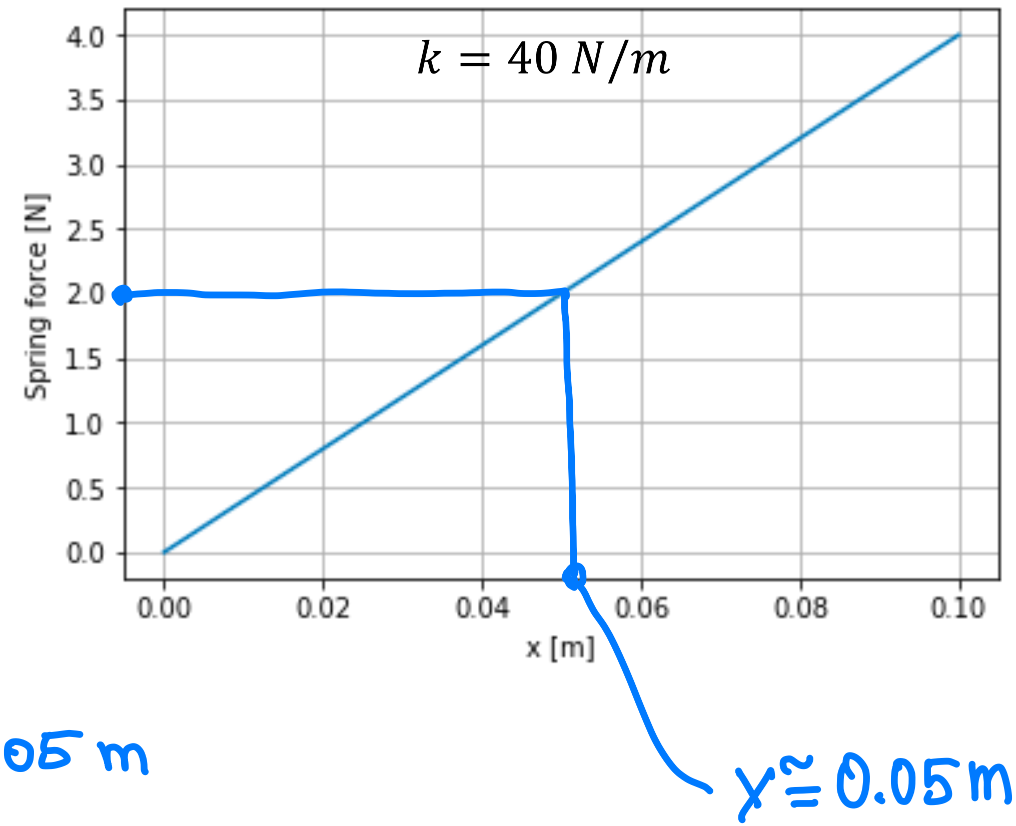
Nonlinear Equations

How can we solve these equations?

- Spring force:

$$F = k\, x$$

What is the displacement when $F = 2\text{N}$?



$$F = kx$$

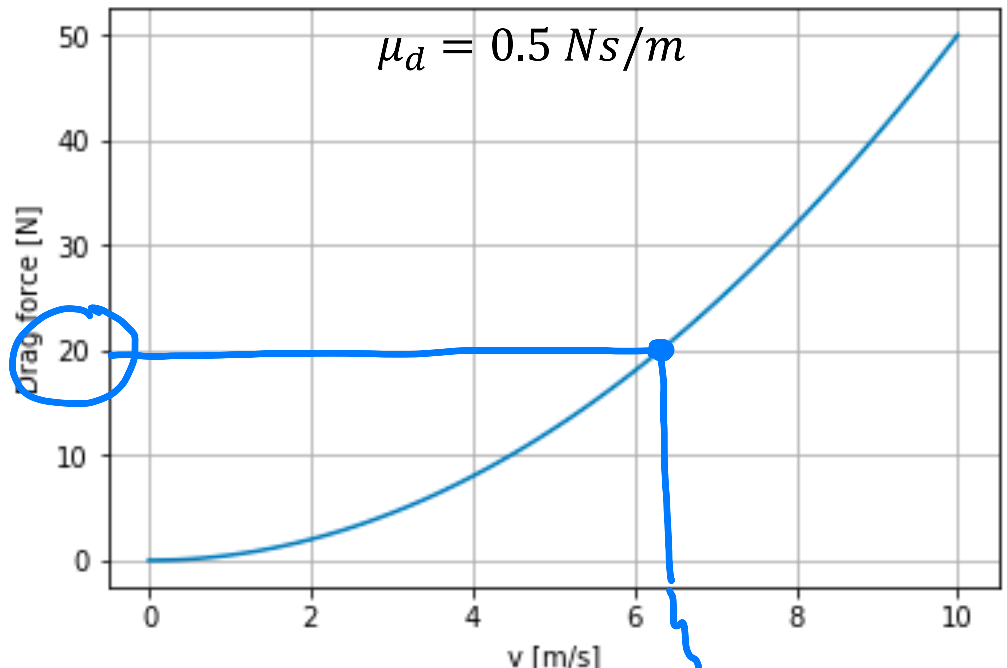
$$x = \frac{F}{k} = \frac{2N}{40N/m} = 0.05\,m$$

How can we solve these equations?

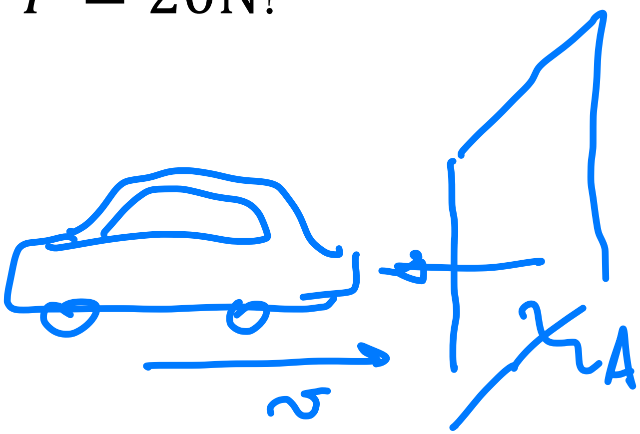
- Drag force:

$$F = 0.5\, C_d\, \rho\, A\, v^2 = \mu_d\ v^2$$

What is the velocity when $F = 20\text{N}$?



$v \cong 6.5\ m/s$



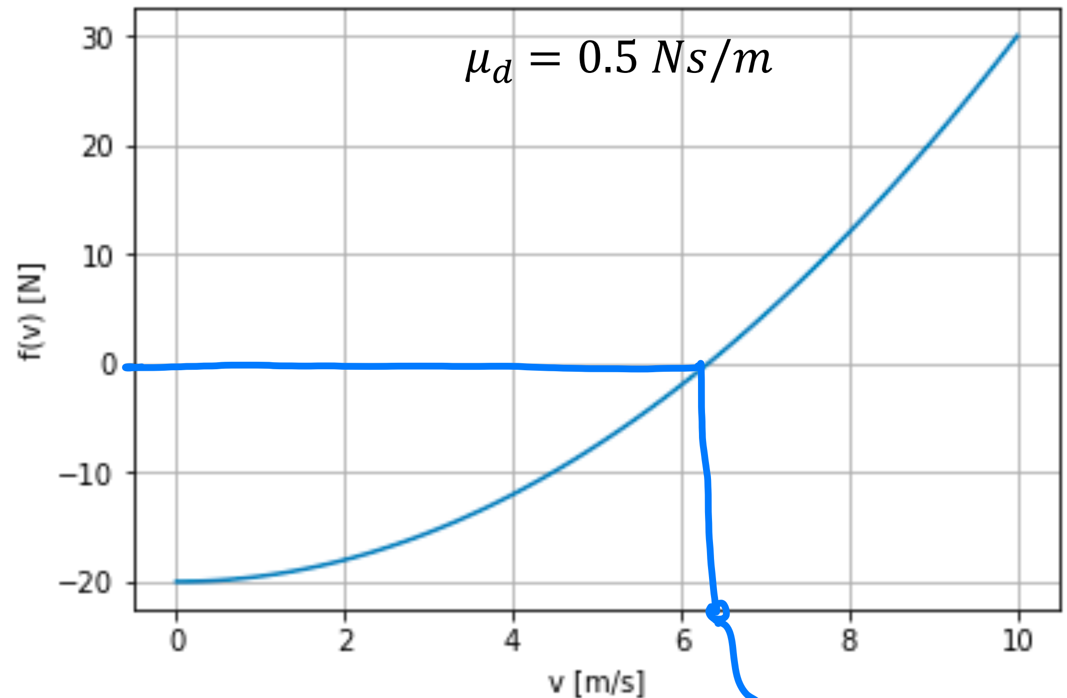
$F = \mu v^2 \longrightarrow v^2 = \dfrac{F}{\mu} \longrightarrow v = \pm\sqrt{F/\mu} \Rightarrow v \cong 6.3\ m/s$

$F = \mu v^2 \rightarrow \underbrace{F - \mu v^2}_{f(v)} = 0$

$f(v) = 0$

$$f(v) = \mu_d \ v^2 - F = 0$$

Find the root (zero) of the nonlinear equation $f(v)$



$v \cong 6.3\,m/s$

Nonlinear Equations in 1D

$\boxed{f(v) = 0}$

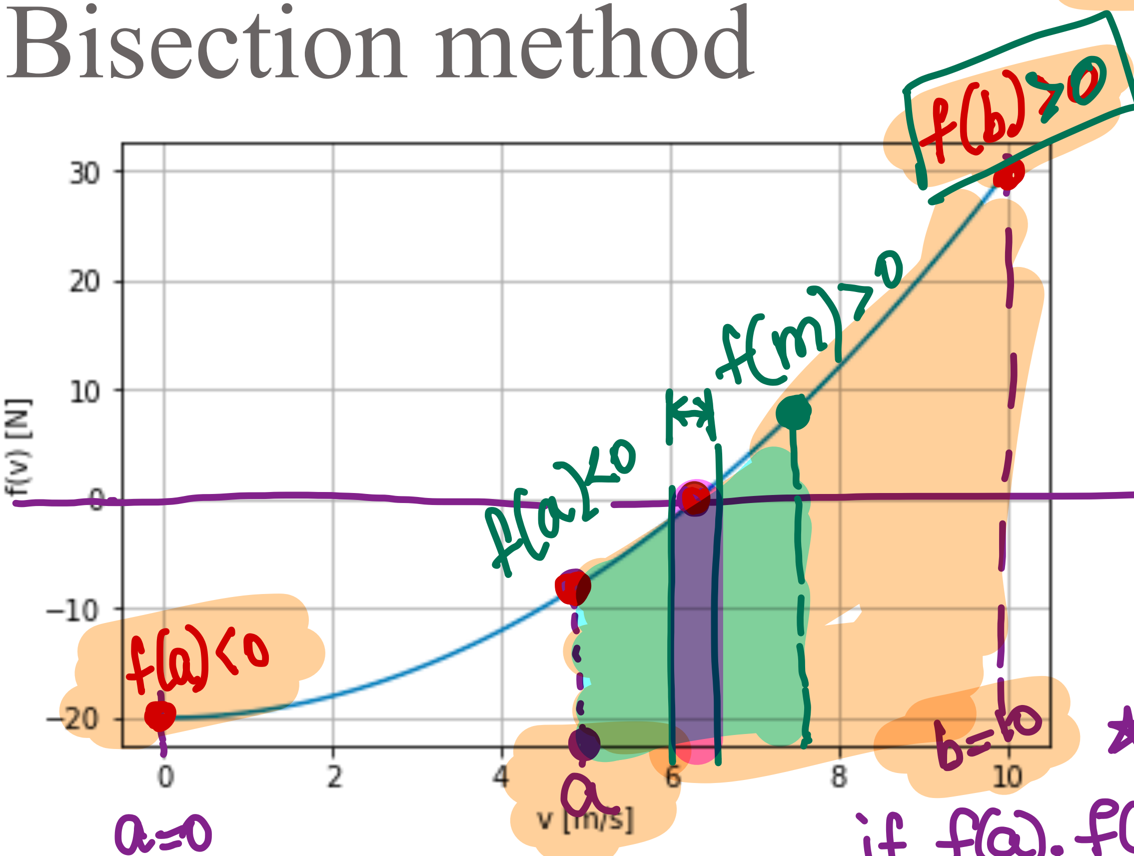
root of f

Goal: Solve $f(x) = 0$ for $f: \mathcal{R} \rightarrow \mathcal{R}$

numerical → iterative

Often called **Root Finding**

Bisection method



a=0

★ Define interval that has the root

$\boxed{\text{interval} : [a, b]}$

$a = 0$; $b = 10$

$t_0 = |b - a| = 10$

★ Define midpoint:

$m = \frac{b+a}{2} = 5$

★ Check the signs!

<u>check interval</u>:

$\text{sign}(f(a)) \neq \text{sign}(f(b))$

$f(a) \cdot f(b) < 0$

if $f(a) \cdot f(m) > 0$:

$t_{new} = [m, b]$

$a = m$

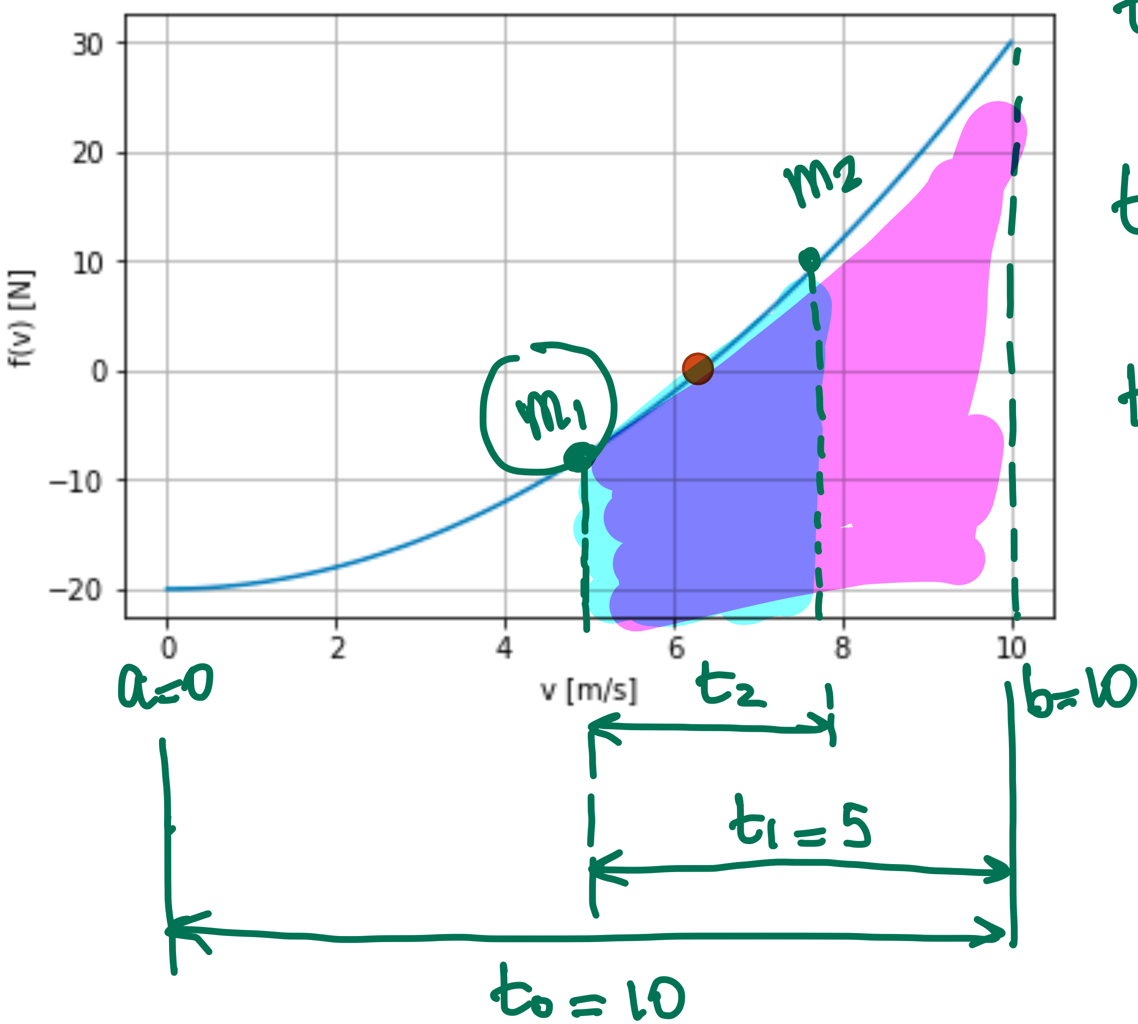
else :

$t_{new} = [a, m]$

$b = m$

interval gets smaller

Bisection method



$t_0 = |b-a| = 10$

$t_1 = \frac{|b-a|}{2} = \frac{t_0}{2}$

$t_2 = \frac{t_1}{2} = \frac{t_0}{2.2}$

$t_3 = \frac{t_2}{2} = \frac{t_0}{8}$

$\vdots$

$$\boxed{t_k = \frac{t_0}{2^k}}$$

★ every iteration, the interval is divided by 2 !

Convergence

An iterative method **converges with rate** r if:

$$\lim_{k\to\infty} \frac{||e_{k+1}||}{||e_k||^r} = C, \quad 0 < C < \infty \qquad r = 1\text{: linear convergence}$$

Linear convergence gains a constant number of accurate digits each step (and $C < 1$ matters!)

For example: Power Iteration

$$\lim_{k\to\infty} \frac{||e_{k+1}||}{||e_k||^{r=1}} = \boxed{\left|\frac{\lambda_2}{\lambda_1}\right|} = \text{constant} = C \Rightarrow \text{linear convergence}$$

$\star\ \lambda_2 \sim \lambda_1 \rightarrow \text{constant} \approx 1 \rightarrow$ slow convergence

$\lambda_1 = \alpha\lambda_2 \longrightarrow C = \frac{1}{\alpha} \longrightarrow$ faster convergence as α increases

Convergence

An iterative method **converges with rate** r if:

Power method

$$\lim_{k\to\infty} \frac{||e_{k+1}||}{||e_k||^r} = C, \qquad 0 < C < \infty$$

- $r = 1$: linear convergence
- $r > 1$: superlinear convergence $1 < r < 2$
- $r = 2$: quadratic convergence

Linear convergence gains a constant number of accurate digits each step (and $C < 1$ matters!)

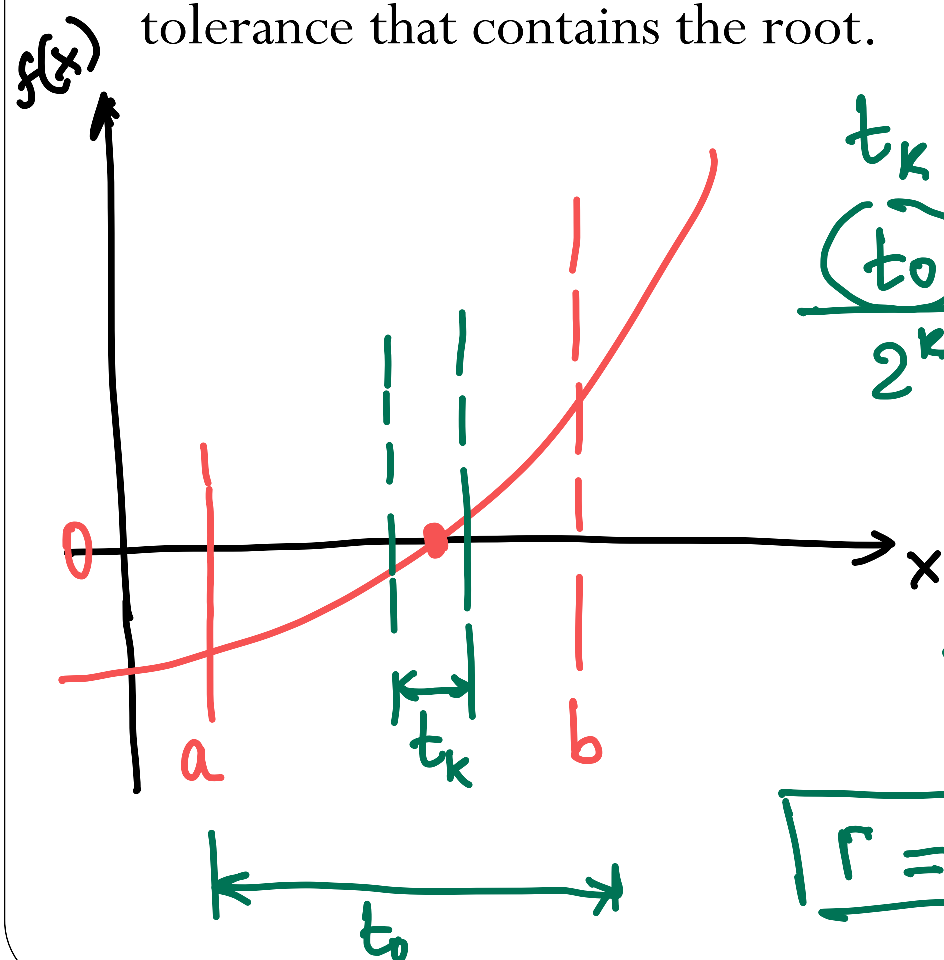
Quadratic convergence doubles the number of accurate digits in each step (however it only starts making sense once $||e_k||$ is small (and C does not matter much)

Convergence

x^* is the root

$\cancel{x_k} \longrightarrow t_k$ $\cancel{\text{error} = x^* - x_k}$

- The bisection method does not estimate x_k, the approximation of the desired root x. It instead finds an interval smaller than a given tolerance that contains the root.



$t_k < tol \longrightarrow$ stop

$$\frac{t_0}{2^k} < tol$$

$$\text{error} = t_k$$

$$\frac{|e_{k+1}|}{|e_k|^r} = \frac{|b-a|/2^{k+1}}{|b-a|/2^k} = \frac{1}{2} = C$$

$\boxed{r = 1}$ $\boxed{C = \frac{1}{2}}$ $\Rightarrow$ linear convergence!

Example:

in general: $t_k < tol$

$$\frac{|b-a|}{2^k} < tol$$

$$2^k > \frac{|b-a|}{tol}$$

$$k > \log_2\left(\frac{|b-a|}{tol}\right)$$

Consider the nonlinear equation

$$f(x) = 0.5x^2 - 2$$

and solving $\mathrm{f(x)} = 0$ using the Bisection Method. For each of the initial intervals below, how many iterations are required to ensure the root is accurate within 2^{-4}?

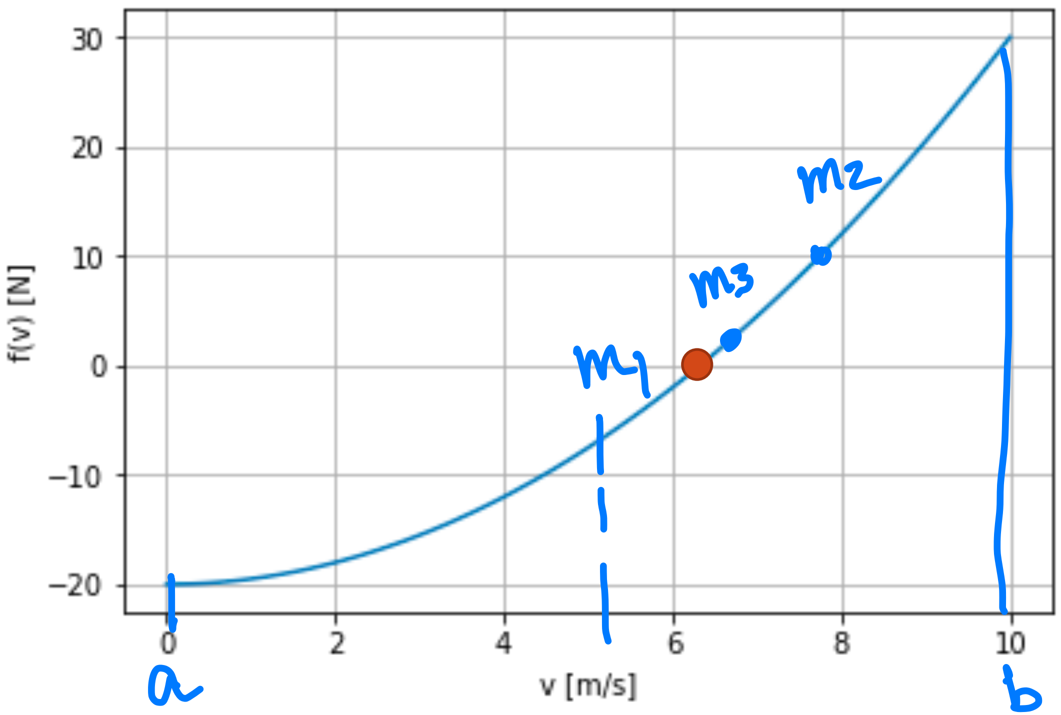
$f(a)$ $f(b)$

A) $[-10, -1.8]$ $f(a) \cdot f(b) < 0 \rightarrow$ ok! $\rightarrow k > \log_2\left(\frac{8.2}{2^{-4}}\right) \simeq 7.3$ (8 iterations)

~~*B)* $[-3, -2.1]$~~ $f(a) \cdot f(b) > 0 \rightarrow$ not ok!

C) $[-4, 1.9]$ $f(a) \cdot f(b) < 0 \rightarrow$ ok! $\rightarrow k > \log_2\left(\frac{5.9}{2^{-4}}\right) \simeq 6.56$ (7 iterations)

Bisection method



f(a) , f(b) , f(m)

new fc evaluation

f(m)

Algorithm:

1. Take two points, a and b, on each side of the root such that $f(a)$ and $f(b)$ have opposite signs.
2. Calculate the midpoint $m = \frac{a+b}{2}$
3. Evaluate $f(m)$ and use m to replace either a or b, keeping the signs of the endpoints opposite.

Bisection Method - summary

- The function must be continuous with a root in the interval $[a, b]$
- Requires only one function evaluations for each iteration!
  - The first iteration requires two function evaluations.
- Given the initial internal $[a, b]$, the length of the interval after k iterations is $\frac{b-a}{2^k}$
- Has linear convergence

Newton's method

- Recall we want to solve $f(x) = 0$ for $f: \mathcal{R} \to \mathcal{R}$
- The Taylor expansion:

$$\underbrace{f(x_k + h)}_{\text{nonlinear}} \approx \overbrace{\underline{f(x_k)} + \underline{f'(x_k)}h}^{\text{linear approximation of } f(x)} = \hat{f}(h)$$

gives a linear approximation for the nonlinear function f near x_k.

$$f(x_k + h) = 0$$

$$\hat{f}(h) = 0 \implies f(x_k) + f'(x_k)h = 0$$

$$\boxed{h = -\frac{f(x_k)}{f'(x_k)}} \rightarrow \text{Newton step}$$

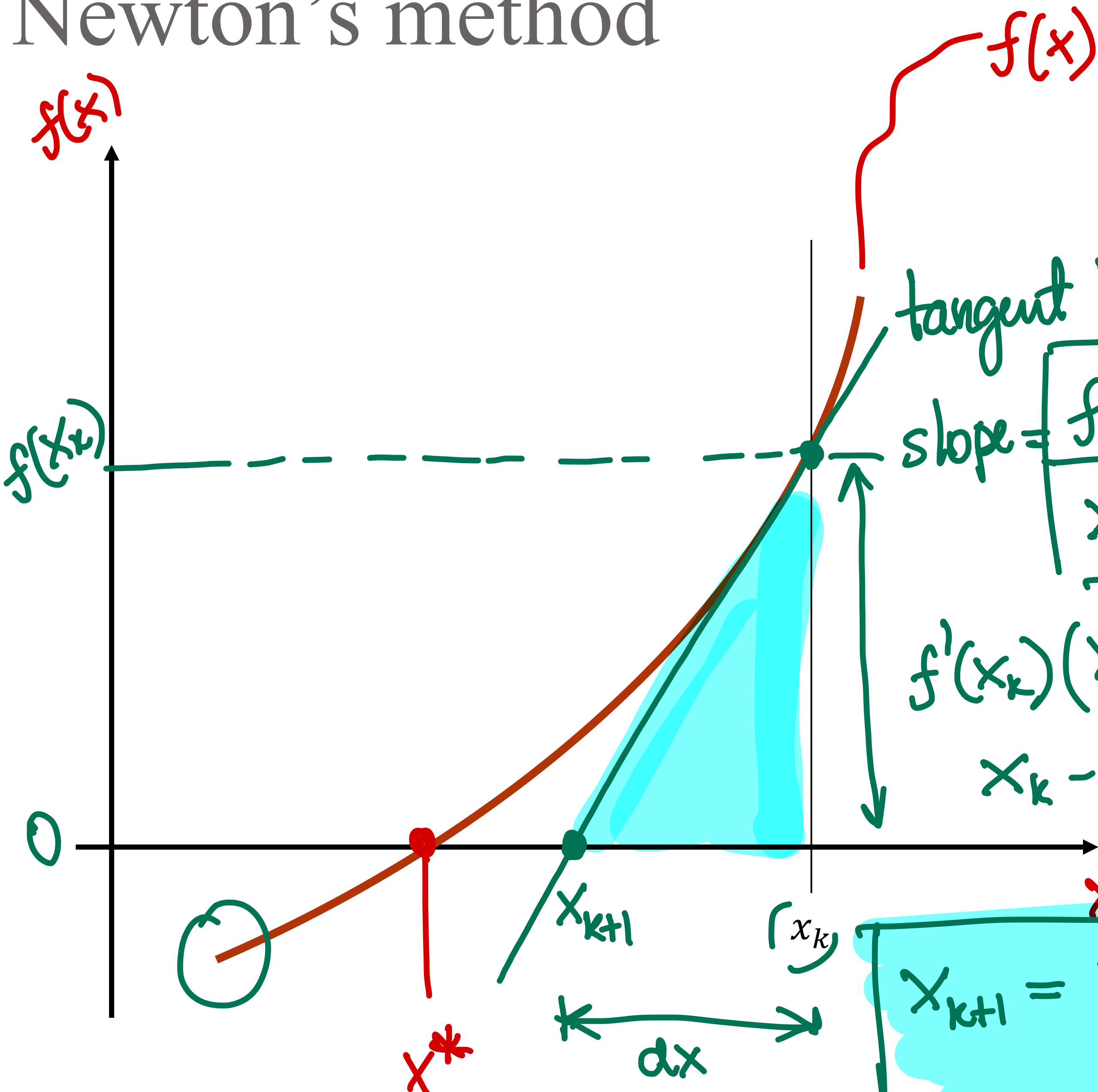
Newton algorithm:

x_0 = random (initial guess)

$$f(x_0), f'(x_0) \Rightarrow h \implies \boxed{x_{k+1} = x_k + h} \rightarrow \text{Newton update}$$

Newton's method

Find x^* s.t. $f(x^*) = 0$



tangent line at x_k

$$\boxed{\text{slope} = \frac{f(x_k) - 0}{x_k - x_{k+1}} = f'(x_k)}$$

$$f'(x_k)(x_k - x_{k+1}) = f(x_k)$$

$$x_k - x_{k+1} = \frac{f(x_k)}{f'(x_k)}$$

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}}$$

Example

$x_1 = ?$

$x_0 = 0$

Consider solving the nonlinear equation

$$\boxed{5 = 2.0\, e^x + x^2} \Rightarrow f(x) = 2e^x + x^2 - 5 = 0$$

What is the result of applying **one iteration** of Newton's method for solving nonlinear equations with initial starting guess $x_0 = 0$, i.e. what is x_1?

$x_{k+1} = x_k + h \qquad h = -\dfrac{f(x_k)}{f'(x_k)}$

A) -2
B) 0.75
C) -1.5
$\boxed{D)\ 1.5}$
E) 3.0

$f'(x) = 2e^x + 2x$

$x_0 \Rightarrow f(x_0) = 2 - 5 = -3$

$f'(x_0) = 2$

$h = -\dfrac{f}{f'} = -\dfrac{(-3)}{2}$

$h = 1.5$

$x_1 = x_0 + h = 0 + 1.5 \Rightarrow \boxed{x_1 = 1.5}$

Newton's Method - summary

- Must be started with initial guess close enough to root (convergence is only local). Otherwise it may not converge at all.

- Requires function and first derivative evaluation at each iteration (think about two function evaluations)

- Typically has quadratic convergence

$$\lim_{k\to\infty} \frac{||e_{k+1}||}{||e_k||^{2}} = \boxed{C}, \qquad 0 < C < \infty$$

$r=2$

- What can we do when the derivative evaluation is too costly (or difficult to evaluate)?

Secant method

$df \Rightarrow$ approximation for $f'(x)$

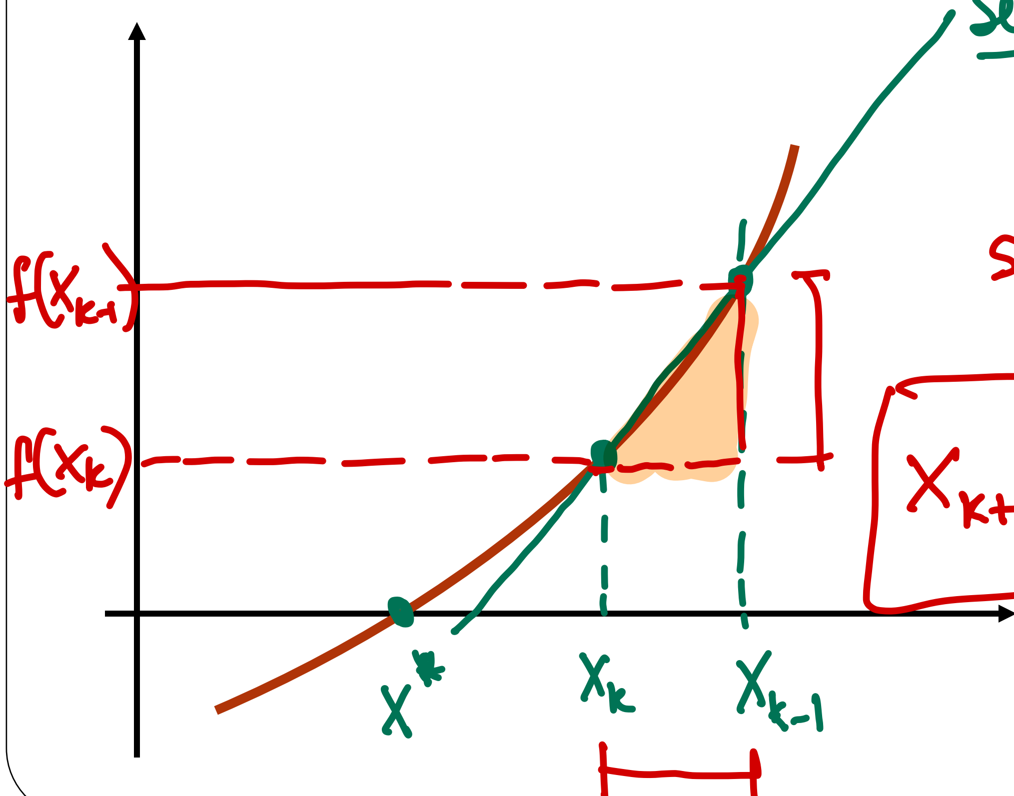
Also derived from Taylor expansion, but instead of using $f'(x_k)$, it approximates the tangent with the secant line:

$$x_{k+1} = x_k - f(x_k)/\cancel{f'(x_k)} \longrightarrow x_{k+1} = x_k - \frac{f(x_k)}{df(x_k)}$$

secant line

(2 points)!

$\boxed{x_0, x_1}$



$$\text{slope} = \frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k} = df$$

$$\boxed{x_{k+1} = x_k - \frac{f(x_k)}{\dfrac{f(x_k) - f(x_{k-1})}{(x_k - x_{k-1})}}}$$

Secant Method - summary

- ❑ Still local convergence
- ❑ Requires only one function evaluation per iteration (only the first iteration requires two function evaluations)
- ❑ Needs two starting guesses
- ❑ Has slower convergence than Newton's Method – superlinear convergence

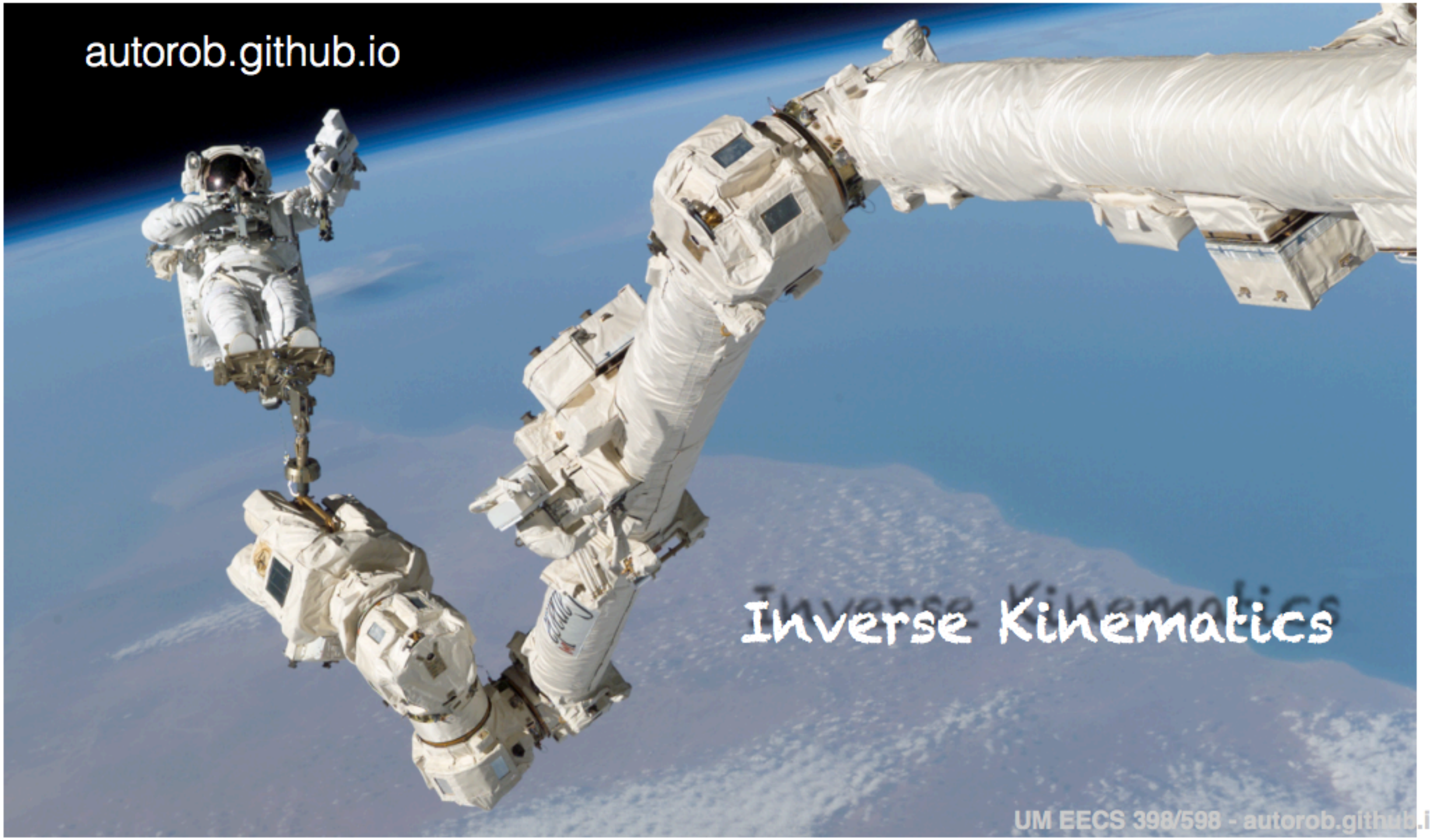
$$\lim_{k\to\infty}\frac{||e_{k+1}||}{||e_k||^r}=C, \qquad 1<r<2$$

$f' \longrightarrow df$

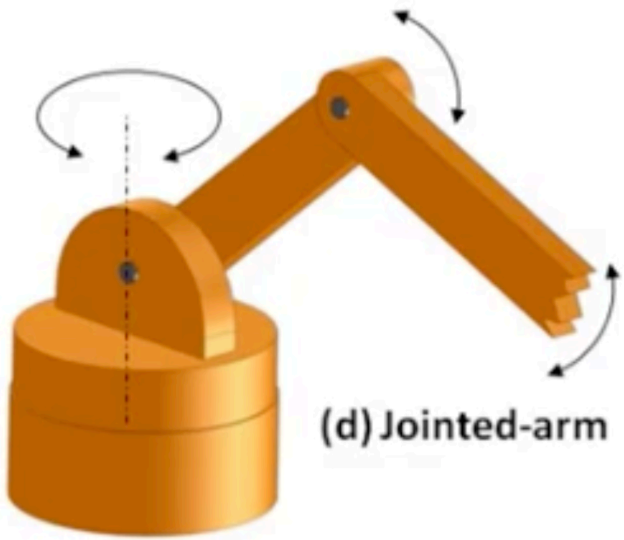
1D methods for root finding:

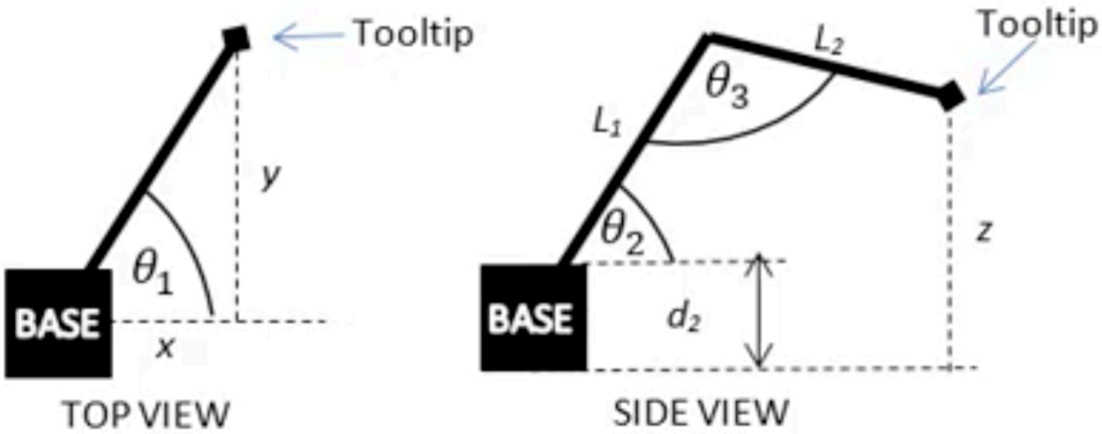
Method	Update	Convergence	Cost
Bisection	Check signs of $f(a)$ and $f(b)$ $t_k = \frac{\lvert b-a\rvert}{2^k}$	Linear ($r = 1$ and $c = 0.5$)	One function evaluation per iteration, no need to compute derivatives
Secant	$x_{k+1} = x_k + h$ $h = -f(x_k)/f'(x_k)$	Superlinear ($r = 1.618$), local convergence properties, convergence depends on the initial guess	One function evaluation per iteration (two evaluations for the initial guesses only), no need to compute derivatives
Newton	$x_{k+1} = x_k + h$ $h = -f(x_k)/dfa$ $dfa = \frac{f(x_k) - f(x_{k-1})}{(x_k - x_{k-1})}$	Quadratic ($r = 2$), local convergence properties, convergence depends on the initial guess	Two function evaluations per iteration, requires first order derivatives

Nonlinear system of equations



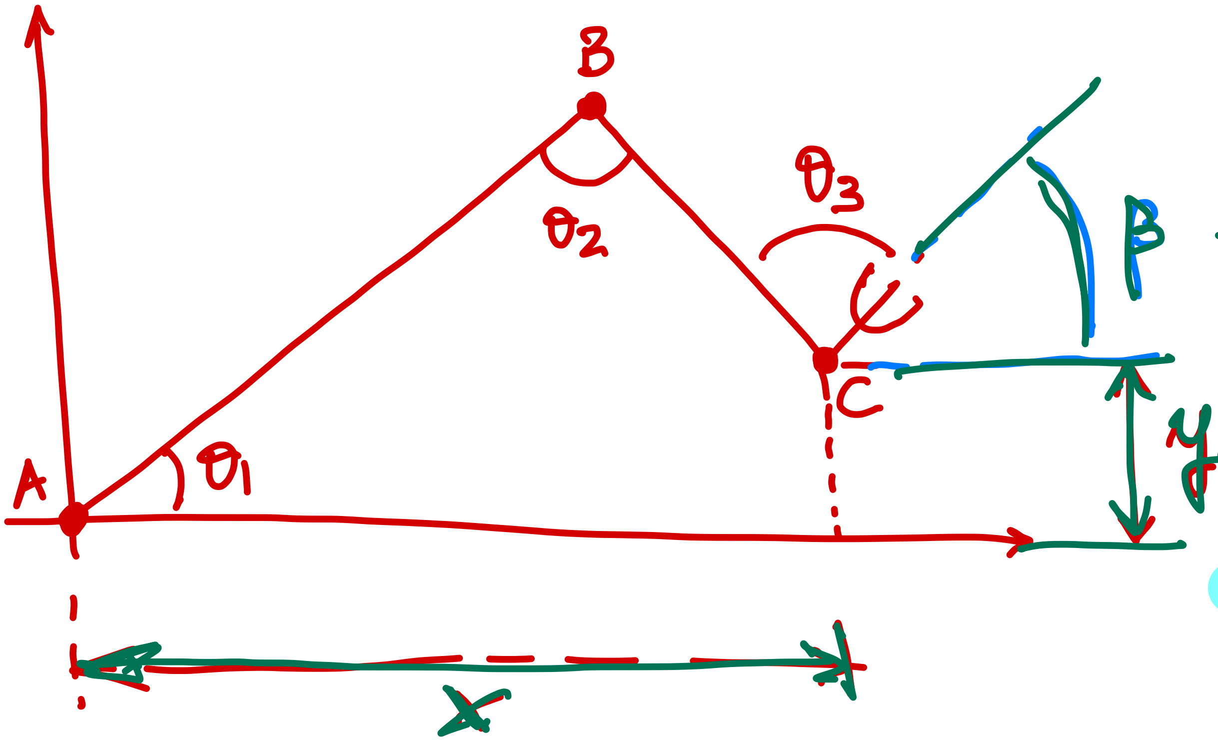
Robotic arms





https://www.youtube.com/watch?v=NRgNDlVtmz0 (Robotic arm 1)
https://www.youtube.com/watch?v=9DqRkLQ5Sv8 (Robotic arm 2)
https://www.youtube.com/watch?v=DZ_ocmY8xEI (Blender)

Inverse Kinematics



Given:

x, y, β

Find:

$\theta_1, \theta_2, \theta_3$

$$\boxed{\underset{\sim}{f}(\underset{\sim}{\theta}) = \underset{\sim}{0}}$$

$$\left.\begin{aligned} f_1(\theta_1, \theta_2, \theta_3) &= 0 \\ f_2(\theta_1, \theta_2, \theta_3) &= 0 \\ f_3(\theta_1, \theta_2, \theta_3) &= 0 \end{aligned}\right\}$$

3 eq.
+
3 unknowns } → solve

iterative

Nonlinear system of equations

Goal: Solve $\boldsymbol{f}(\boldsymbol{x}) = \mathbf{0}$ for $\boldsymbol{f}: \mathcal{R}^n \to \mathcal{R}^n$

$$\underset{\sim}{f}(\underset{\sim}{x}) = \underset{\sim}{0}$$

$$\underset{\sim}{f}(\underset{\sim}{x}) = \begin{bmatrix} f_1(x_1, x_2, x_3, \ldots, x_n) \\ f_2(x_1, x_2, x_3, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, x_3, \ldots, x_n) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Suppose

$$\begin{cases} x_1^2 + 2x_1x_2 + x_2^3 = 4 \\ 2x_1 + 3x_2 = 5 \end{cases} \Longrightarrow \begin{aligned} f_1 &= -x_1^2 - 2x_1x_2 - x_2^3 + 4 = 0 \\ f_2 &= -2x_1 - 3x_2 + 5 = 0 \end{aligned}$$

Newton's method (ND)

Approximate the nonlinear function $\boldsymbol{f}(\boldsymbol{x})$ by a linear function using Taylor expansion:

$$\underbrace{\underbrace{\underset{\sim}{f}(\underset{\sim}{x}+\underset{\sim}{s})}_{\text{vector}}}_{\text{nonlinear}} \cong \underbrace{\underbrace{\underset{\sim}{f}(\underset{\sim}{x})}_{\text{vector}} + \underbrace{\underbrace{\underset{\sim}{J}(\underset{\sim}{x})}_{\text{matrix}} \underset{\sim}{s}}_{\text{vector}}}_{\text{linear approximation}}$$

$\underset{\sim}{s}$ ← vector

$$\underset{\sim}{f} = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ f_2 \\ \vdots \\ f_n(x_1, \ldots, x_n) \end{bmatrix} \qquad \underset{\sim}{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}_{n \times n}$$

↑ Jacobian

$$J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$\underbrace{f(x+s)}_{=0} \simeq \underbrace{f(x) + J(x)\, s}_{=0}$$

$$f(x) + J(x)\, s = 0$$

$$\boxed{J(x)\, s = -f(x)}$$

→ solve for s

Linear system of equation

x_0 = initial vector

$$x_{k+1} = x_k + s$$

Newton's method

$\underset{\sim}{f}(\underset{\sim}{x}) = \underset{\sim}{0}$

Algorithm:

$\underset{\sim}{x}_0$: initial guess

for $i = 1, 2, \dots$

$n^2 \leftarrow$ evaluate $J(x_k) = J$

evaluate $f(x_k) = f$

$O(n^3) \leftarrow$ solve $\underset{\sim}{J}\,\underset{\sim}{s} = -\underset{\sim}{f} \rightarrow$ Find $\underset{\sim}{s}$

update $\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k + \underset{\sim}{s}$

iterative process sequence of linear system of equation

Convergence:

- Typically has quadratic convergence
- Drawback: Still only locally convergent

Cost:

n^2

- Main cost associated with computing the Jacobian matrix and solving the Newton step.

n^3

Example

$$\underset{\sim}{f} = \begin{bmatrix} 2y + x - 2 \\ x^2 + 4y^2 - 4 \end{bmatrix}$$

Consider solving the nonlinear system of equations

$$2 = 2y + x$$
$$4 = x^2 + 4y^2$$

What is the result of applying one iteration of Newton's method with the following initial guess?

$$\boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$\underset{\sim}{J} = \begin{bmatrix} 1 & 2 \\ 2x & 8y \end{bmatrix}$$

$$\underset{\sim}{x}_1 = \underline{x}_0 + \underset{\sim}{s} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

$J(x_0)$, $f(x_0)$

$$\underline{f}(\underline{x}_0) = \begin{bmatrix} -1 \\ -3 \end{bmatrix} \qquad \underline{\underline{J}}(\underline{x}_0) = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}$$

$$\underset{\sim}{J}\,\underline{s} = -\underline{f}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}\begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$$

$$s_1 + 2s_2 = 1$$
$$2s_1 = 3 \longrightarrow s_1 = 1.5$$
$$2s_2 = 1 - 1.5 = -0.5$$
$$s_2 = -0.25$$

$$\underset{\sim}{x}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 1.5 \\ -0.25 \end{bmatrix} = \begin{bmatrix} 2.5 \\ -0.25 \end{bmatrix}$$

Newton's method

$\boldsymbol{x}_0 = initial\ guess$

$For\ k = 1,2, \ldots$

Evaluate $\mathbf{J} = \boldsymbol{J}(\boldsymbol{x}_k)$

Evaluate $\boldsymbol{f}(\boldsymbol{x}_k)$

la.solve

Factorization of Jacobian (for example $\mathbf{LU} = \boldsymbol{J}$)

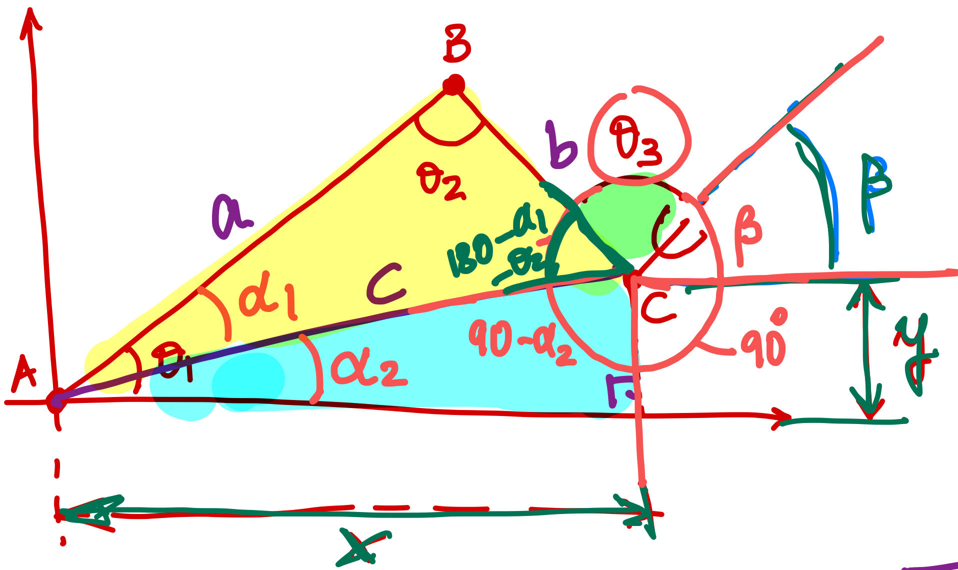
Solve using factorized J (for example $\mathbf{LU}\,\boldsymbol{s}_k = -\boldsymbol{f}(\boldsymbol{x}_k)$)

Update $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$

Newton's method - summary

- Typically quadratic convergence (local convergence)
- Computing the Jacobian matrix requires the equivalent of n^2 function evaluations for a dense problem (where every function of $\boldsymbol{f}(\boldsymbol{x})$ depends on every component of $\boldsymbol{x}$).
- Computation of the Jacobian may be cheaper if the matrix is sparse.
- The cost of calculating the step $\boldsymbol{s}$ is $O(n^3)$ for a dense Jacobian matrix (Factorization + Solve)
- If the same Jacobian matrix $\boldsymbol{J}(\boldsymbol{x}_k)$ is reused for several consecutive iterations, the convergence rate will suffer accordingly (trade-off between cost per iteration and number of iterations needed for convergence)

Inverse Kinematics



$x, y, \beta \longrightarrow \theta_1, \theta_2, \theta_3$

$c = \sqrt{x^2 + y^2}$ ✓

a, b given ✓

$$\boxed{\theta_1 = \alpha_1 + \alpha_2}$$

$\alpha_1 = \theta_1 - \alpha_2$

$\alpha_2 = \tan^{-1}(y/x)$ ✓

$$c^2 = a^2 + b^2 - 2ab\cos\theta_2 \rightarrow \boxed{f_1 = c^2 - a^2 - b^2 + 2ab\cos\theta_2 = 0}$$

$$b^2 = a^2 + c^2 - 2ac\cos\alpha_1 \rightarrow \boxed{f_2 = b^2 - a^2 - c^2 + 2ac\cos(\theta_1 - \alpha_2) = 0}$$

$$(\cancel{180} - \alpha_1 - \theta_2) + \theta_3 + \beta + \cancel{90} + (\cancel{90} - \alpha_2) = \cancel{360}$$

$$-\alpha_1 - \theta_2 + \theta_3 + \beta - \alpha_2 = 0$$

$$-(\theta_1 - \cancel{\alpha_2}) - \theta_2 + \theta_3 + \beta \cancel{-\alpha_2} = 0$$

$$\boxed{f_3 = -\theta_1 - \theta_2 + \theta_3 + \beta = 0}$$