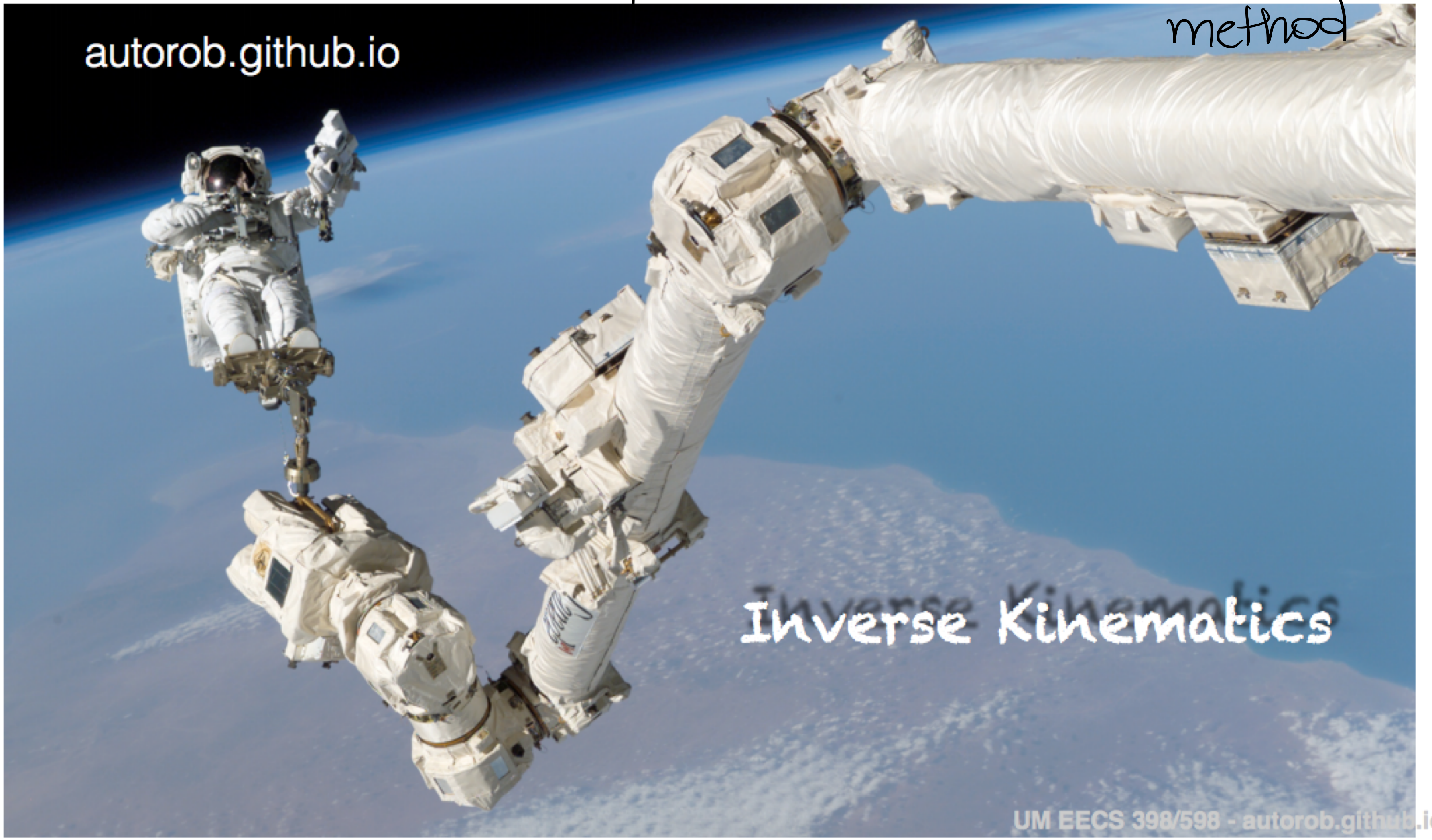
# Nonlinear Equations

linear $f(x) = y \Rightarrow Ax = y$

nonlinear : $f(x_k) = y$

iterative method



# How can we solve these equations?

- Spring force:

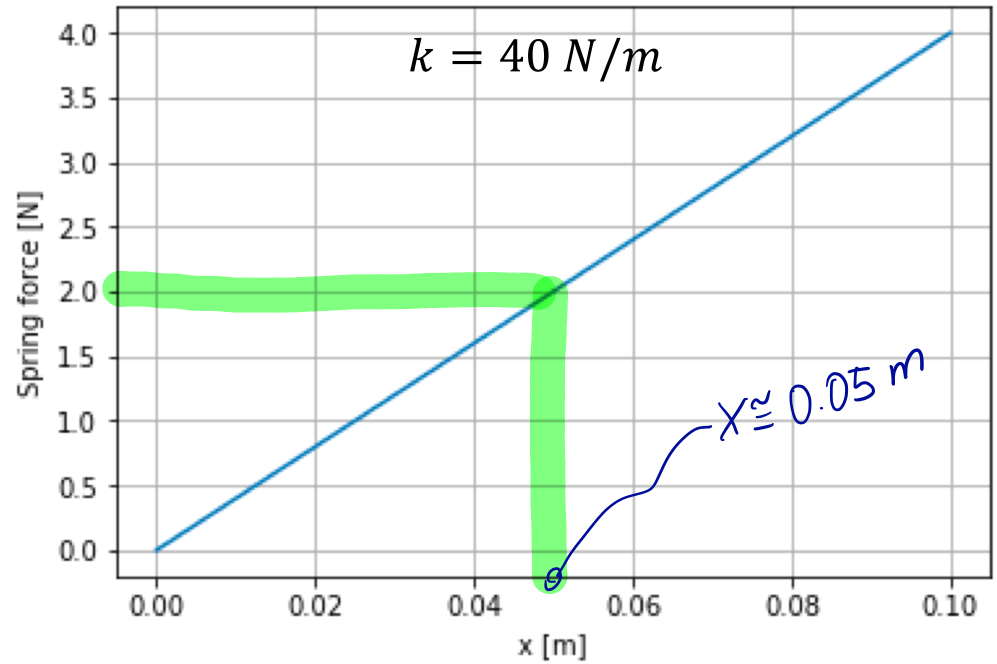
$F = k\,x$

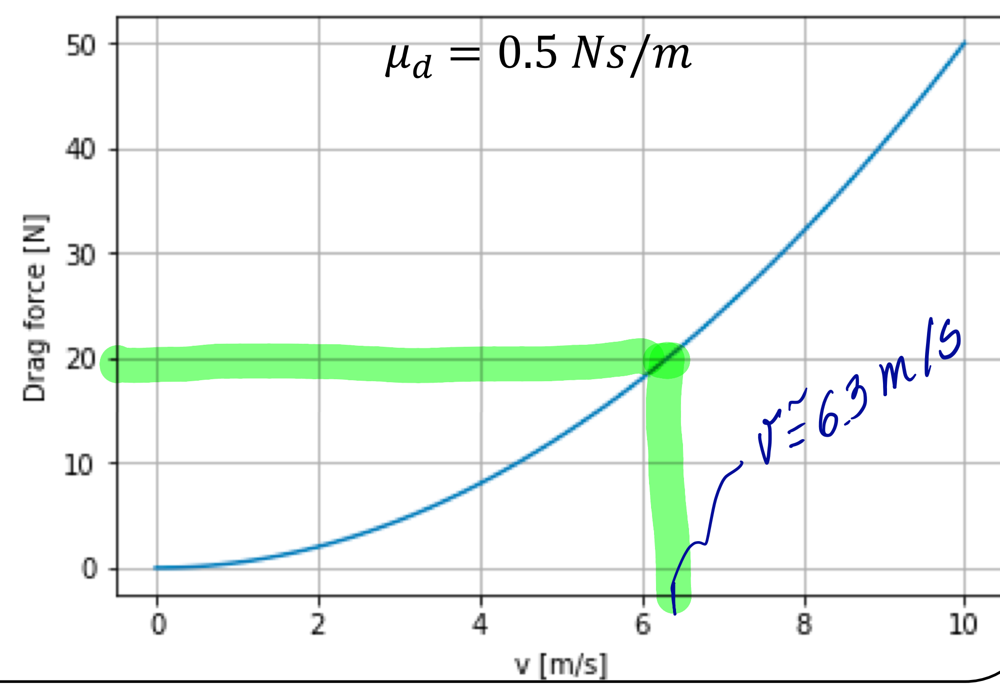
What is the displacement when $F = 2\text{N}$?

- Drag force:

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

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



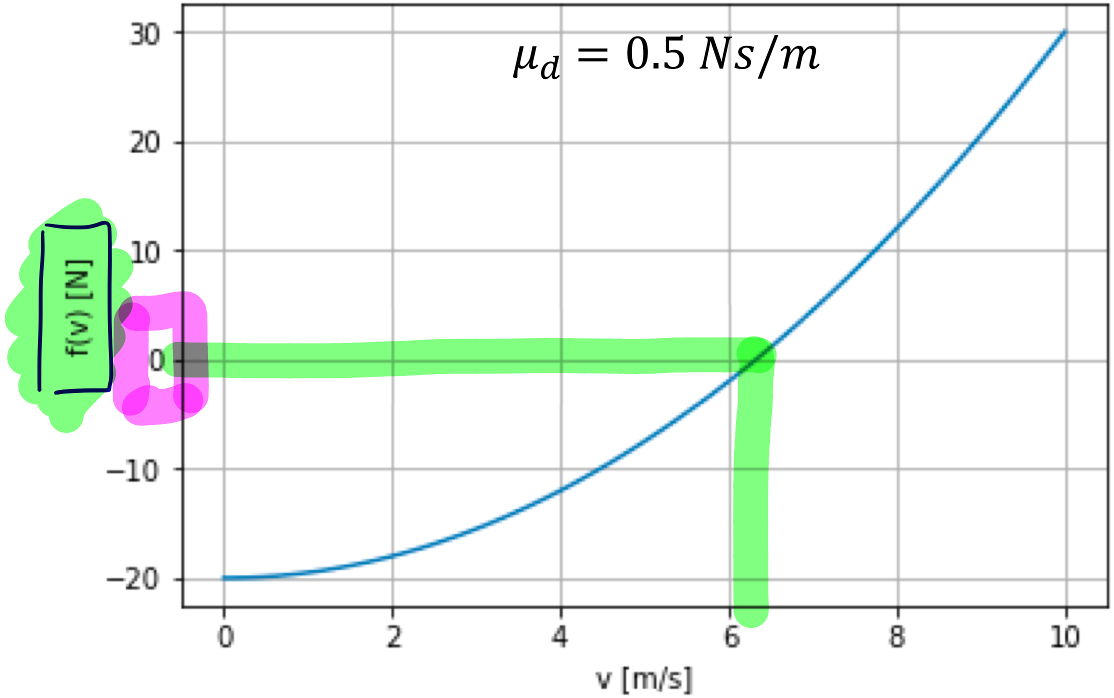
- Spring force:

$$f(x) = k\,x - F = 0$$

- Drag force:

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

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

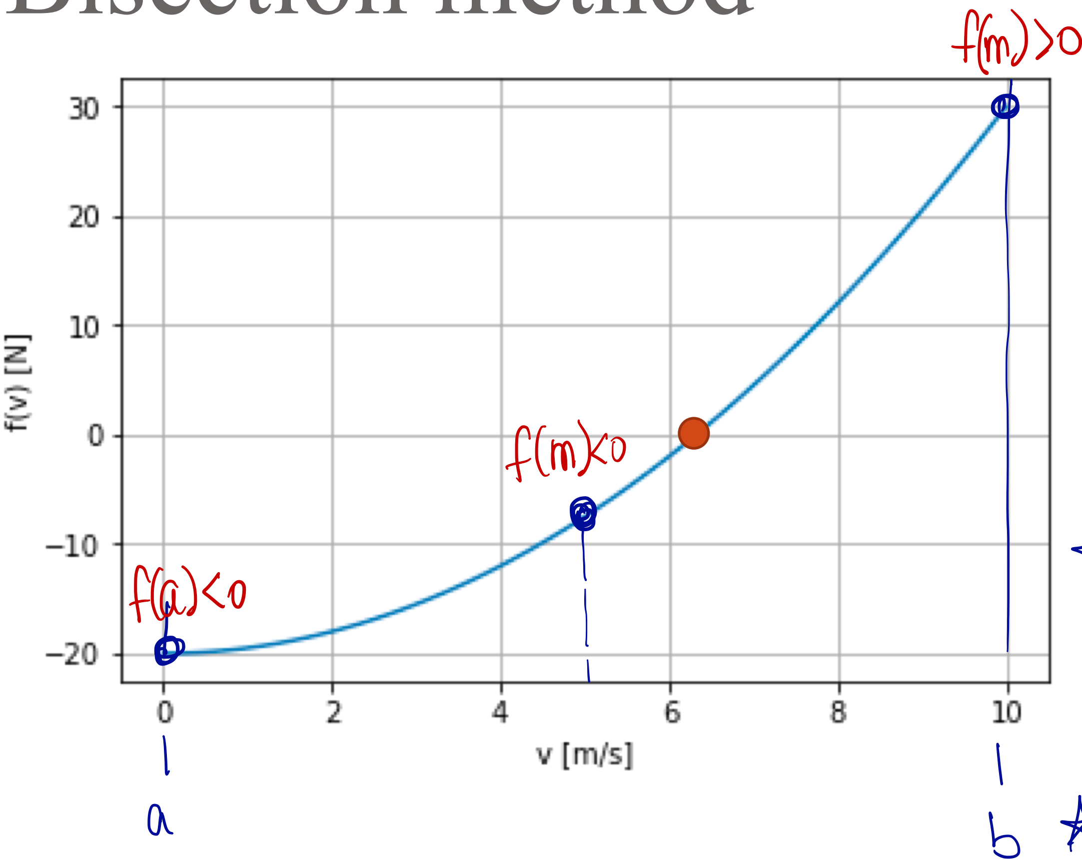



# Nonlinear Equations in 1D

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

Often called **Root Finding**

# Bisection method




★ start with interval that contains the root

$t_0 = [a, b]$

$a = 0 \quad , \quad b = 10$

$\Delta t_0 = |b - a| = 10$

★ Get midpoint:

$m = \frac{b - a}{2} \Rightarrow m = 5$

★ Check signs:

```
if sign(f(a)) == sign(f(m)):
    new_interval  t_k = [m, b]  =>  a = m
else:
    new_interval  t_k = [a, m]  =>  b = m
```

$t_1 \Rightarrow \Delta t_1 = \frac{\Delta t_0}{2}$

$t_2 \Rightarrow \Delta t_2 = \frac{\Delta t_1}{2}$

$t_3 \Rightarrow \Delta t_3 = \frac{\Delta t_2}{2}$

# Convergence

- 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.

$\Delta t_k < tol \Rightarrow$ stops

At each iteration: $\Delta t_k = \frac{\Delta t_{k-1}}{2}$

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

Convergence rate: $\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|} = \frac{\Delta t_0 / 2^{k+1}}{\Delta t_0 / 2^k} = \frac{2^k}{2^{k+1}} = \frac{1}{2}$

# Convergence

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

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

recall power iteration:

$$\lim_{k\to\infty} \frac{||e_{k+1}||}{||e_k||} = \left|\frac{\lambda_2}{\lambda_1}\right|$$

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

now Bisection

" $= 0.5$

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)

# Example:

$$\Delta t_k = \frac{|b-a|}{2^k} < 2^{-4}$$

$$2^k > \frac{|b-a|}{tol} \Rightarrow \boxed{k > \log_2\left(\frac{|b-a|}{tol}\right)}$$

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 $2^{-4}$?

*A)* $[-10, -1.8]$ $k > \log_2\left(\frac{8.2}{2^{-4}}\right) = 7.03 \rightarrow$ at least 8 iter

*B)* $[-3, -2.1]$ $\longrightarrow$ it can't be used since $\text{sign}(f(a)) = \text{sign}(f(b))$

*C)* $[-4, 1.9]$ $k > \log_2\left(\frac{5.9}{2^{-4}}\right) = 6.56 \rightarrow$ at least 7 iter

# Bisection Method - summary

~~Mark the incorrect statement about the Bisection Method:~~

- ❑ 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

Demo: "Bisection Method"

# Newton's method

- Recall we want to solve $f(x) = 0$ for $f\colon \mathcal{R} \to \mathcal{R}$

- The Taylor expansion:

$$\overbrace{f(\underbrace{x_k + h}_{x_{k+1}})}^{\text{nonlinear function}} \approx \underbrace{f(x_k) + f'(x_k)h}_{\text{linear approx}}$$

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

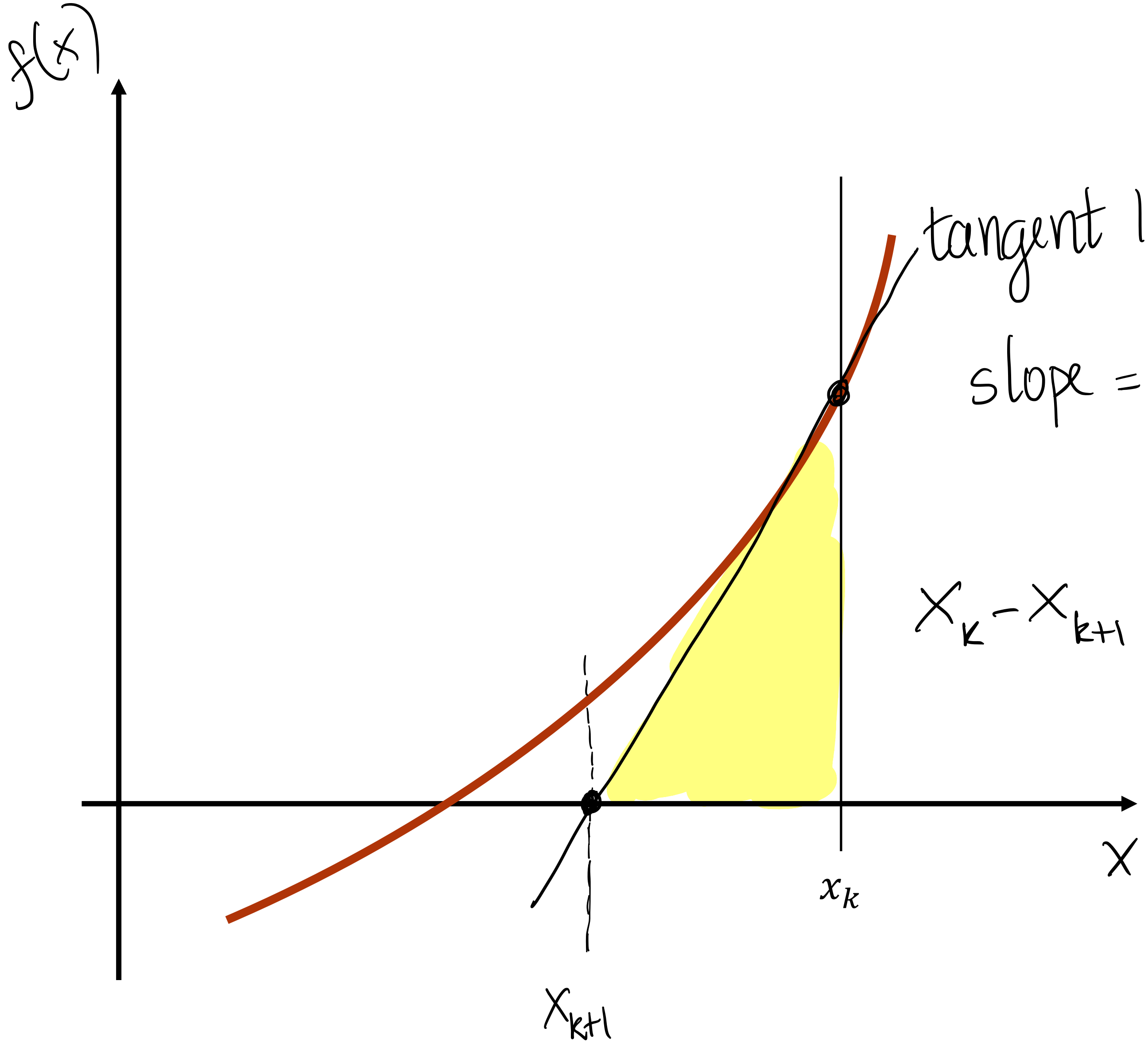
$$f(x_k + h) = 0 \longrightarrow f(x_k) + f'(x_k)h = 0$$

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

$x_0$ = initial guess

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

# Newton's method




tangent line at $x_k$

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

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

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

same ✓

# Iclicker question

Consider solving the nonlinear equation

$$5 = 2.0\, e^{x} + x^2$$

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 $x_1$?

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

$$f(x) = 2e^{x} + x^2 - 5$$

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

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{(2e^0 + 0 - 5)}{2e^0 + 0} = \frac{-(-3)}{2}$$

$$\boxed{x_1 = 1.5}$$

# 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)

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

- Typically has quadratic convergence

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

Demo: "Newton's Method" and "Convergence of Newton's Method"

# Secant method

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

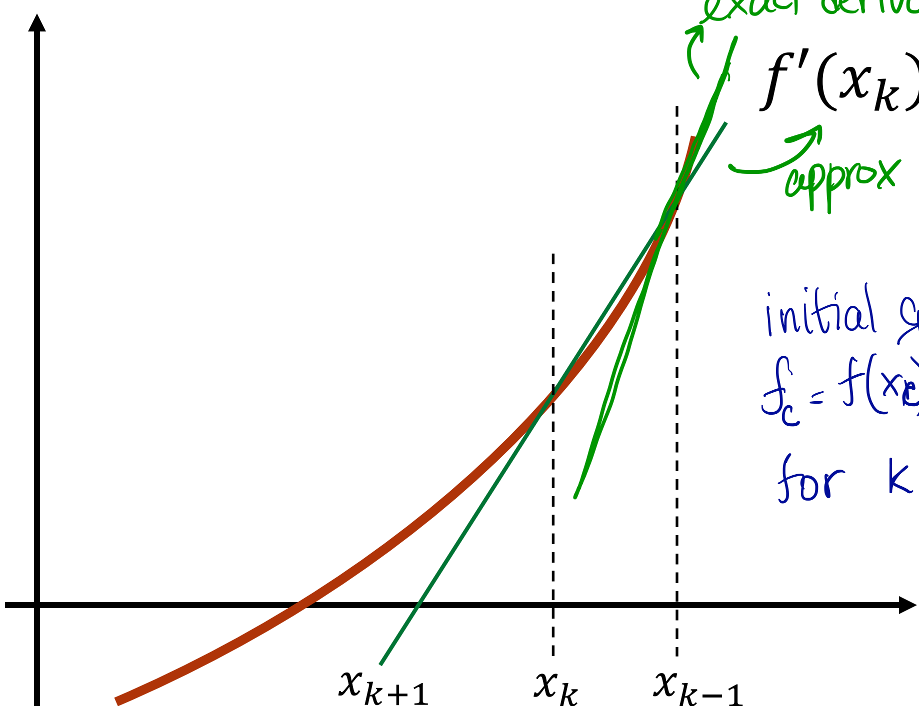
$$x_{k+1} = x_k - f(x_k)/f'(x_k)$$

Secant line:

exact derivative

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

approx




initial guess $x_c, x_b$ (two!)

$x_c \to x_1$, $x_b \to x_0$

$f_c = f(x_c)$ $\quad f_b = f(x_b)$

for $k = 1, 2, \ldots$

$$\text{dfapprox} = \frac{f_c - f_b}{x_c - x_b}$$

$$x_{k+1} = x_k - f_c/\text{dfapprox}$$

$f_b = f_c \qquad f_c = f(x_{k+1})$

only 1 f. eval !!

# 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

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

Demo: "Secant Method"

Demo: "Convergence of Secant Method"

# Nonlinear system of equations

$$\begin{cases} x_1^2 + 2x_1x_2 + x_2^3 - 4 = 0 \\ 2x_1 + 3x_2 - 5 = 0 \end{cases}$$

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

$$\begin{bmatrix} \rule{3em}{0.4pt} \\ \rule{3em}{0.4pt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

In other words, $\boldsymbol{f}(\boldsymbol{x})$ is a vector-valued function

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, x_3, \dots, x_n) \end{bmatrix}$$

If looking for a solution to $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{y}$, then instead solve

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{y} = \mathbf{0}$$

# Newton's method

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

nonlinear

linear approx

similar to Taylor but for ND

$$\underbrace{\boldsymbol{f}(\overbrace{\boldsymbol{x}+\boldsymbol{s}}^{x_{k+1}})}_{\text{nonlinear}} \approx \underbrace{\boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\,\boldsymbol{s}}_{\text{linear approx}}$$

$\frac{df}{dx_1}$ $\frac{df}{dx_n}$

where $\boldsymbol{J}(\boldsymbol{x})$ is the Jacobian matrix of the function $\boldsymbol{f}$:

gradient of $f(x)$

$$\boldsymbol{J}(\boldsymbol{x}) = \begin{pmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_1(\boldsymbol{x})}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_n(\boldsymbol{x})}{\partial x_n} \end{pmatrix} \quad \text{or} \quad [\boldsymbol{J}(\boldsymbol{x})]_{ij} = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j}$$

we want to find Newton step!

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

(or $b + As = 0 \rightarrow As = -b$ solve for $s$)

$$\boxed{J\,s = -f(x)} \rightarrow \text{solve for } s$$

# Newton's method

$\underset{\sim}{J}(\underset{\sim}{x})\, \underset{\sim}{s} = -\underset{\sim}{f}(\underset{\sim}{x}) \rightarrow$ solve for $\underset{\sim}{s}$

**Algorithm:** start with $x_0$ = initial guess

for $k = 1, 2, \ldots$

$J = J(x_k) \rightarrow$ evaluate Jacobian $\rightarrow O(n^2)$

$b = -f(x_k) \rightarrow$ evaluate function $\rightarrow O(n)$

$J s_k = b \rightarrow$ solve for $s_k \rightarrow O(n^3)$

$x_{k+1} = x_k + s_k \rightarrow$ update

**Convergence:**

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

sometimes not available

**Cost:**

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

very expensive method

# Example

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?

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

$$f(x_0) = \begin{bmatrix} -1 \\ -3 \end{bmatrix}$$

$$\underset{\sim}{f}(\underset{\sim}{x}) = \begin{bmatrix} 2y + x - 2 \\ 4y^2 + x^2 - 4 \end{bmatrix} \qquad \underset{=}{J} = \begin{bmatrix} 1 & 2 \\ 2x & 8y \end{bmatrix} \qquad J_0 = J(x_0) = \begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \Rightarrow \begin{array}{l} a + 2b = 1 \Longrightarrow 2b = 1 - 1.5 \Rightarrow b = -0.25 \,// \\ 2a = 3 \Rightarrow a = 1.5 \,// \end{array}$$

$$\underset{\sim}{s}_0 = \begin{bmatrix} 1.5 \\ -0.25 \end{bmatrix} \Longrightarrow x_1 = \begin{bmatrix} 2.5 \\ -0.25 \end{bmatrix} //$$

# 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) for $k = 1, 2, \dots$

$J = \begin{cases} J(x_k) \longrightarrow \text{evaluate Jacobian} \longrightarrow O(n^2) \\ \tilde{J}(x_k) \longrightarrow \text{some approx. of Jacobian} \end{cases}$

$b = -f(x_k) \rightarrow$ evaluate function $\longrightarrow O(n)$

Compute factorization of $J \longrightarrow O(n^3)$

$J s_k = b \longrightarrow$ with already factorized $J \longrightarrow O(n^2)$

$x_{k+1} = x_k + s_k \longrightarrow$ update

→ perform these two steps only every few iterations

→ cheaper

→ slower convergence

$\}$ trade-off !