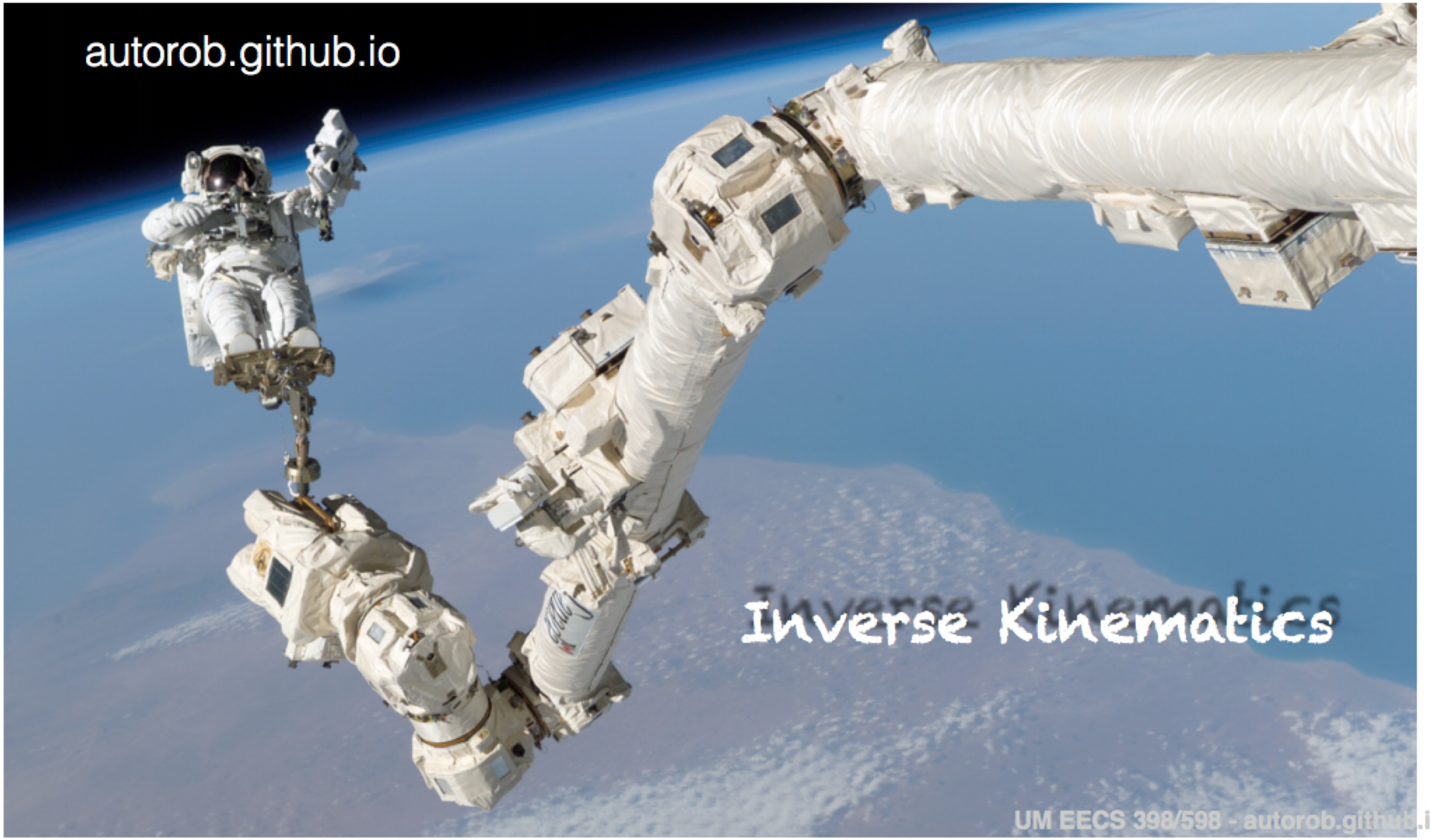
Nonlinear Equations

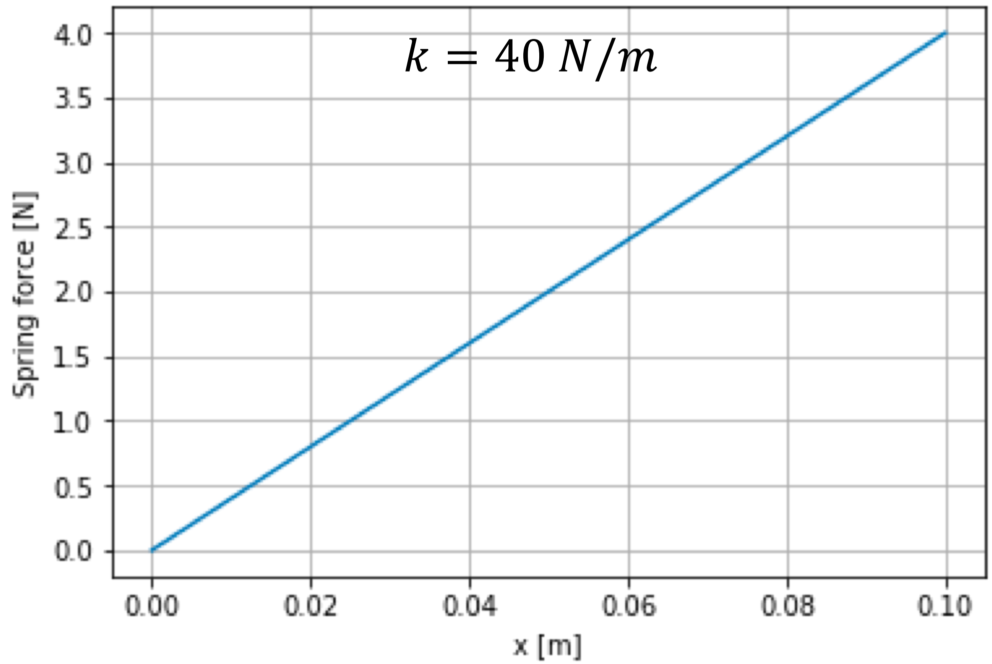

autorob.github.io
Inverse Kinematics
UM EECS 398/598 - autorob.github.io

How can we solve these equations?

- Spring force:

$F = k\,x$

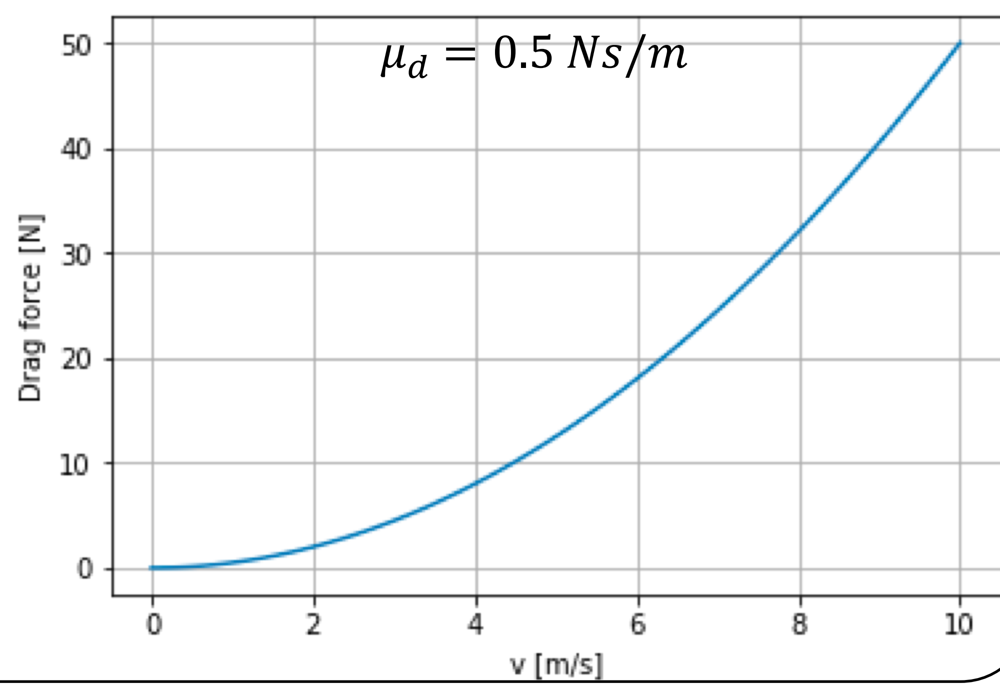
What is the displacement when $F = 2\text{N}$?



- Drag force:

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

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



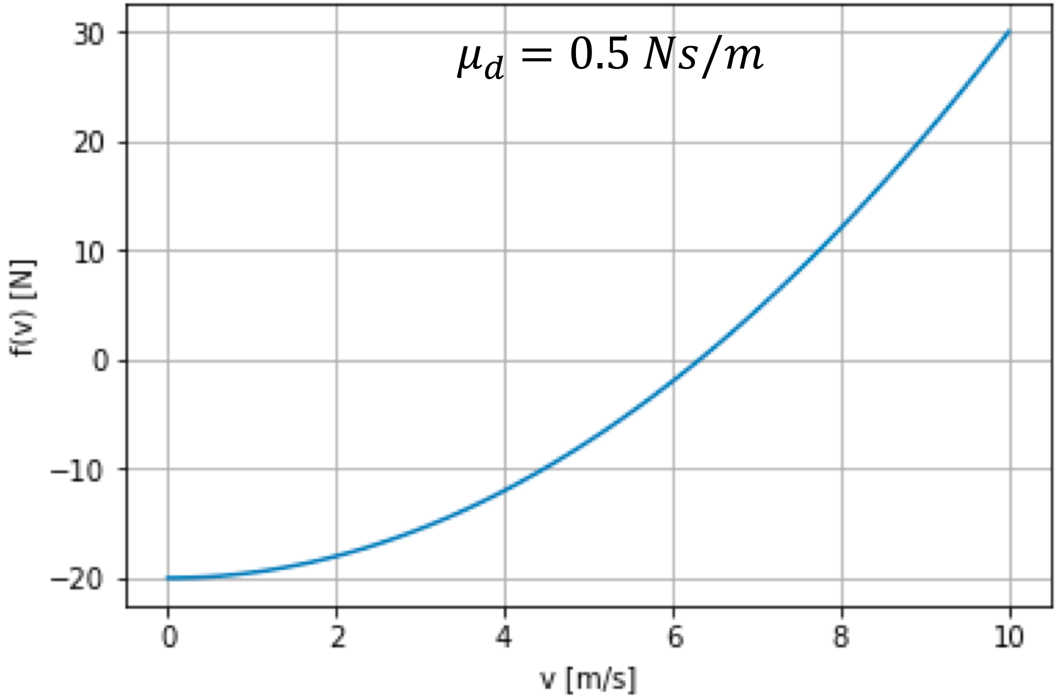
- Spring force:

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

- Drag force:

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

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

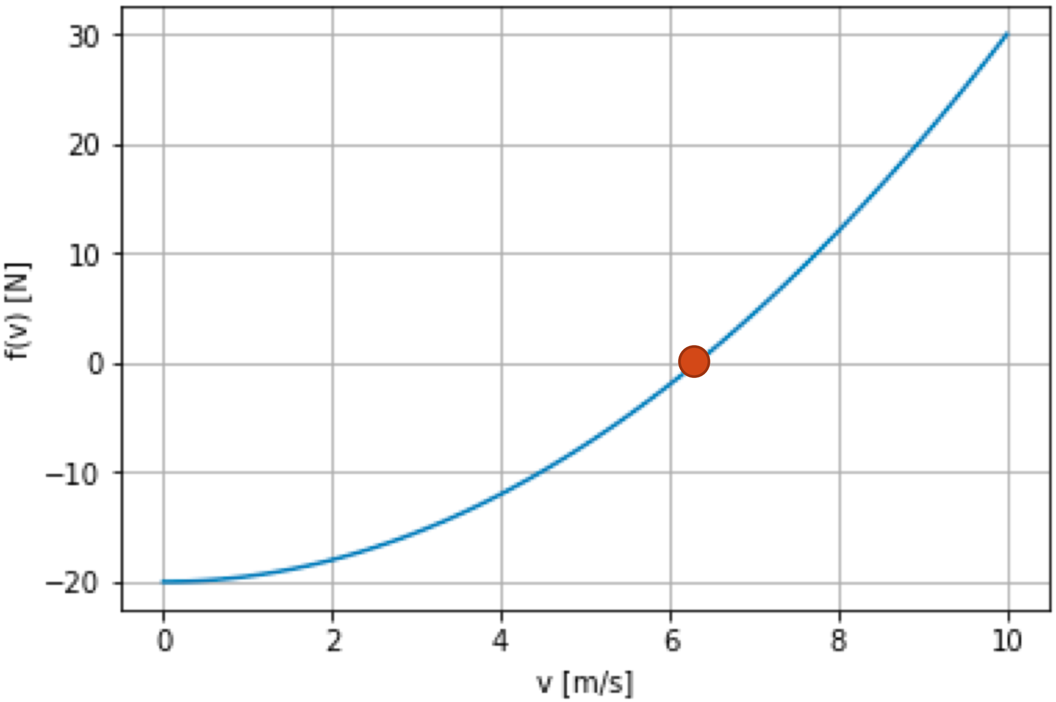


Nonlinear Equations in 1D

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

Often called **Root Finding**

Bisection method



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.

Convergence

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

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

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

Example:

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) $[-10, -1.8]$

B) $[-3, -2.1]$

C) $[-4, 1.9]$

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

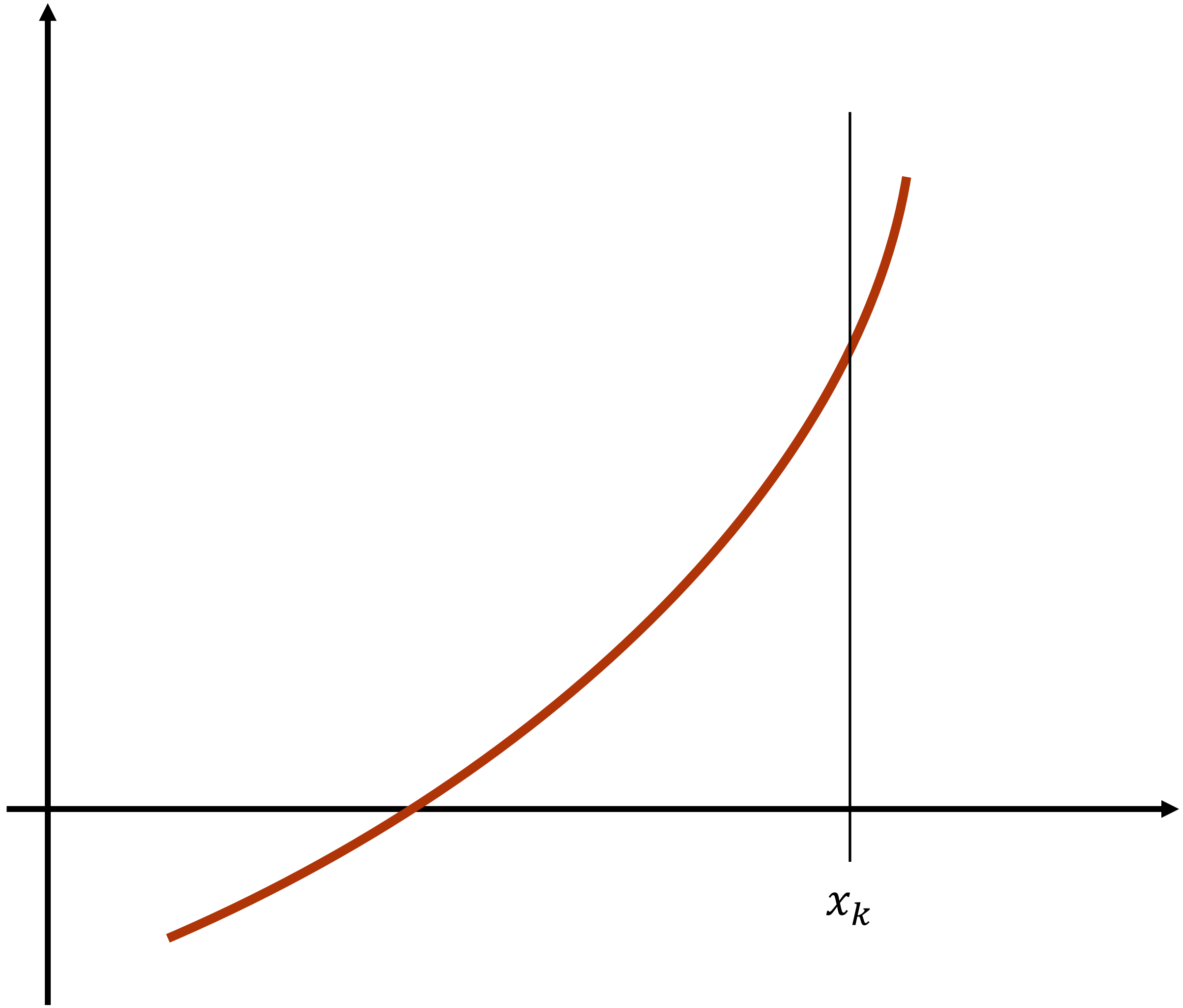
- Recall we want to solve $f(x) = 0$ for $f: \mathcal{R} \to \mathcal{R}$

- The Taylor expansion:

$$f(x_k + h) \approx f(x_k) + f'(x_k)h$$

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

Newton’s method



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
D) 1.5
E) 3.0

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"

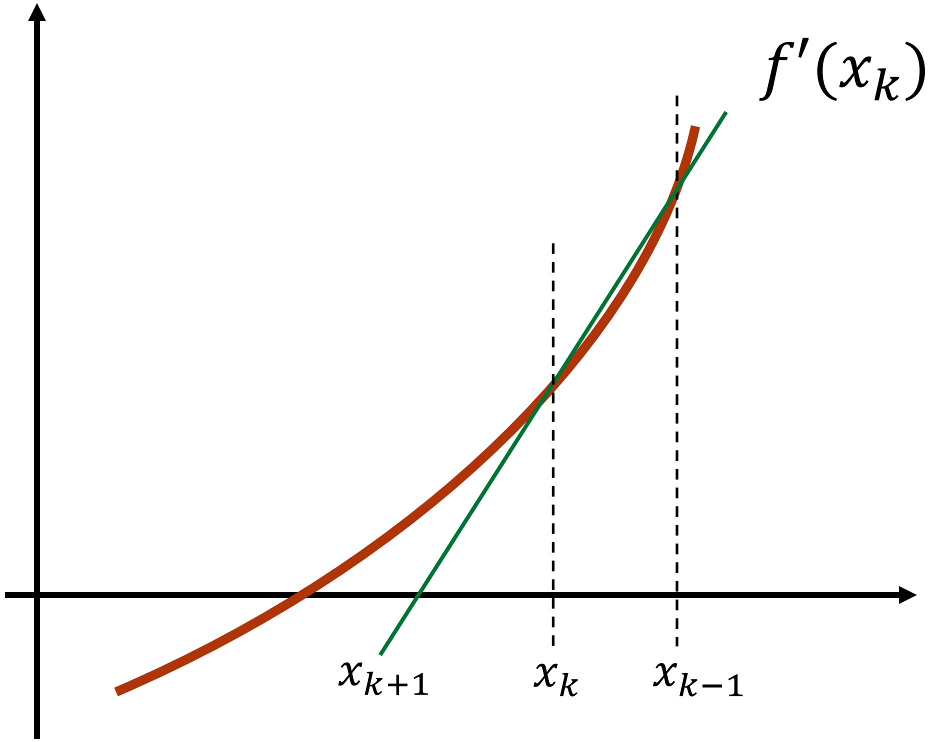
Secant method

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:

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



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:

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

- **Algorithm:**

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

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

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

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

Newton's method

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

$$\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{s}) \approx \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\,\boldsymbol{s}$$

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

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

Newton's method

Algorithm:

Convergence:

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

Cost:

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

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)

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?

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

Finite Difference

Find an approximate for the Jacobian matrix:

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

In 1D:

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

In ND:

$$[\boldsymbol{J}(\boldsymbol{x})]_{ij} = \frac{\partial f_i(\boldsymbol{x})}{\partial x_j} \approx \frac{f_i\left(\boldsymbol{x}+h\,\boldsymbol{\delta}_j\right) - f_i(\boldsymbol{x})}{h}$$