NTNU | Norwegian University of Science and Technology

TTK4135 – Lecture 17
Nonlinear Equations

Lecturer: Lars Imsland

Outline

- A brief summary of Ch. 10: (Nonlinear) Least Squares

- **Nonlinear equations** (Ch. 11)
  - Newton's method for solving nonlinear equations
  - Convergence
  - Merit functions

Reference: N&W Ch. 11-11.1

Gradient and Jacobian

- The *gradient* of a scalar function $f(x)$ of several variables is

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1} & \frac{\partial f}{\partial x_2} & \cdots & \frac{\partial f}{\partial x_n} \end{pmatrix}^\top$$

- Say $f(x) = \begin{pmatrix} f_1(x) & f_2(x) & \dots & f_m(x) \end{pmatrix}^\top$. We define the *Jacobian* as the *m* by *n* matrix

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \nabla f_1(x)^\top \\ \nabla f_2(x)^\top \\ \vdots \\ \nabla f_m(x)^\top \end{pmatrix}$$

A brief aside: Nonlinear least squares (Ch. 10)

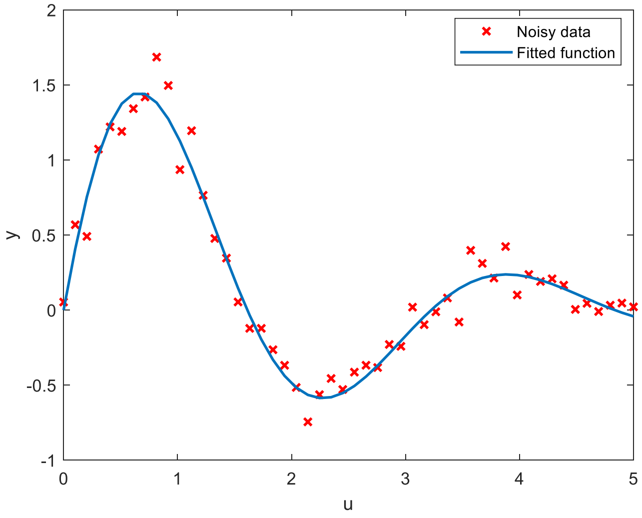
- Consider the following problem: We have a number of (noisy) data

$$(u_1, y_1),\ (u_1, y_1),\ \ldots, (u_m, y_m)$$

and want to fit the function

$$y = \theta_1 e^{\theta_2 u} \sin(\theta_3 u)$$

to the data



-

A brief aside: Nonlinear least squares (Ch. 10)

- Consider the following problem: We have a number of (noisy) data

$$(u_1, y_1),\ (u_1, y_1),\ \ldots, (u_m, y_m)$$

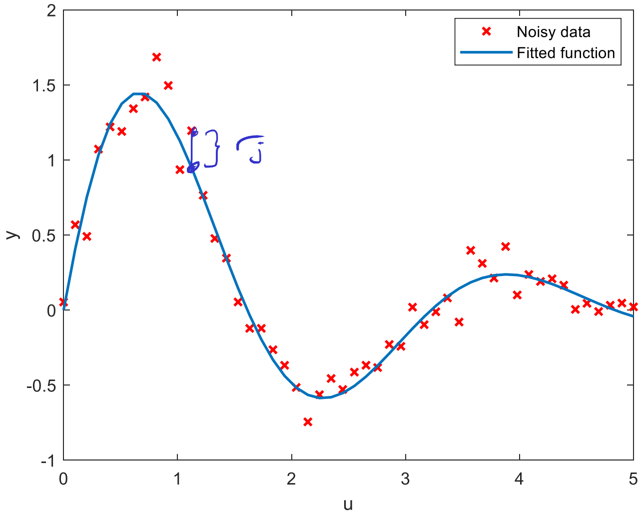
and want to fit the function

$$y = \theta_1 \mathrm{e}^{\theta_2 u} \sin(\theta_3 u)$$

to the data

- (Nonlinear) least squares formulation:

$$\theta = \arg\min_{\theta \in \mathbb{R}^3} \sum_{j=1}^{m} \underbrace{\left(y_j - \theta_1 \mathrm{e}^{\theta_2 u_j} \sin(\theta_3 u_j)\right)^2}_{\text{residual } r_j(\theta)}$$



-

A brief aside: Nonlinear least squares (Ch. 10)

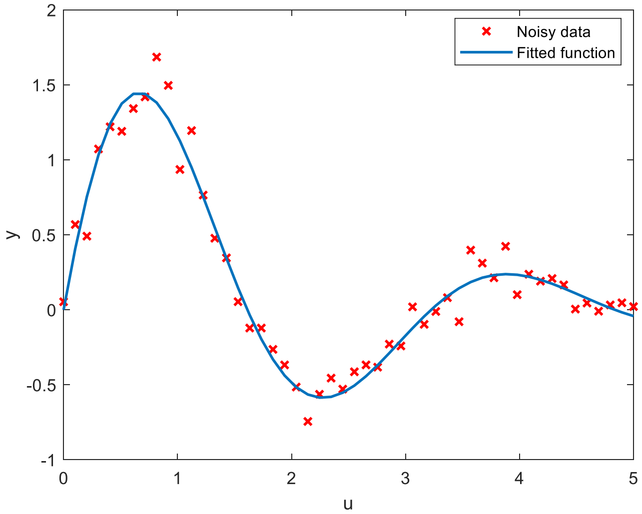
- Consider the following problem: We have a number of (noisy) data

$$(u_1, y_1),\ (u_1, y_1),\ \ldots, (u_m, y_m)$$

and want to fit the function

$$y = \theta_1 \mathrm{e}^{\theta_2 u} \sin(\theta_3 u)$$

to the data



- (Nonlinear) least squares formulation:

$$\theta = \arg\min_{\theta \in \mathbb{R}^3} \sum_{j=1}^{m} \underbrace{\left(y_j - \theta_1 \mathrm{e}^{\theta_2 u_j} \sin(\theta_3 u_j)\right)^2}_{\text{residual } r_j(\theta)}$$

- Generalizations:
  - (Statistical) Machine Learning: Regression, or parametric learning
  - Control theory: System identification (fitting dynamic models to data)

How to solve nonlinear least squares problems

This is an unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j(x)^2$$

(typically, *m*>>*n*)

We can use the linesearch optimization methods of Ch. 3-6!

How to solve nonlinear least squares problems

This is an unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j(x)^2$$

(typically, *m*>>*n*)

We can use the linesearch optimization methods of Ch. 3-6!
If we want to use Newton's method, we need gradient and Hessian of objective function:

- First find Jacobian of *residuals* $r_j(x)$:

$$r(x) = \begin{pmatrix} r_1(x) & r_2(x) & \dots & r_m(x) \end{pmatrix}^\top \qquad J(x) = \begin{pmatrix} \nabla r_1(x)^\top \\ \nabla r_2(x)^\top \\ \vdots \\ \nabla r_m(x)^\top \end{pmatrix}$$

How to solve nonlinear least squares problems

This is an unconstrained optimization problem:

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} \sum_{j=1}^{m} r_j(x)^2$$

(typically, *m*>>*n*)

We can use the linesearch optimization methods of Ch. 3-6!
If we want to use Newton's method, we need gradient and Hessian of objective function:

- First find Jacobian of *residuals* $r_j(x)$:

$$r(x) = \begin{pmatrix} r_1(x) & r_2(x) & \dots & r_m(x) \end{pmatrix}^\top \qquad J(x) = \begin{pmatrix} \nabla r_1(x)^\top \\ \nabla r_2(x)^\top \\ \vdots \\ \nabla r_m(x)^\top \end{pmatrix}$$

- Gradient and Hessian of *objective* $f(x) = \frac{1}{2}\|r(x)\|^2$:

$$\nabla f(x) = \sum_{j=1}^{m} r_j(x) \nabla r_j(x) = J(x)^\top r(x)$$

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla r_j(x) \nabla r_j(x)^\top + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x) = J(x)^\top J(x) + \sum_{j=1}^{m} r_j(x) \nabla^2 r_j(x)$$

Gauss-Newton method

- For these problems, a good approximation of the Hessian is

$$\nabla^2 f(x) = J(x)^\top J(x) + \cancel{\sum_{j=1}^{m} r_j(x)\nabla^2 r_j(x)} \approx J(x)^\top J(x)$$

- The Gauss-Newton method for nonlinear least squares problems: Use Newton's method with this Hessian approximation
  - Note: Only first-order derivatives are needed!
  - Make it work far from solution: Use linesearch with Wolfe-conditions, etc. (same as before)

- (Using the same approximation with trust-region instead of linesearch is the *Levenberg-Marquardt* algorithm – implemented in Matlab-function `lsqnonlin`)

Linear least squares

- Say you want to fit a polynomial $y = \theta_1 + \theta_2 u + \theta_3 u^2 + \ldots$ to data $(u_1, y_1),\ (u_1, y_1),\ \ldots, (u_m, y_m)$
- Define $x = (\theta_1\ \theta_2\ \theta_3\ \ldots)^\top$ and formulate least squares optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2}\sum_{j=1}^{m} r_j(x)^2 = \frac{1}{2}\sum_{j=1}^{m}\left(y_j - (1\ u_j\ u_j^2 \ldots)\, x\right)^2 = \frac{1}{2}\|y - Ax\|^2$$

Linear in parameters!

where the *regressor matrix A* is

$$A = \begin{pmatrix} 1 & u_1 & u_1^2 & \ldots \\ 1 & u_2 & u_2^2 & \ldots \\ \vdots & \vdots & \vdots & \\ 1 & u_m & u_m^2 & \ldots \end{pmatrix}$$

- Easy to show that the solution is given from

$$A^\top A x = A^\top y \quad \Rightarrow \quad x = \left(A^\top A\right)^{-1} A^\top y$$

  - Solve by Cholesky or (better) QR (see book 10.2). Matlab: `x = A\y`.

Observe: The Gauss-Newton approximation $A^\top A$ is exact for linear problems!

Nonlinear equations

Nonlinear equations

$$x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$$

$$\left.\begin{aligned} r_1(x) &= 0 \\ r_2(x) &= 0 \\ &\vdots \\ r_n(x) &= 0 \end{aligned}\right\} \quad r(x) = 0, \quad r(x) := \begin{pmatrix} r_1(x) \\ r_2(x) \\ \vdots \\ r_n(x) \end{pmatrix}$$

Note: # eqs = # vars

Ex.

$$r(x) = \begin{pmatrix} x_2^2 - 1 \\ \sin(x_1) - x_2 \end{pmatrix} = 0$$

$\leftarrow x_2 = \pm 1$

$\leftarrow x_1 = \pm \frac{\pi}{2}, \pm \frac{3\pi}{2}, \ldots$

3 cases:

- No solution
- One solution
- Many solutions

Problem to solve: $\boxed{\text{Find } x \text{ s.t. } r(x) = 0}$

Nonlinear equations

Find x s.t. $r(x) = 0$

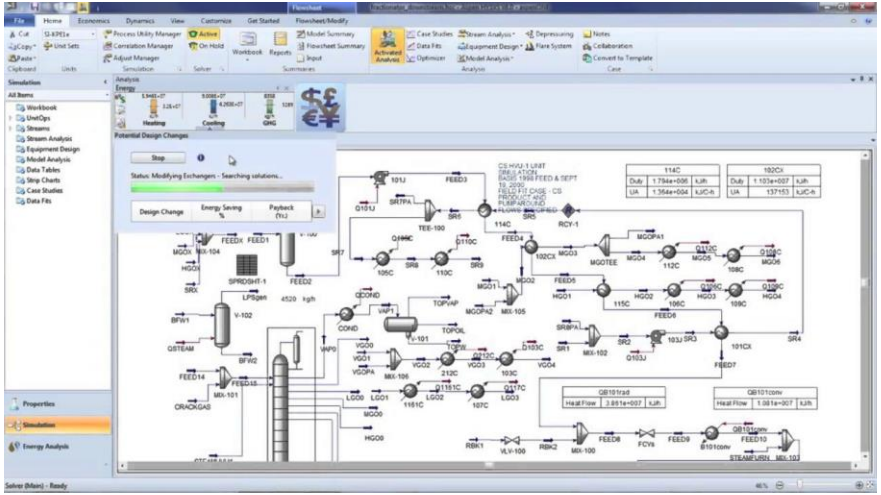
Compare unconstrained opt. $\min_x f(x)$

Find x s.t. $\underbrace{\nabla f(x) = 0}_{\text{Nec. cond. of opt.}}$

Why study nonlinear equations? – Examples

- Given nonlinear system $\dot{x} = f(x)$, the steady state is found by solving $f(x) = 0$

- Flowsheet analysis in chemical/process engineering (steady state simulators, solving mass- and energy balances)



Aspen Hysys

- Simulation methods (ModSim): For implicit Runge-Kutta, we need to solve nonlinear equations

- Newton's method for nonlinear equations is important for SQP methods (next lecture)

Derivation of Newton's method for nonlinear equations

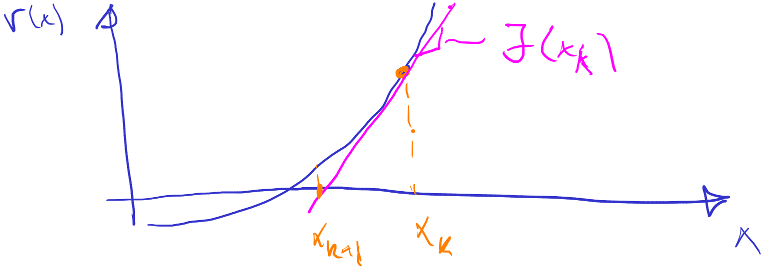
Multidimensional Taylor variant:

$$r(x_k+p) = r(x_k) + \int_0^1 \underbrace{J(x_k+tp)}_{\text{Approximate with } J(x_k)} p \, dt$$

Model of $r(x_k+p)$

$$r(x_k+p) \approx M_k(p) = r(x_k) + J(x_k)\,p$$

$$M_k(p) = 0 \Rightarrow \boxed{p_k = -J^{-1}(x_k)\, r(x_k)}$$

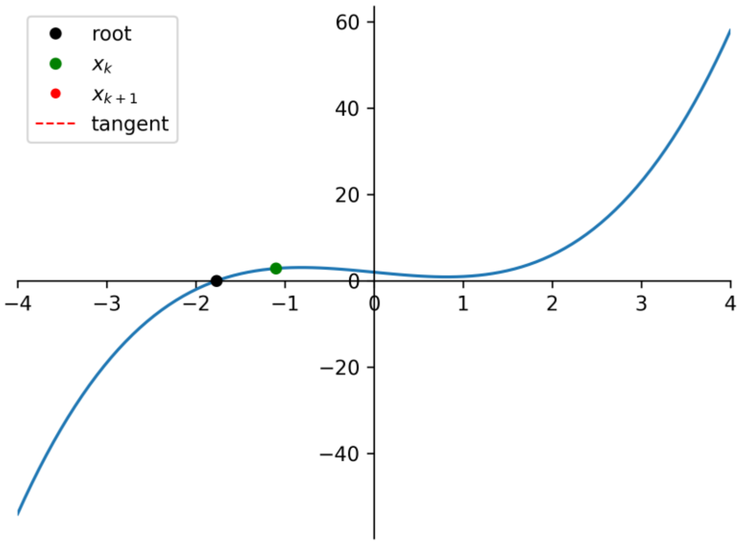


$$x_{k+1} = x_k + p_k$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

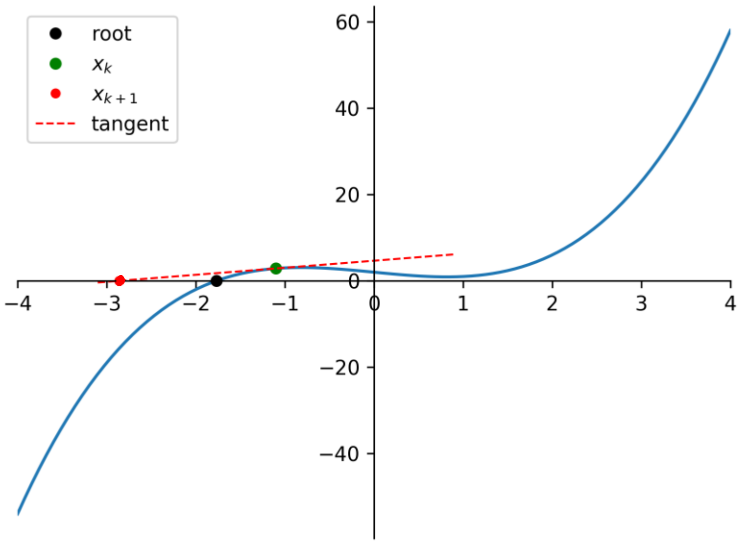


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

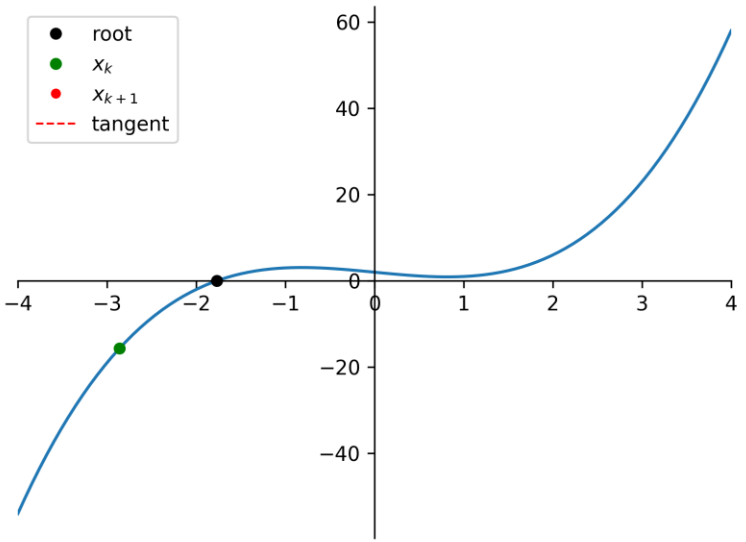


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

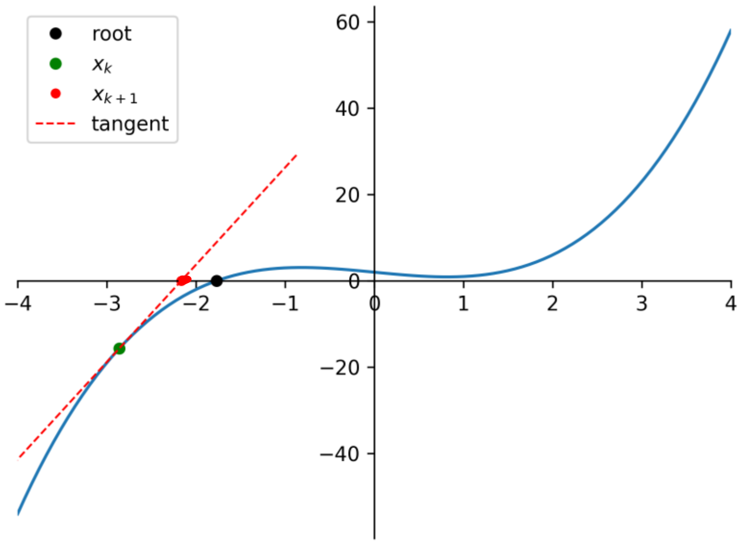


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

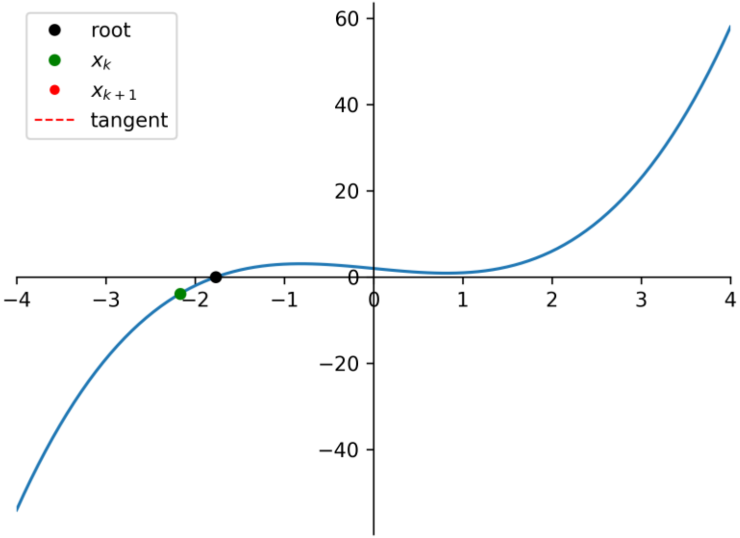


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

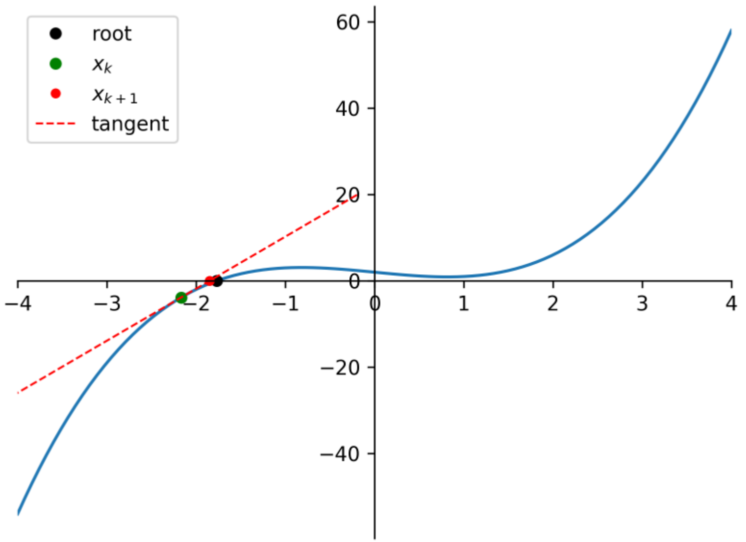


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

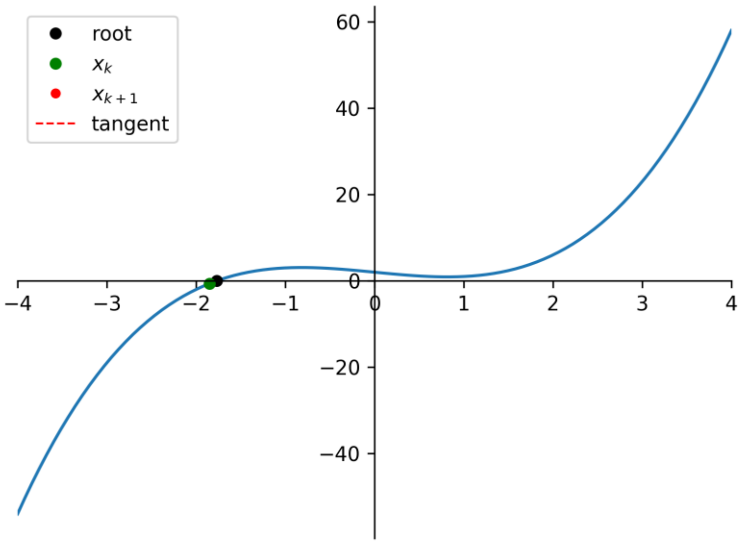


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

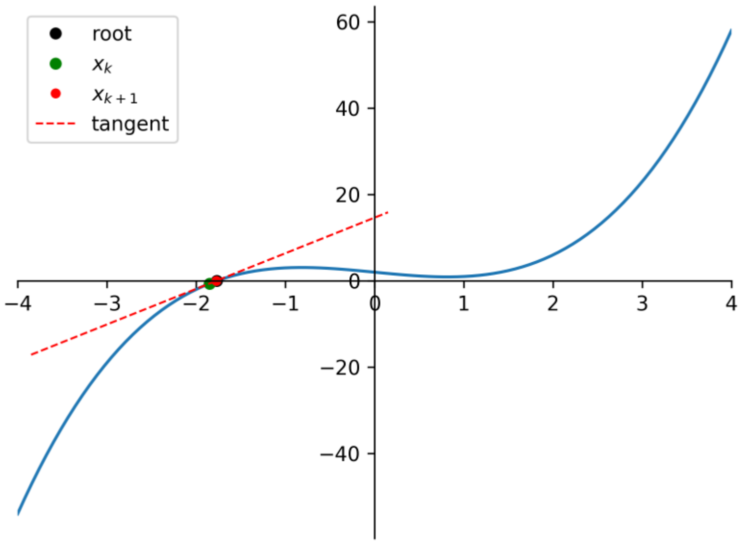


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

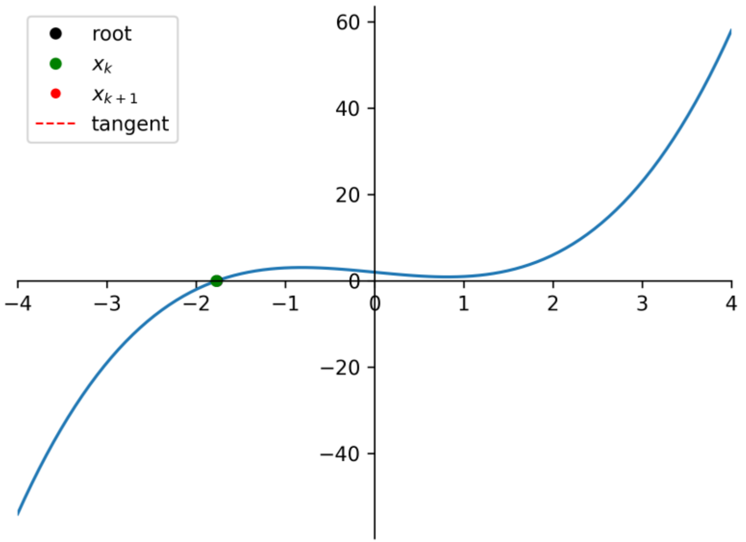


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

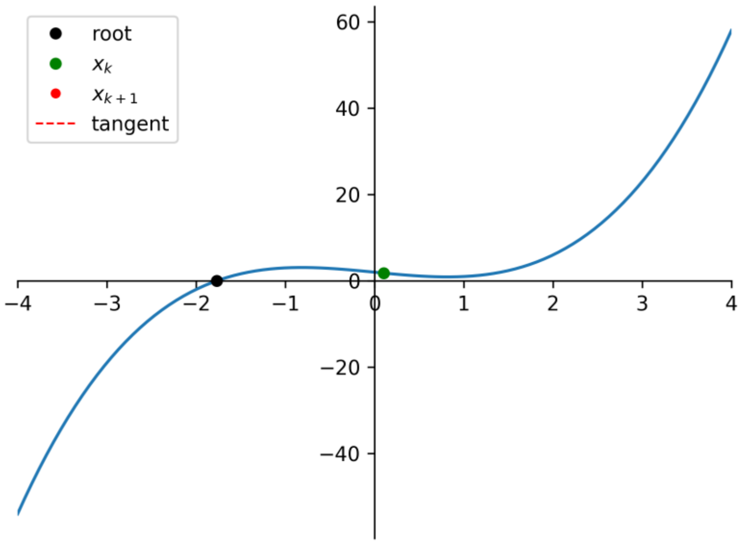


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

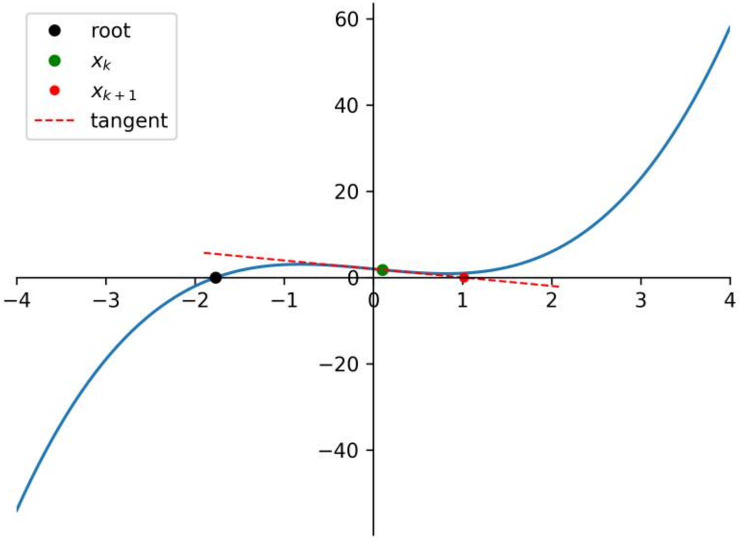


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

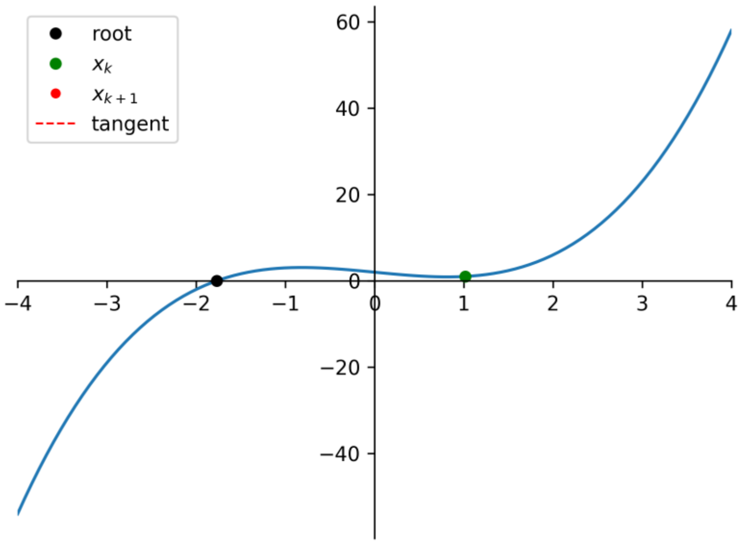


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

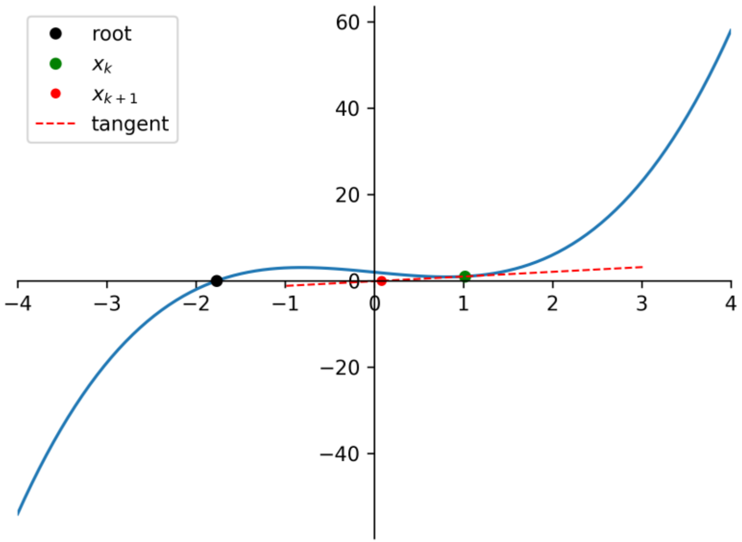


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

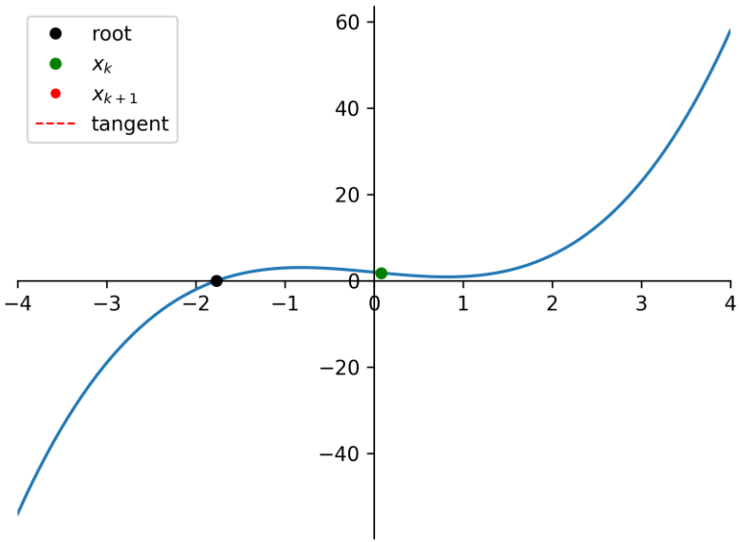


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

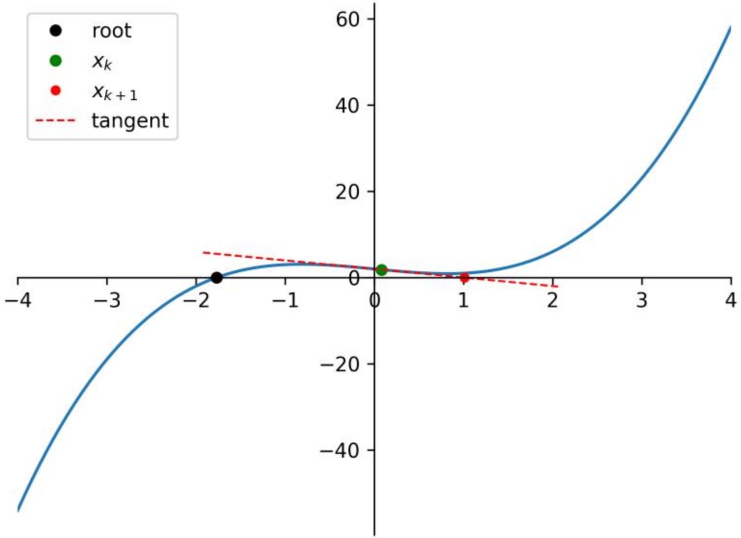


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

$$\text{Scalar case: } x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$$

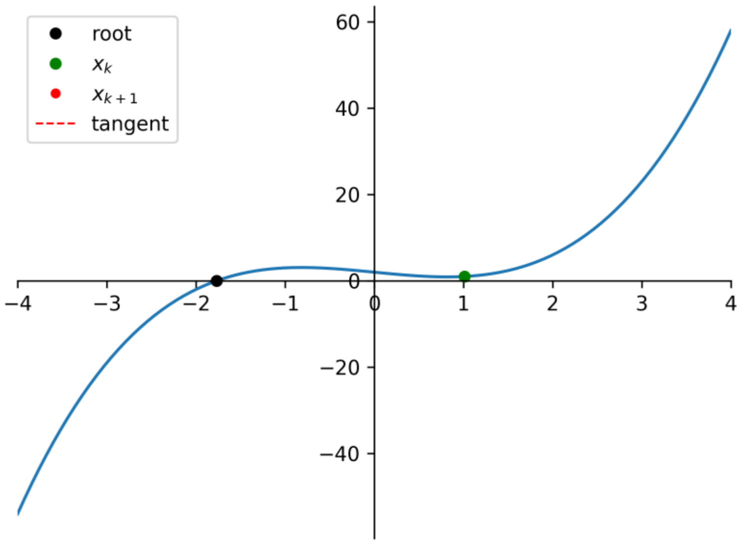


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

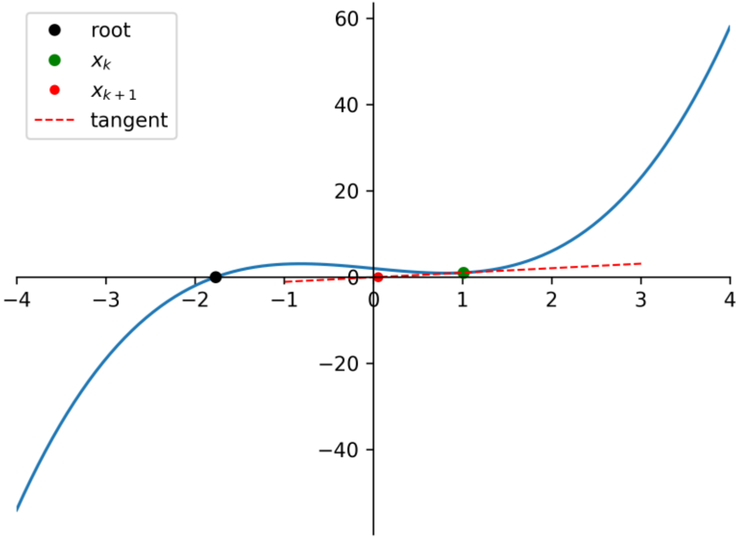


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$

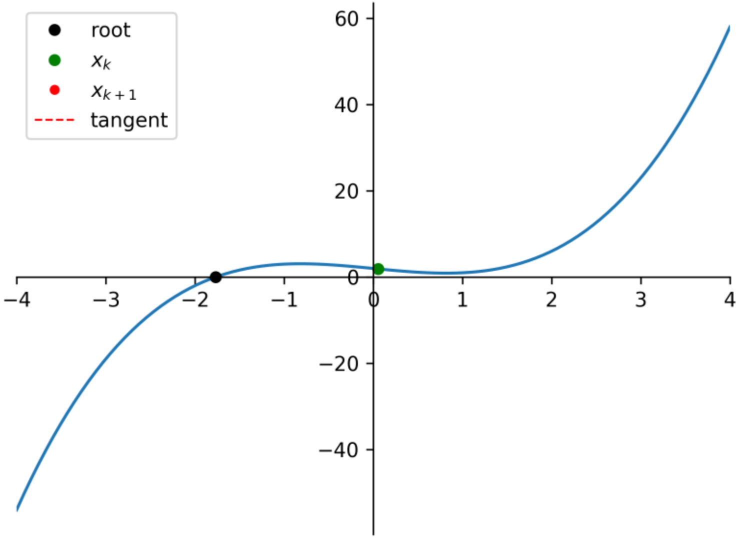


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

$$x_{k+1} = x_k + p_k, \quad p_k = -J(x_k)^{-1} r(x_k)$$

Scalar case: $x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}$



$$r(x) = x^3 - 2x + 2$$

Newton's method for nonlinear equations (Alg. 11.1)

Alg. 11.1

Choose x_0, $\varepsilon > 0$, $k_{max} > 0$

$k = 0$

while $\|r(x_k)\| > \varepsilon$ AND $k < k_{max}$

  Solve $J(x_k)\, p_k = -r(x_k)$ ← $[LU]$

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

  $k = k + 1$

end

$Ax = b$

$L\underbrace{(Ux)}_{y} = b$

Newton's method for nonlinear equations (Alg. 11.1)

Why called Newton?

$r(x) = 0$

Recall $\min_x f(x) \to$ Nec. cond. $\underbrace{\nabla f(x)}_{} = 0$

Jacobian of ∇f:

$J(x) = \nabla^2 f(x)$

$$p_k = -J^{-1}(x_k)\, r(x_k) = \underbrace{-\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)}_{\text{Newton direction for nonlinear opt.}}$$

Thm 11.2: Convergence of Newton's method is quadratic

(when Jacobian is non-singular)

Proof: (scalar case)

$$x_{k+1} = x_k - \frac{r(x_k)}{r'(x_k)}, \quad r'(x) \neq 0$$

$$\boxed{\text{Taylor: } r(x_k + p) = r(x_k) + r'(x_k) \cdot p + \frac{p^2}{2} r''(x_k + tp), \ t \in (0,1)}$$

$$r(x_{k+1}) = r(x_k + p) = r\left(x_k - \frac{r(x_k)}{r'(x_k)}\right), \quad p = -\frac{r(x_k)}{r'(x_k)}$$

$$= \cancel{r(x_k)} + \cancel{r'(x_k)}\left(-\frac{\cancel{r(x_k)}}{\cancel{r'(x_k)}}\right) + \frac{r(x_k)^2}{2\, r'(x_k)^2} r''(x_k + tp)$$

$$\Rightarrow \boxed{r(x_{k+1}) \leq M\, r(x_k)^2}, \quad M > \frac{r''(x_k + tp)}{2\, r'(x_k)^2}$$

↳ quadratic convergence: very fast!

Practical issues with Newton's method: Jacobian

- $J(x_k)$ may become singular (far from a solution)
- $J(x_k)$ may be (too) expensive to compute

Remedy: Use Broyden's method, a Quasi-Newton-type update formula for $J(x)$

$$B_k \approx J(x_k)$$

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)^T s_k}{s_k^T s_k}, \quad y_k = r(x_{k+1}) - r(x_k), \quad s_k = x_{k+1} - x_k$$

Practical issues with Newton's method: Merit function

(Or: How to ensure convergence when far from solution?)

Use line search!

Define merit function

$$f(x) = \frac{1}{2}\|r(x)\|_2^2 = \frac{1}{2}\sum_{i=1}^{n} r_i(x)^2$$

Note:

$$\nabla f(x) = \sum_{i=1}^{n} r_i(x)\nabla r_i(x) = J^T(x)\, r(x)$$

When $J(x_k)$ is non-singular, $p_k = -J^{-1}(x_k)\, r(x_k)$ is a descent direction for $f(x)$:

$$p_k^T \nabla f(x_k) = -r^T(x_k)\,\cancel{J^{-T}(x_k)\, J(x_k)}\, r(x_k) = -\|r(x_k)\|^2 < 0$$

Line search: $x_{k+1} = x_k + \alpha_k p_k$, α_k satisfies Wolfe cond. on $f(x)$