



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

$$\nabla f(x) = \left(\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \cdots \quad \frac{\partial f}{\partial x_n} \right)^\top$$

- Say $f(x) = (f_1(x) \quad f_2(x) \quad \cdots \quad f_m(x))^\top$. 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)

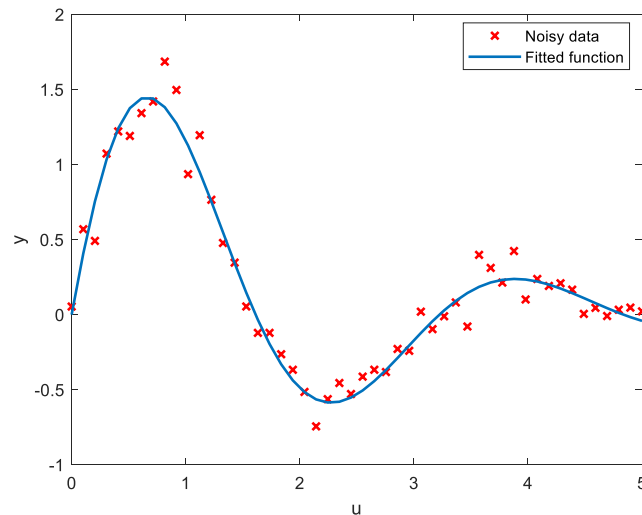
- Consider the following problem: We have a number of (noisy) data

$$(u_1, y_1), (u_2, y_2), \dots, (u_m, y_m)$$

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

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

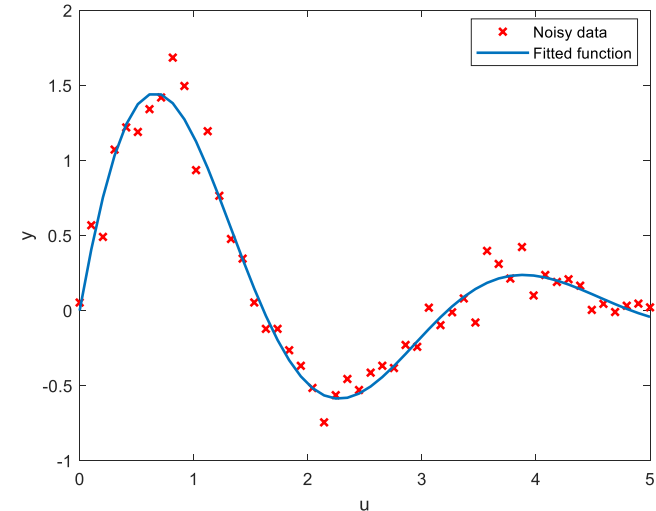
and want to fit the function

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

to the data

- (Nonlinear) least squares formulation:

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



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

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

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

and want to fit the function

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

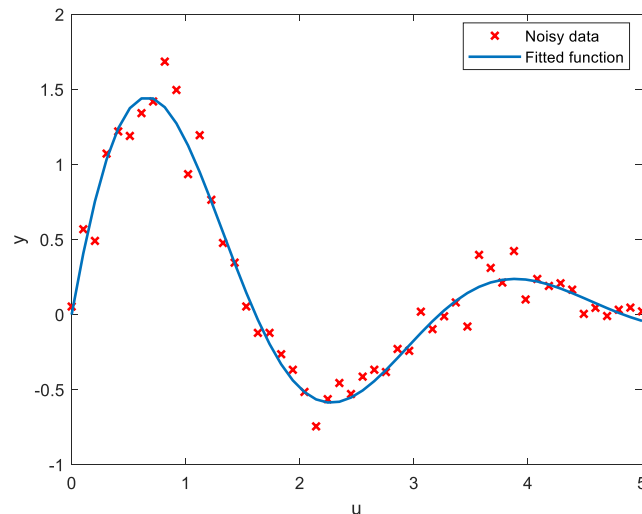
to the data

- (Nonlinear) least squares formulation:

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

- 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**:

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

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

How to solve nonlinear least squares problems

This is an **unconstrained optimization problem**:

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

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** $r_j(x)$:

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

How to solve nonlinear least squares problems

This is an **unconstrained optimization problem**:

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

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** $r_j(x)$:

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

- 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

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

- 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 $y = \theta_1 + \theta_2 u + \theta_3 u^2 + \dots$ to data $(u_1, y_1), (u_1, y_1), \dots, (u_m, y_m)$
- Define $x = (\theta_1 \ \theta_2 \ \theta_3 \ \dots)^\top$ and formulate least squares optimization problem

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

where the *regressor matrix* A is

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

Linear in parameters!

- Easy to show that the solution is given from

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

- Solve by Cholesky or (better) QR (see book 10.2). Matlab: $\mathbf{x} = \mathbf{A} \backslash \mathbf{y}$.

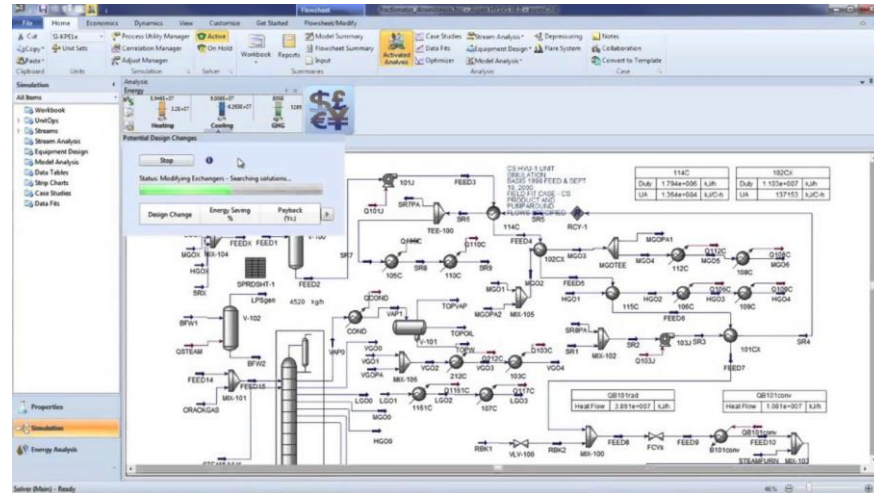
Observe: The Gauss-Newton approximation $A^\top A$ is exact for linear problems!

Nonlinear equations

Nonlinear equations

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

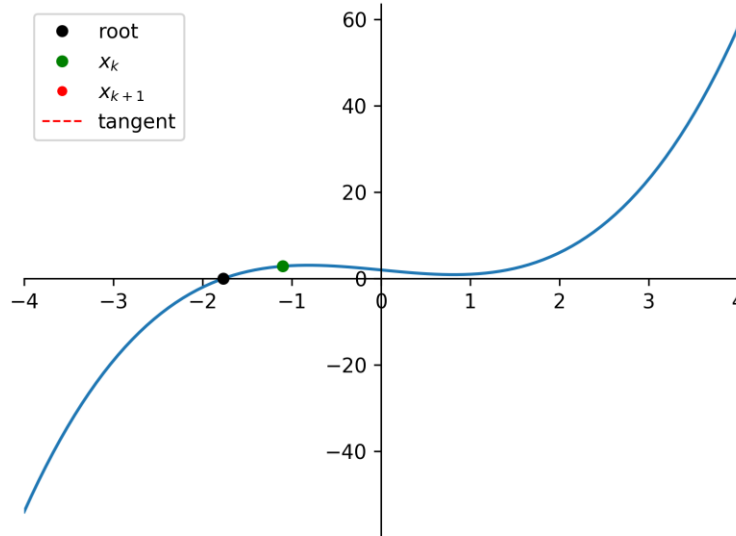
- Simulation methods (ModSim): For implicit Runge-Kutta, we need to solve nonlinear equations
- Newton's method for nonlinear equations is important for SQP methods (next lecture)

Derivation of Newton's method for nonlinear equations

Newton's Method (aka Newton-Raphson method)

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

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

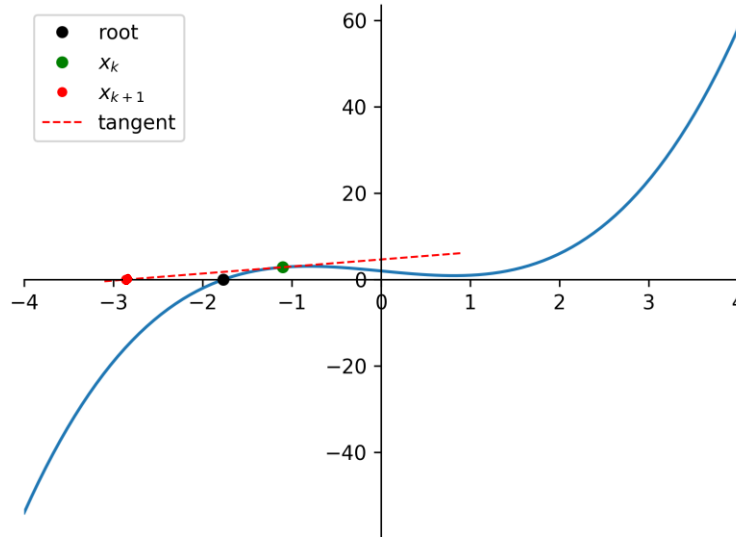


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

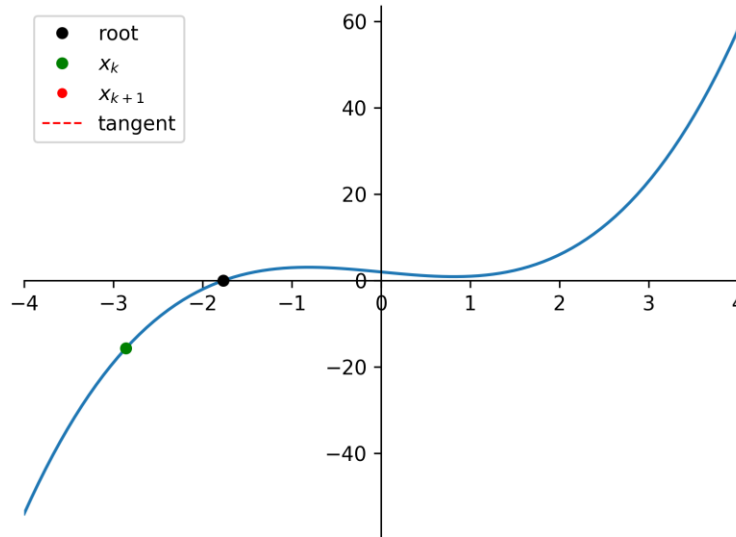


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

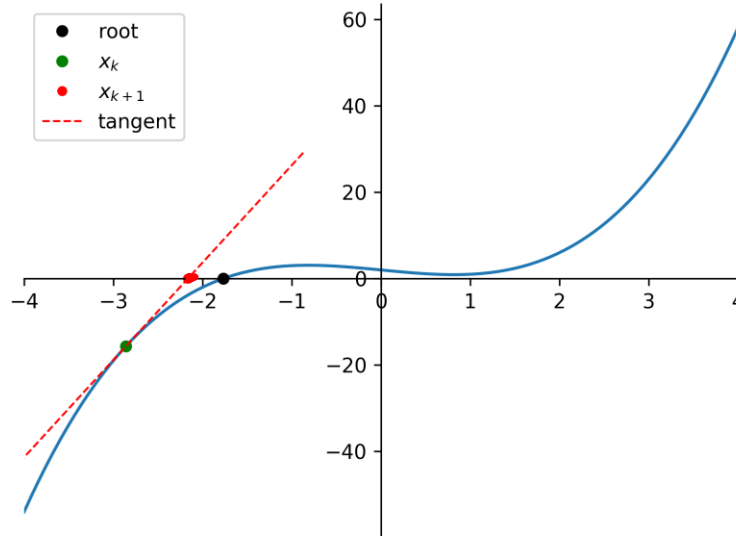


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

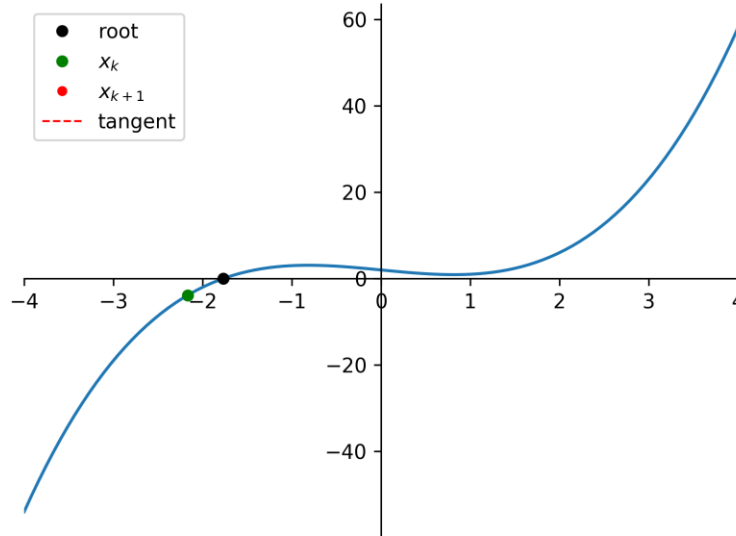


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

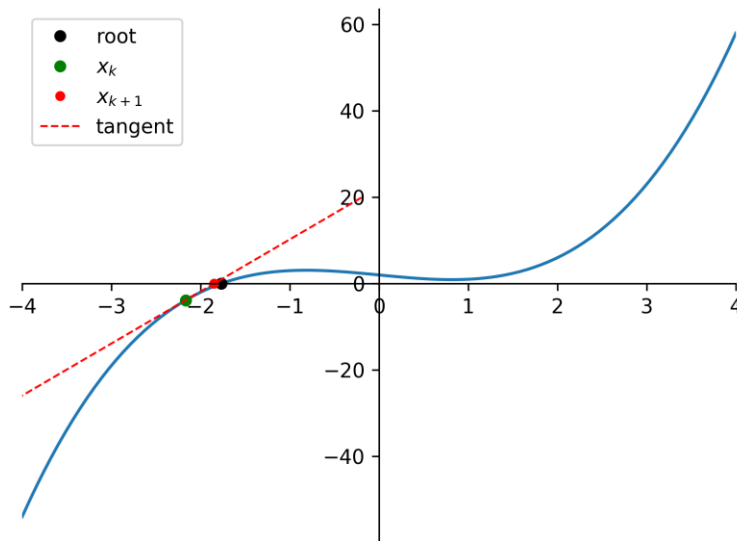


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

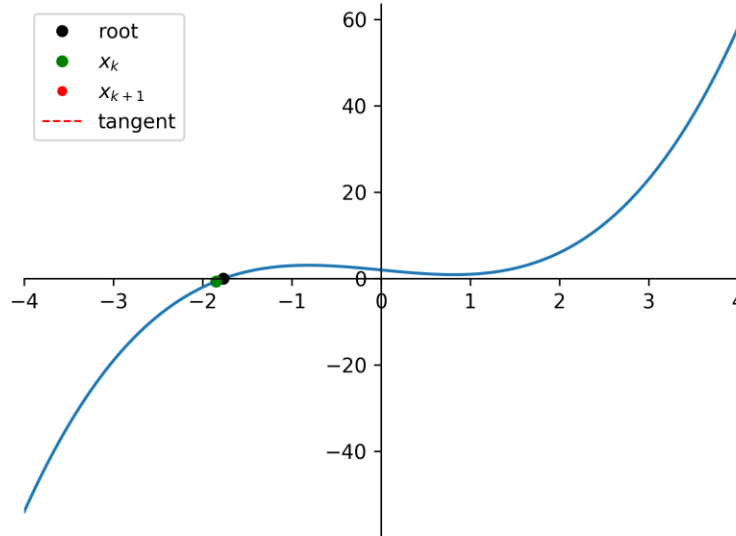


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

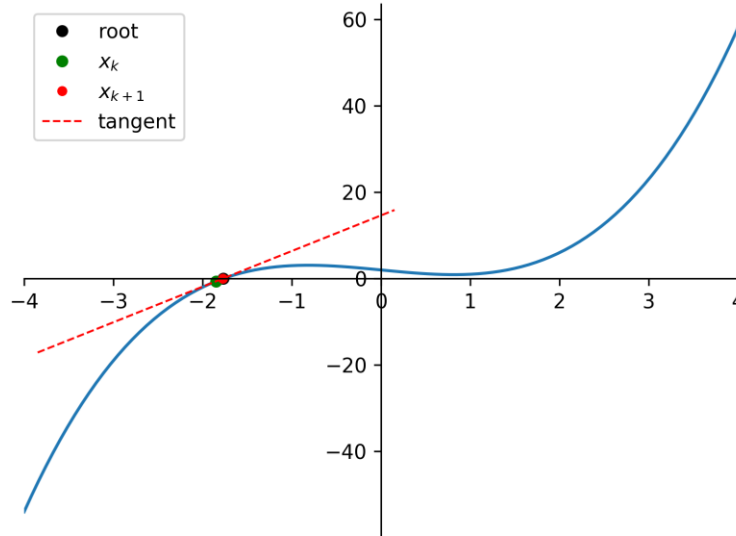


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

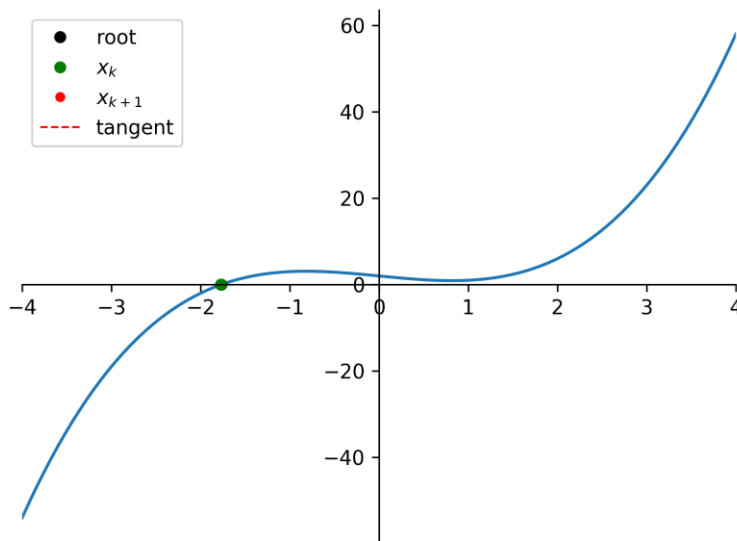


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

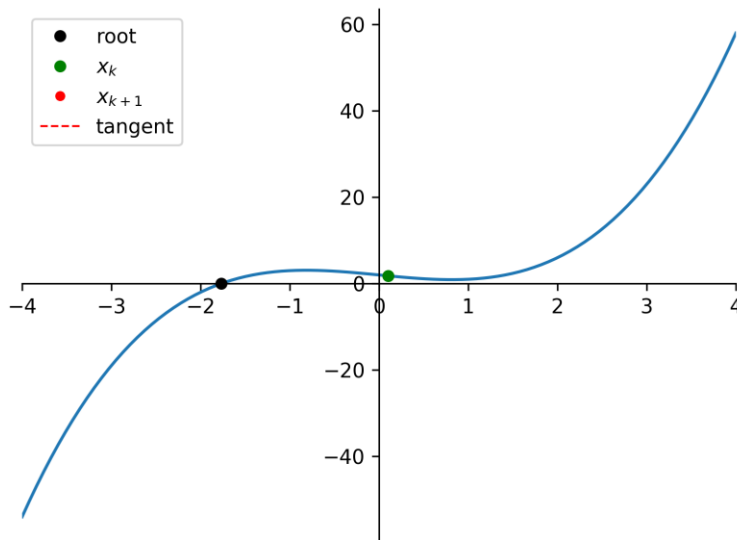


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

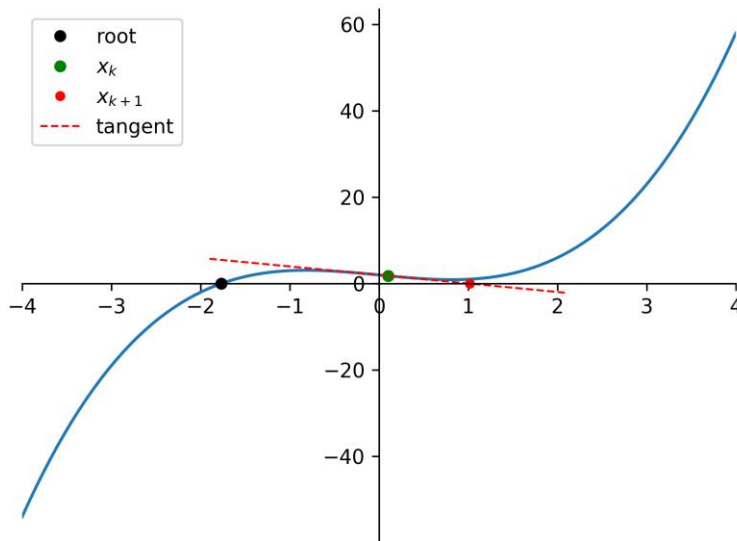


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

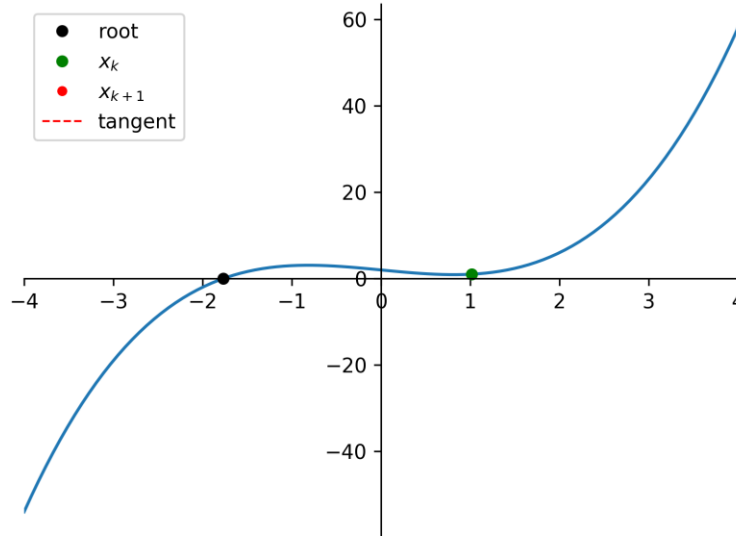


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

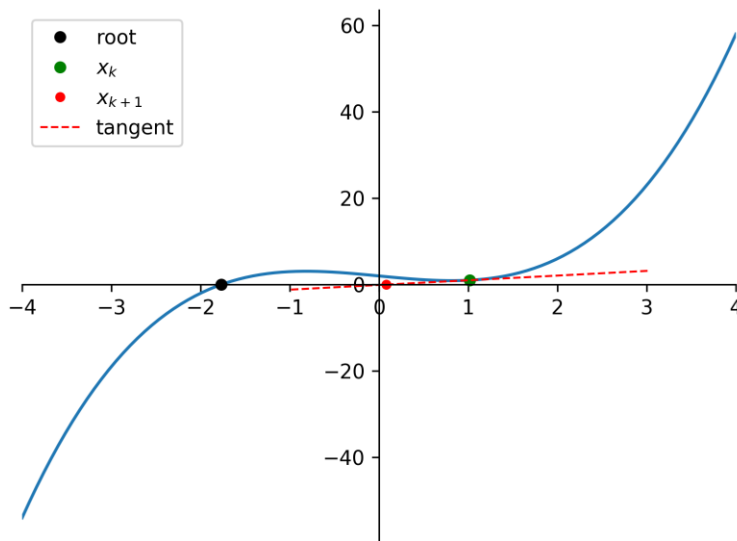


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

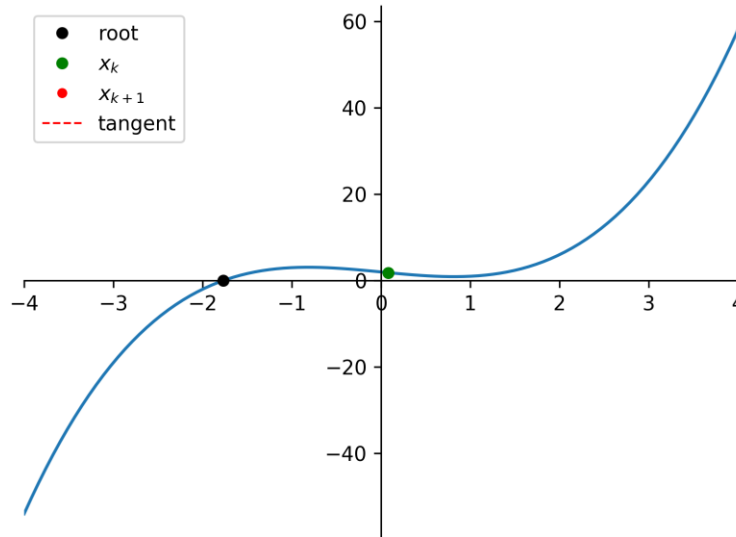


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

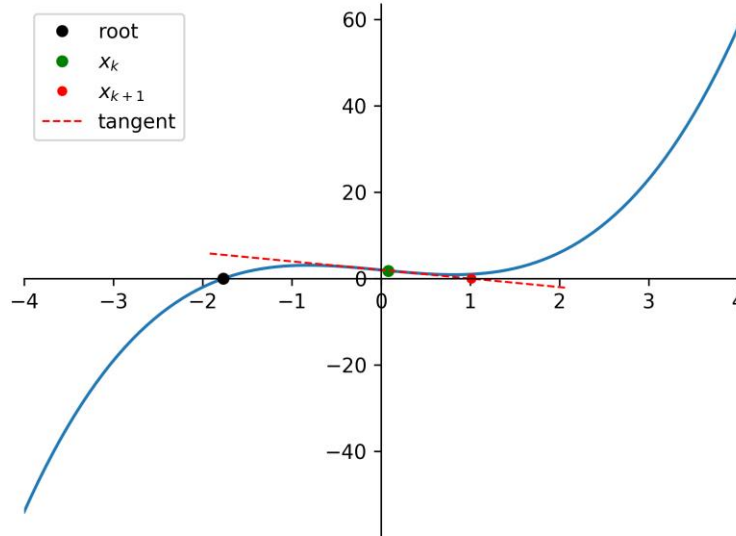


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

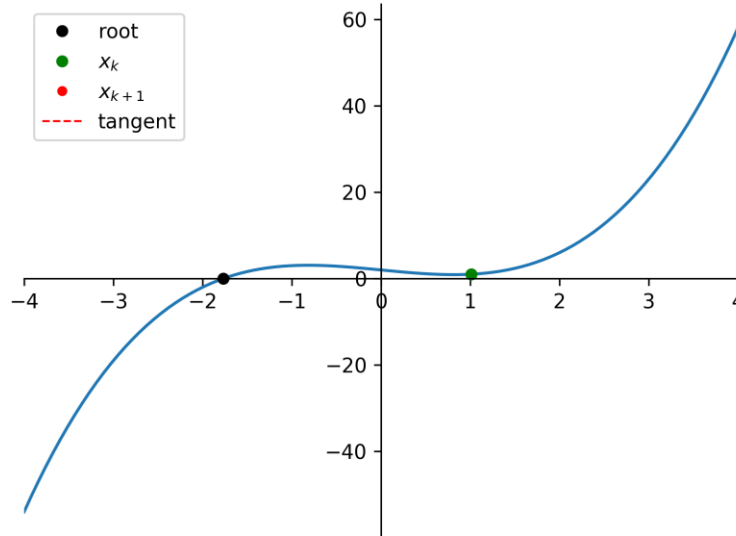


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

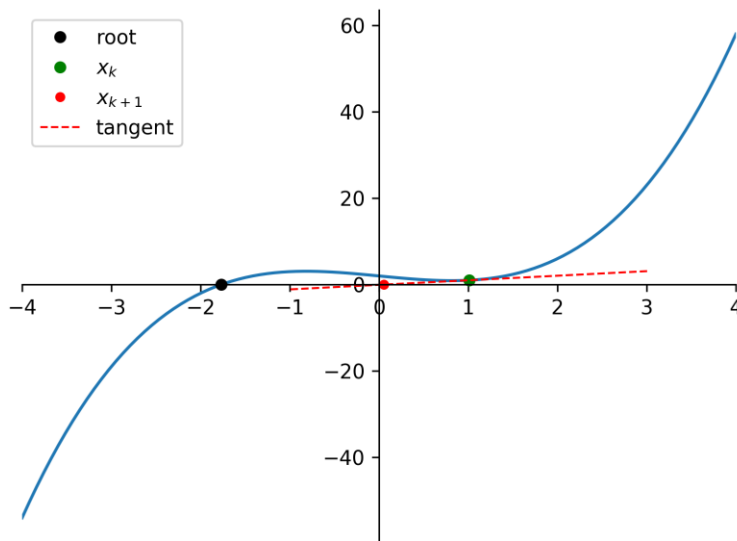


$$r(x) = x^3 - 2x + 2$$

Newton's Method (aka Newton-Raphson method)

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

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

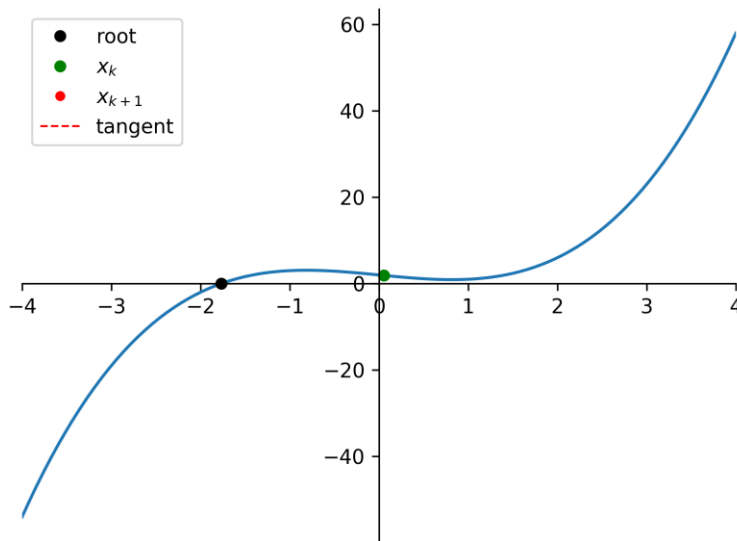


$$r(x) = x^3 - 2x + 2$$

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 for nonlinear equations (Alg. 11.1)

Thm 11.2: Convergence of Newton's method is quadratic

(when Jacobian is non-singular)

Practical issues with Newton's method: Jacobian

Practical issues with Newton's method: Merit function

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