Lecture 17: Newton's method for solving nonlinear equations (Ch. 11)

- A brief summary of Ch. 10
- **Nonlinear equations**
- Newton's method for solving nonlinear equations (Ch. 11)
- 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)

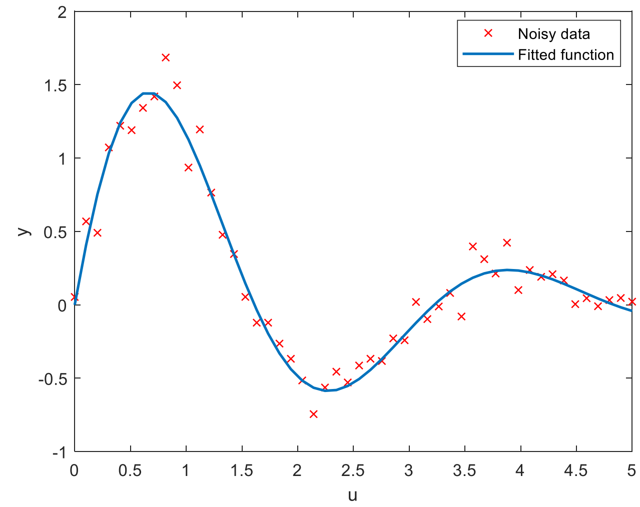
- Consider the following problem: We have a number of (noisy) data

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

and want to fit the function

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

to the data



- (Nonlinear) least squares formulation:

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

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

How to solve nonlinear least squares

Formulation as unconstraind optimization problem (m>>n):

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

Say we want to use Newton's method. We need gradient and Hessian of objective function:

- First find gradient 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 of *objective* $f(x) = \frac{1}{2}\|r(x)\|^2$:

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

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

Gauss-Newton method

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

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

- 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, 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 least squares optimization problem

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

  where the regressor matrix *A* is

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

- Easy to show that solution is given from

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

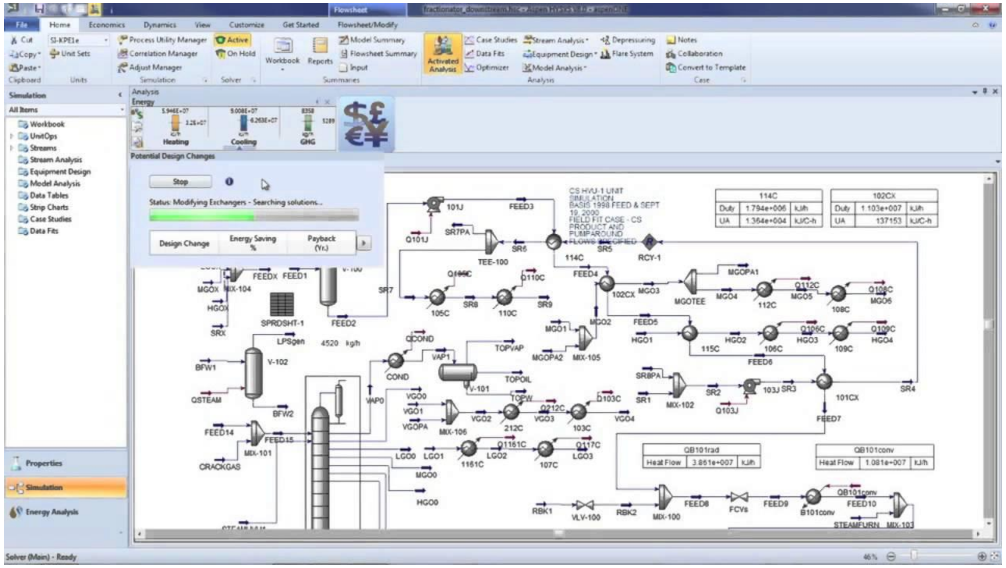
  – Solve by Cholesky or (better) QR (see book 10.2)

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

Why study nonlinear equations

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

- Flowsheet analysis in chemical engineering (steady state simulators)



- ModSim: Using implicit Runge-Kutta, we need to solve nonlinear equations
- Newton's method for nonlinear equations is important for deriving/analysing SQP methods (next time)

Newton’s method