Technical Computing for the Earth Sciences, Lecture 2:

Solving some linearizable inverse problems for the Earth sciences

EARS 80.03

Linearizable inverse problems

You're already solving a nonlinear-but-linearizable inverse problem:

Project 2!

- We'll just review some of the concepts here

Linear inverse problems

Generalized linear inverse problem: you have

- some data y
- an equation of the form

  $$y = f(x,\ parameters...)\ [+\ g(\mathrm{x},\ parameters...) + ...]$$

  where f, g are nonlinear functions of x and one or more parameters for which we wish to invert

- Can't solve in one step any more, have to iterate!
- The derivative of a nonlinear function gives us one simple way to approximate a nonlinear function with a linear one (tangent line) -> *Newton's method*

Newton's method

18.2.1 Basic Gauss–Newton algorithm

The idea behind the Gauss–Newton algorithm is simple: We alternate between finding an affine approximation of the function f at the current iterate, and then solving the associated linear least squares problem to find the next iterate. This combines two of the most powerful ideas in applied mathematics: *Calculus* is used to form an affine approximation of a function near a given point, and *least squares* is used to compute an approximate solution of the resulting affine equations.

We now describe the algorithm in more detail. At each iteration k, we form the affine approximation $\hat{f}$ of f at the current iterate $x^{(k)}$, given by the Taylor approximation

$$\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)}), \qquad (18.5)$$

where the $m \times n$ matrix $Df(x^{(k)})$ is the Jacobian or derivative matrix of f (see §8.2.1 and §C.1). The affine function $\hat{f}(x; x^{(k)})$ is a very good approximation of $f(x)$ provided x is near $x^{(k)}$, *i.e.*, $\|x - x^{(k)}\|$ is small.

The next iterate $x^{(k+1)}$ is then taken to be the minimizer of $\|\hat{f}(x; x^{(k)})\|^2$, the norm squared of the affine approximation of f at $x^{(k)}$. Assuming that the derivative matrix $Df(x^{(k)})$ has linearly independent columns (which requires $m \geq n$), we have

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)})\right)^{-1} Df(x^{(k)})^T f(x^{(k)}). \qquad (18.6)$$

This iteration gives the basic Gauss–Newton algorithm.

Algorithm 18.1 BASIC GAUSS–NEWTON ALGORITHM FOR NONLINEAR LEAST SQUARES

given a differentiable function $f : \mathbf{R}^n \to \mathbf{R}^m$, an initial point $x^{(1)}$.

For $k = 1, 2, \ldots, k^{\max}$

1. *Form affine approximation at current iterate using calculus.* Evaluate the Jacobian $Df(x^{(k)})$ and define

$$\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)}).$$

2. *Update iterate using linear least squares.* Set $x^{(k+1)}$ as the minimizer of $\|\hat{f}(x; x^{(k)})\|^2$,

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)})\right)^{-1} Df(x^{(k)})^T f(x^{(k)}).$$

Newton's method

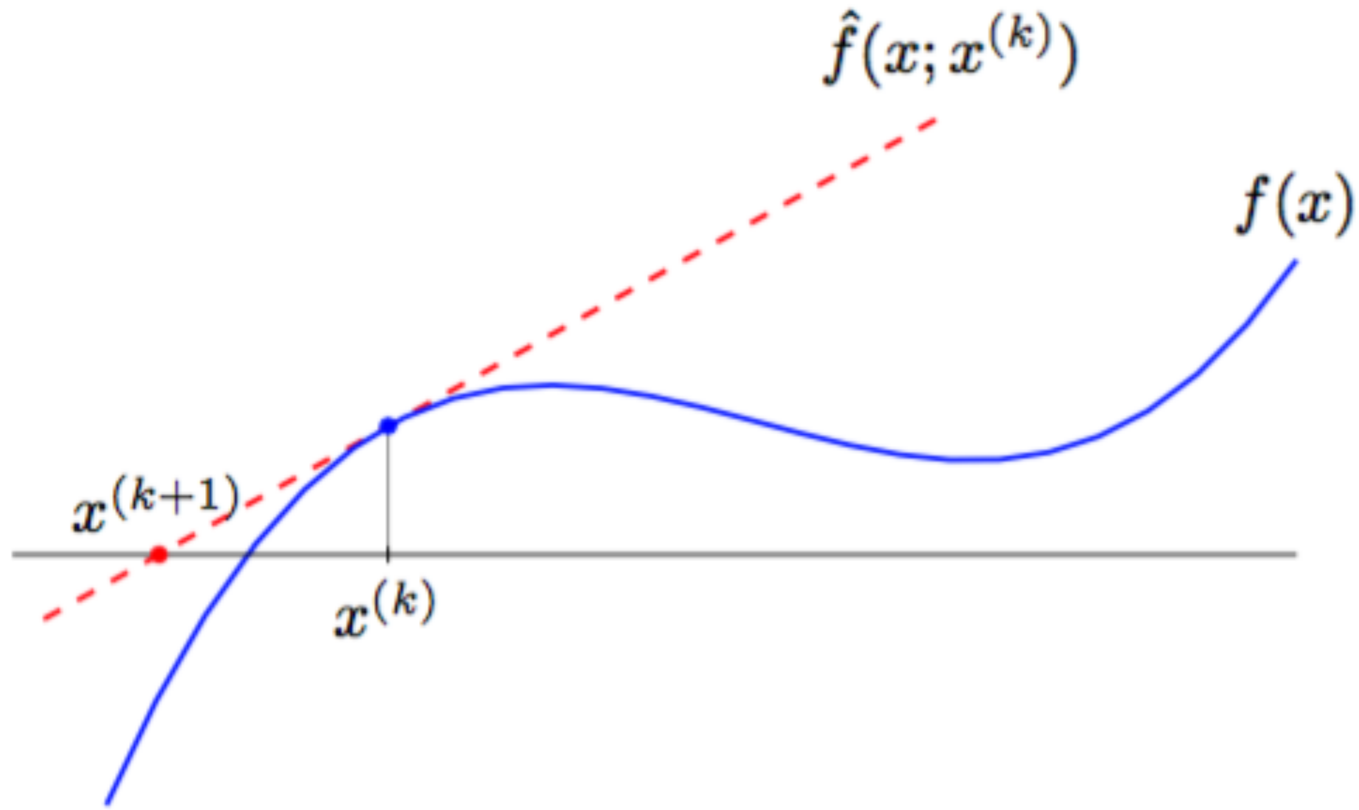

Figure 18.2 One iteration of the Newton algorithm for solving an equation $f(x) = 0$ in one variable.

Newton’s method

Newton's method

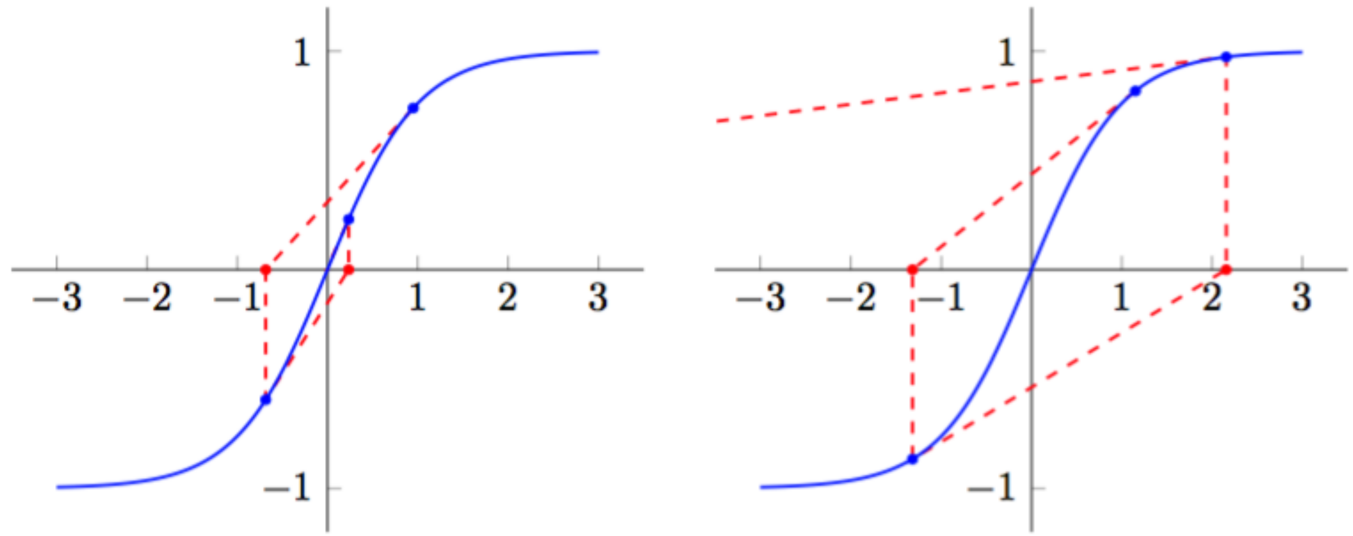

Figure 18.3 The first iterations in the Newton algorithm for solving $f(x) = 0$, for two starting points: $x^{(1)} = 0.95$ and $x^{(1)} = 1.15$.

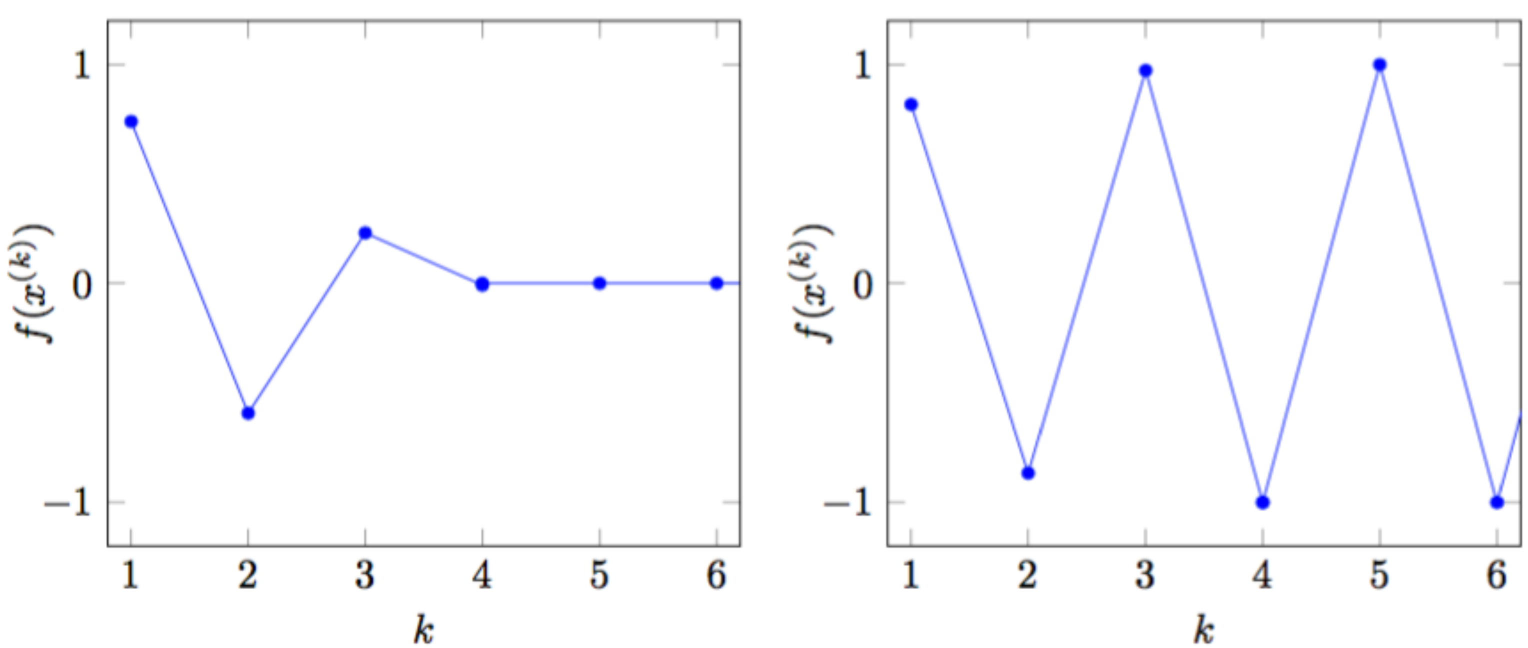

Figure 18.4 Value of $f(x^{(k)})$ versus iteration number k for Newton's method in the example of figure 18.3, started at $x^{(1)} = 0.95$ and $x^{(1)} = 1.15$.

Some more things to think about

- If we’re doing this in multiple dimensions, will there neccessarily be one point (x_0, y_0, z_0, etc.) that satisfies all the linear approximations at once?

Some more things to think about

- If we're doing this in multiple dimensions, will there neccessarily be one point (x_0, y_0, z_0, etc.) that satisfies all the linear approximations at once? **No. If there were, we wouldn't have to iterate!**

- What do we want to find instead?

Some more things to think about

- If we're doing this in multiple dimensions, will there neccessarily be one point (x_0, y_0, z_0, etc.) that satisfies all the linear approximations at once? **No. If there were, we wouldn't have to iterate!**

- What do we want to find instead? **The least-squares approximation!**

- So linear algebra saves the day again, because that's exactly what our left-inverse / "pseudoinverse" already gives us if there isn't an exact solution!

Some more things to think about

- If we're doing this in multiple dimensions, will there neccessarily be one point (x_0, y_0, z_0, etc.) that satisfies all the linear approximations at once? **No. If there were, we wouldn't have to iterate!**

- What do we want to find instead? **The least-squares approximation!**

- So linear algebra saves the day again, because that's exactly what our left-inverse / "pseudoinverse" already gives us if there isn't an exact solution!

- Newton's method is the most approachable, but there are other more complicated methods for linearizing nonlinear inverse problems, e.g. *Levenberg–Marquardt* (discussed in B&V 18.3)