Jim Lambers
MAT 419/519
Summer Session 2011-12
Lecture 11 Notes

These notes correspond to Section 3.4 in the text.

Broyden's Method

One of the drawbacks of using Newton's Method to solve a system of nonlinear equations $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ is the computational expense that must be incurred during each iteration to evaluate the partial derivatives of $\mathbf{g}$ at $\mathbf{x}^{(k)}$, and then solve a system of linear equations involving the resulting Jacobian matrix. The algorithm does not facilitate the re-use of data from previous iterations, and in some cases evaluation of the partial derivatives can be unnecessarily costly.

An alternative is to modify Newton's Method so that *approximate* partial derivatives are used, since the slightly slower convergence resulting from such an approximation is offset by the improved efficiency of each iteration. However, simply replacing the analytical Jacobian matrix $J_{\mathbf{g}}(\mathbf{x})$ of $\mathbf{g}$ with a matrix consisting of finite difference approximations of the partial derivatives does not do much to reduce the cost of each iteration, because the cost of solving the system of linear equations is unchanged.

However, because $J_{\mathbf{g}}(\mathbf{x})$ consists of the partial derivatives evaluated at an element of a convergent sequence, intuitively Jacobian matrices from consecutive iterations are "near" one another in some sense, which suggests that it should be possible to cheaply update an approximate Jacobian matrix from iteration to iteration, in such a way that the *inverse* of the Jacobian matrix, which is what is really needed during each Newton iteration, can be updated efficiently as well.

This is the case when an $n \times n$ matrix B has the form

$$B = A + \mathbf{u} \otimes \mathbf{v},$$

where $\mathbf{u}$ and $\mathbf{v}$ are given vectors in $\mathbb{R}^n$, and $\mathbf{u} \otimes \mathbf{v}$ is the *outer product* of $\mathbf{u}$ and $\mathbf{v}$, defined by

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_n \\ u_2v_1 & u_2v_2 & \cdots & u_2v_n \\ \vdots & \ddots & & \vdots \\ u_nv_1 & u_nv_2 & \cdots & u_nv_n \end{bmatrix}.$$

This modification of A to obtain B is called a *rank-one update*. This is because $\mathbf{u} \otimes \mathbf{v}$ has rank one, since every column of $\mathbf{u} \otimes \mathbf{v}$ is a scalar multiple of $\mathbf{u}$. To obtain B^{-1} from A^{-1}, we note that if

$$A\mathbf{x} = \mathbf{u},$$

then

$$B\mathbf{x} = (A + \mathbf{u} \otimes \mathbf{v})\mathbf{x} = A\mathbf{x} + \mathbf{u}\mathbf{v}^T\mathbf{x} = \mathbf{u} + \mathbf{u}(\mathbf{v} \cdot \mathbf{x}) = (1 + \mathbf{v} \cdot \mathbf{x})\mathbf{u},$$

which yields

$$B^{-1}\mathbf{u} = \frac{1}{1 + \mathbf{v} \cdot A^{-1}\mathbf{u}} A^{-1}\mathbf{u}.$$

On the other hand, if $\mathbf{x}$ is such that $\mathbf{v} \cdot A^{-1}\mathbf{x} = 0$, then

$$BA^{-1}\mathbf{x} = (A + \mathbf{u} \otimes \mathbf{v})A^{-1}\mathbf{x} = AA^{-1}\mathbf{x} + \mathbf{u}\mathbf{v}^T A^{-1}\mathbf{x} = \mathbf{x} + \mathbf{u}(\mathbf{v} \cdot A^{-1}\mathbf{x}) = \mathbf{x},$$

which yields

$$B^{-1}\mathbf{x} = A^{-1}\mathbf{x}.$$

This takes us to the following more general problem: given a matrix C, we wish to construct a matrix D such that the following conditions are satisfied:

- $D\mathbf{w} = \mathbf{z}$, for given vectors $\mathbf{w}$ and $\mathbf{z}$
- $D\mathbf{y} = C\mathbf{y}$, if $\mathbf{y}$ is orthogonal to a given vector $\mathbf{g}$.

In our application, $C = A^{-1}$, $D = B^{-1}$, $\mathbf{w} = \mathbf{u}$, $\mathbf{z} = 1/(1 + \mathbf{v} \cdot A^{-1}\mathbf{u})A^{-1}\mathbf{u}$, and $\mathbf{g} = A^{-T}\mathbf{v}$.

To solve this problem, we set

$$D = C + \frac{(\mathbf{z} - C\mathbf{w}) \otimes \mathbf{g}}{\mathbf{g} \cdot \mathbf{w}}.$$

Then, if $\mathbf{g} \cdot \mathbf{y} = 0$, the second term in the definition of D vanishes, and we obtain $D\mathbf{y} = C\mathbf{y}$, but in computing $D\mathbf{w}$, we obtain factors of $\mathbf{g} \cdot \mathbf{w}$ in the numerator and denominator that cancel, which yields

$$D\mathbf{w} = C\mathbf{w} + (\mathbf{z} - C\mathbf{w}) = \mathbf{z}.$$

Applying this definition of D, we obtain

$$B^{-1} = A^{-1} + \frac{\left[\left(\frac{1}{1+\mathbf{v}\cdot A^{-1}\mathbf{u}} A^{-1}\mathbf{u} - A^{-1}\mathbf{u}\right) \otimes \mathbf{v}\right] A^{-1}}{\mathbf{v} \cdot A^{-1}\mathbf{u}} = A^{-1} - \frac{A^{-1}(\mathbf{u} \otimes \mathbf{v})A^{-1}}{1 + \mathbf{v} \cdot A^{-1}\mathbf{u}}.$$

This formula for the inverse of a rank-one update is known as the *Sherman-Morrison Formula*.

We now return to the problem of approximating the Jacobian of $\mathbf{g}$, and efficiently obtaining its inverse, at each iterate $\mathbf{x}^{(k)}$. We begin with an exact Jacobian, $D_0 = J_{\mathbf{g}}(\mathbf{x}^{(0)})$, and use D_0 to compute the first iterate, $\mathbf{x}^{(1)}$, using Newton's Method as follows:

$$\mathbf{d}^{(0)} = -D_0^{-1}\mathbf{g}(\mathbf{x}^{(0)}), \quad \mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \mathbf{d}^{(0)}.$$

Then, we use the approximation

$$g'(x_1) \approx \frac{g(x_1) - g(x_0)}{x_1 - x_0}.$$

Generalizing this approach to a system of equations, we seek an approximation D_1 to $J_{\mathbf{g}}(\mathbf{x}^{(1)})$ that has these properties:

- $D_1(\mathbf{x}^{(1)} - \mathbf{x}^{(0)}) = \mathbf{g}(\mathbf{x}^{(1)}) - \mathbf{g}(\mathbf{x}^{(0)})$ (the *Secant Condition*)
- If $\mathbf{z} \cdot (\mathbf{x}^{(1)} - \mathbf{x}^{(0)}) = 0$, then $D_1\mathbf{z} = J_{\mathbf{g}}(\mathbf{x}^{(0)})\mathbf{z} = D_0\mathbf{z}$.

It follows from previous discussion that

$$D_1 = D_0 + \frac{\mathbf{y}_0 - D_0\mathbf{d}^{(0)}}{\mathbf{d}^{(0)} \cdot \mathbf{d}^{(0)}} \otimes \mathbf{d}^{(0)},$$

where

$$\mathbf{d}^{(0)} = \mathbf{x}^{(1)} - \mathbf{x}^{(0)}, \quad \mathbf{y}^{(0)} = g(\mathbf{x}^{(1)}) - \mathbf{g}(\mathbf{x}^{(0)}).$$

However, it can be shown (Chapter 3, Exercise 15) that $\mathbf{y}_0 - D_0\mathbf{d}^{(0)} = \mathbf{g}(\mathbf{x}^{(1)})$, which yields the simplified formula

$$D_1 = D_0 + \frac{1}{\mathbf{d}^{(0)} \cdot \mathbf{d}^{(0)}}\mathbf{g}(\mathbf{x}^{(1)}) \otimes \mathbf{d}^{(0)}.$$

Once we have computed D_0^{-1}, we can apply the Sherman-Morrison formula to obtain

$$\begin{aligned} D_1^{-1} &= D_0^{-1} - \frac{D_0^{-1}\left(\frac{1}{\mathbf{d}^{(0)}\cdot\mathbf{d}^{(0)}}\mathbf{g}(\mathbf{x}^{(1)}) \otimes \mathbf{d}^{(0)}\right) D_0^{-1}}{1 + \mathbf{d}^{(0)} \cdot D_0^{-1}\left(\frac{1}{\mathbf{d}^{(0)}\cdot\mathbf{d}^{(0)}}\mathbf{g}(\mathbf{x}^{(1)})\right)} \\ &= D_0^{-1} - \frac{D_0^{-1}\left(\mathbf{g}(\mathbf{x}^{(1)}) \otimes \mathbf{d}^{(0)}\right) D_0^{-1}}{\mathbf{d}^{(0)} \cdot \mathbf{d}^{(0)} + \mathbf{d}^{(0)} \cdot D_0^{-1}\mathbf{g}(\mathbf{x}^{(1)})} \\ &= D_0^{-1} - \frac{(\mathbf{u}^{(0)} \otimes \mathbf{d}^{(0)})D_0^{-1}}{\mathbf{d}^{(0)} \cdot (\mathbf{d}^{(0)} + \mathbf{u}^{(0)})}, \end{aligned}$$

where $\mathbf{u}^{(0)} = D_0^{-1}\mathbf{g}(\mathbf{x}^{(1)})$. Then, as D_1 is an approximation to $J_{\mathbf{g}}(\mathbf{x}^{(1)})$, we can obtain our next iterate $\mathbf{x}^{(2)}$ as follows:

$$D_1\mathbf{d}^{(1)} = -\mathbf{g}(\mathbf{x}^{(1)}), \quad \mathbf{x}^{(2)} = \mathbf{x}^{(1)} + \mathbf{d}^{(1)}.$$

Repeating this process, we obtain the following algorithm, which is known as *Broyden's Method*:

Choose $\mathbf{x}^{(0)}$
$D_0 = J_{\mathbf{g}}(\mathbf{x}^{(0)})$
$\mathbf{d}^{(0)} = -D_0^{-1}\mathbf{g}(\mathbf{x}^{(0)})$
$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \mathbf{d}^{(0)}$
$k = 0$
while not converged **do**
  $\mathbf{u}^{(k)} = D_k^{-1}\mathbf{g}(\mathbf{x}^{(k+1)})$
  $c_k = \mathbf{d}^{(k)} \cdot (\mathbf{d}^{(k)} + \mathbf{u}^{(k)})$
  $D_{k+1}^{-1} = D_k^{-1} - \frac{1}{c_k}[\mathbf{u}^{(k)} \otimes \mathbf{d}^{(k)}]D_k^{-1}$
  $k = k + 1$

$\mathbf{d}^{(k)} = -D_k^{-1}\mathbf{g}(\mathbf{x}^{(k)})$
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$
end

Note that it is not necessary to compute D_k for $k \geq 1$; only D_k^{-1} is needed. It follows that no systems of linear equations need to be solved during an iteration; only matrix-vector multiplications are required, thus saving an order of magnitude of computational effort during each iteration compared to Newton's Method.

Example We consider the system of equations $\mathbf{g}(\mathbf{x}) = \mathbf{0}$, where

$$\mathbf{g}(x, y, z) = \begin{bmatrix} x^2 + y^2 + z^2 - 3 \\ x^2 + y^2 - z - 1 \\ x + y + z - 3 \end{bmatrix}.$$

We will begin with one step of Newton's Method to solve this system of equations, with initial guess $\mathbf{x}^{(0)} = (x^{(0)}, y^{(0)}, z^{(0)}) = (1, 0, 1)$.

As computed in a previous example,

$$J_{\mathbf{g}}(x, y, z) = \begin{bmatrix} 2x & 2y & 2z \\ 2x & 2y & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Therefore, the Newton iterate $\mathbf{x}^{(1)}$ is obtained by solving the system of equations

$$J_{\mathbf{g}}(\mathbf{x}^{(0)})(\mathbf{x}^{(1)} - \mathbf{x}^{(0)}) = -\mathbf{g}(\mathbf{x}^{(0)}),$$

or, equivalently, $D_0\mathbf{d}^{(0)} = -\mathbf{g}(\mathbf{x}^{(0)})$ where

$$D_0 = J_{\mathbf{g}}(\mathbf{x}^{(0)}) = \begin{bmatrix} 2x^{(0)} & 2y^{(0)} & 2z^{(0)} \\ 2x^{(0)} & 2y^{(0)} & -1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{d}^{(0)} = \begin{bmatrix} x^{(1)} - x^{(0)} \\ y^{(1)} - y^{(0)} \\ z^{(1)} - z^{(0)} \end{bmatrix},$$

$$\mathbf{g}(\mathbf{x}^{(0)}) = \begin{bmatrix} (x^{(0)})^2 + (y^{(0)})^2 + (z^{(0)})^2 - 3 \\ (x^{(0)})^2 + (y^{(0)})^2 - z^{(0)} - 1 \\ x^{(0)} + y^{(0)} + z^{(0)} - 3 \end{bmatrix}.$$

Substituting $(x^{(0)}, y^{(0)}, z^{(0)}) = (1, 0, 1)$ yields the system

$$\begin{bmatrix} 2 & 0 & 2 \\ 2 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x^{(1)} - 1 \\ y^{(1)} \\ z^{(1)} - 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

This system has the solution $\mathbf{d}^{(0)} = (\frac{1}{2}, \frac{1}{2}, 0)$ which yields $\mathbf{x}^{(1)} = (\frac{3}{2}, \frac{1}{2}, 1)$.

Now, instead of computing $J_{\mathbf{g}}(\mathbf{x}^{(1)}$, we compute

$$\begin{aligned} D_1 &= D_0 + \frac{1}{\mathbf{d}^{(0)} \cdot \mathbf{d}^{(0)}} \mathbf{g}(\mathbf{x}^{(1)}) \otimes \mathbf{d}^{(0)} \\ &= \begin{bmatrix} 2 & 0 & 2 \\ 2 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} + \frac{1}{1/2} \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1/2 \\ 1/2 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 0 & 2 \\ 2 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix} + 2 \begin{bmatrix} 1/4 & 1/4 & 0 \\ 1/4 & 1/4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 5/2 & 1/2 & 2 \\ 5/2 & 1/2 & -1 \\ 1 & 1 & 1 \end{bmatrix}. \end{aligned}$$

This yields the system

$$\begin{bmatrix} 5/2 & 1/2 & 2 \\ 5/2 & 1/2 & -1 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x^{(2)} - \frac{3}{2} \\ y^{(2)} - \frac{1}{2} \\ z^{(2)} - 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ 0 \end{bmatrix},$$

which has the solution $\mathbf{x}^{(2)} = (\frac{5}{4}, \frac{3}{4}, 1)$. This happens to be the same second iterate computed by Newton's Method, though this is not the case for a general function $\mathbf{g}(\mathbf{x})$. □

In the next lecture, we will learn how to apply Broyden's Method to solve minimization problems.

Exercises

1. Chapter 3, Exercise 14
2. Chapter 3, Exercise 15