PHYS 410/555 Computational Physics

Solution of Nonlinear Systems Using Newton's Method: Summary Notes

We begin by recalling Newton's method for the solution of a single nonlinear equation

$$f(x) = 0 \tag{1}$$

in a single variable, x.

Starting from some initial guess, $x^{(0)}$, Newton's method generates iterates, $x^{(n)}$ via

$$x^{(n+1)} = x^{(n)} - \delta x^{(n)} \tag{2}$$

where $\delta x^{(n)}$ satisfies

$$f'(x^{(n)})\,\delta x^{(n)} = r^{(n)} \equiv f(x^{(n)})\,. \tag{3}$$

Here, $f'(x) \equiv df/dx$, and $r^{(n)}$ is defined as the *residual* associated with (1), which, assuming that the iteration converges, is driven to 0 as $n \to \infty$.

Equations (2) and (3) can be combined in the more compact form:

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}\,. \tag{4}$$

When Newton's method converges, it tends to do so *rapidly*; we can expect the number of significant digits (accurate digits) in $x^{(n)}$ to roughly *double* at each iteration (quadratic convergence).

We now proceed to the case of nonlinear *systems*. We now wish to solve

$$\mathbf{f}(\mathbf{x}) = \mathbf{0} \tag{5}$$

where

$$\mathbf{x} = (x_1, x_2, \ldots, x_d) \tag{6}$$

$$\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_d(\mathbf{x})) \tag{7}$$

$$f_i(\mathbf{x}) = f_i(x_1, x_2, \ldots, x_d) \tag{8}$$

Expression (5) is a non-linear system of d equations, $f_i(x_1, x_2, \ldots, x_d) = 0$, in d unknowns, $x_1, x_2, \ldots, x_d$.

For illustrative purposes, we consider the following specific system, where $d = 2$:

$$\sin(xy) = \frac{1}{2} \tag{9}$$

$$y^2 = 6x + 2 \tag{10}$$

As in the scalar case ($d = 1$), Newton's method for systems is again iterative; we start from some initial guess, $\mathbf{x}^{(0)}$, then generate iterates:

$$\mathbf{x}^{(0)} \to \mathbf{x}^{(1)} \cdots \mathbf{x}^{(n)} \to \mathbf{x}^{(n+1)} \cdots \to \mathbf{x}^{\star} \tag{11}$$

where $\mathbf{x}^{\star}$ is a specific solution of (5). Again, as with the scalar case, with any appoximate solution, $\mathbf{x}^{(n)}$, we associate the residual

$$\mathbf{r}^{(n)} \equiv \mathbf{f}(\mathbf{x}^{(n)}) \tag{12}$$

In the d-dimensional case, the analogue of $f'(x)$ is the *Jacobian matrix*, $\mathbf{J}$, of the first derivatives of $\mathbf{f}$. $\mathbf{J}$ has elements J_{ij} given by

$$J_{ij} \equiv \frac{\partial f_i}{\partial x_j} \tag{13}$$

For the example defined above, we have $\mathbf{x} = (x_1, x_2) \equiv (x, y)$, and

$$\begin{bmatrix} \partial f_1/\partial x & \partial f_1/\partial y \\ \partial f_2/\partial x & \partial f_2/\partial y \end{bmatrix} = \begin{bmatrix} y\cos(xy) & x\cos(xy \\ -6 & 2y \end{bmatrix} \tag{14}$$

We can derive the d-dimensional Newton iteration by considering a multi-dimensional Taylor series expansion. Specifically, we expand $\mathbf{f}(\mathbf{x}^\star)$ about the n-th interation, $\mathbf{x}^{(n)}$, as follows:

$$\mathbf{0} = \mathbf{f}(\mathbf{x}^\star) = \mathbf{f}(\mathbf{x}^{(n)}) + \mathbf{J}(\mathbf{x}^{(n)}) \cdot (\mathbf{x}^\star - \mathbf{x}^{(n)}) + O((\mathbf{x}^\star - \mathbf{x}^{(n)})^2) \tag{15}$$

We then drop the higher order terms, and replace the solution $\mathbf{x}^\star$ with the new iterate $\mathbf{x}^{(n+1)}$, yielding:

$$\mathbf{0} = \mathbf{f}(\mathbf{x}^{(n)}) + \mathbf{J}(\mathbf{x}^{(n)}) \cdot (\mathbf{x}^{(n+1)} - \mathbf{x}^{(n)}) \tag{16}$$

Defining $\delta\mathbf{x}^{(n)}$ via

$$\delta\mathbf{x}^{(n)} \equiv \mathbf{x}^{(n)} - \mathbf{x}^{(n+1)} \tag{17}$$

and rearranging (16), we have Newton's method for systems:

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} - \delta\mathbf{x}^{(n)} \tag{18}$$

where $\delta\mathbf{x}^{(n)}$ satisfies the linear system:

$$\mathbf{J}(\mathbf{x}^{(n)}) \cdot \delta\mathbf{x}^{(n)} = \mathbf{f}(\mathbf{x}^{(n)}) \equiv \mathbf{r}^{(n)} \tag{19}$$

Observe that (19) is a $d \times d$ *linear system* for the unknowns $\delta\mathbf{x}^{(n)}$. Also recall that the Jacobian matrix, $\mathbf{J}(\mathbf{x}^{(n)})$ has elements:

$$\mathbf{J}_{ij}(\mathbf{x}^{(n)}) = \left.\frac{\partial f_i}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}^{(n)}} \tag{20}$$

(19) can be solved using an appropriate linear solver (`dgesv`, `dgtsv`, `dgbsv`, ...).

Finally, the following pseudo-code describes the general structure of a typical implementation of a multi-dimensional Newton solver:

```
x = x^(0)
do while ||dx|| > ε
   do i = 1, d
      res(i) = f_i(x)
      do j = 1, d
         J(i,j) = ∂f_i/∂x_j(x)
      end do
   end do
   dx = solve(J · dx = res)
   x = x - dx
end do
```