Notes for 2016-04-06

Fixed points and contraction mappings

As discussed last time, many iterations we consider have the form

$$x^{k+1} = G(x^k)$$

where $G : \mathbb{R}^n \to \mathbb{R}^n$. We call G a *contraction* on Ω if it is Lipschitz with constant less than one, i.e.

$$\|G(x) - G(y)\| \leq \alpha \|x - y\|, \quad \alpha < 1.$$

According to the *contraction mapping theorem* or *Banach fixed point theorem*, when G is a contraction on Ω and $G(\Omega) = \Omega$, there is a unique fixed point $x^* \in \Omega$ (i.e. a point such that $x^* = G(x^*)$).

If we can express the solution of a nonlinear equation as the fixed point of a contraction mapping, we get two immediate benefits:

- We know that a solution exists and is unique (at least, it is unique within some Ω). This is a nontrivial advantage, as it is easy to write nonlinear equations that have no solutions, or have continuous families of solutions, without realizing that there is a problem.

- We have a numerical method — albeit a potentially slow one — for computing the fixed point. We take the fixed point iteration

  $$x^{k+1} = G(x^k)$$

  started from some $x^0 \in \Omega$, and we subtract the fixed point equation $x^* = G(x^*)$ to get an iteration for $e^k = x^k - x^*$:

  $$e^{k+1} = G(x^* + e^k) - G(x^*).$$

  Using contractivity, we get

  $$\|e^{k+1}\| = \|G(x^* + e^k) - G(x^*)\| \leq \alpha \|e^k\|,$$

  which implies that $\|e^k\| \leq \alpha^k \|e^0\| \to 0$.

When error goes down by a factor of α at each step, we say the iteration is *linearly convergent* (or *geometrically convergent*). The name reflects a semilogarithmic plot of (log) error versus iteration count; if the errors lie on a straight line, we have linear convergence. Contractive fixed point iterations converge at least linearly, but may converge more quickly.

Newton's method for nonlinear equations

The idea behind Newton's method[1] is to approximate a nonlinear $f \in C^1$ by linearizations around successive guesses. We then get the next guess by finding where the linearized approximation is zero. That is, we set

$$f(x^{k+1}) \approx f(x^k) + f'(x^k)(x^{k+1} - x^k) = 0,$$

which we can rearrange to

$$x^{k+1} = x^k - f'(x^k)^{-1} f(x^k).$$

To emphasize that we do not want to actually form an inverse, and to set the stage for later variations on the method, we also write the iteration as

$$\begin{aligned} f'(x^k)p^k &= -f(x^k) \\ x^{k+1} &= x^k + p^k. \end{aligned}$$

Superlinear convergence

Suppose $f(x^*) = 0$. Taylor expansion about x^k gives

$$0 = f(x^*) = f(x^k) + f'(x^k)(x^* - x^k) + r(x^k)$$

where the remainder term $r(x^k)$ is $o(\|x^k - x^*\|) = o(\|e^k\|)$. Hence,

$$x^{k+1} = x^* + f'(x^k)^{-1} r(x^k)$$

and subtracting from x^* from both sides gives

$$e^{k+1} = f'(x^k)^{-1} r(x^k) = f'(x^k)^{-1} o(\|e^k\|)$$

If $\|f'(x)^{-1}\|$ is bounded for x near x^* and x^0 is close enough, this is sufficient to guarantee *superlinear convergence*. When we have a stronger condition, such as f' Lipschitz, we get *quadratic convergence*, i.e. $e^{k+1} = O(\|e^k\|^2)$. Of course, this is all local theory – we need a good initial guess!

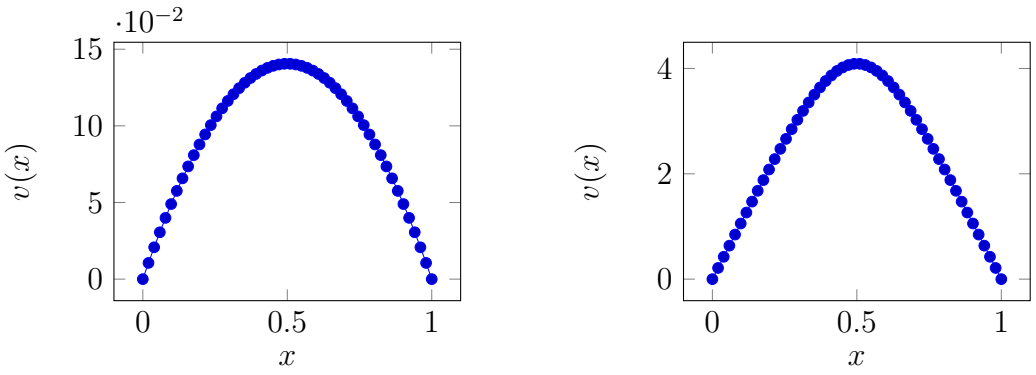

Figure 1: Two solutions for the blow-up example.

A worked example

We now consider an example problem, a nonlinear system that comes from a discretized PDE reaction-diffusion model describing (for example) the steady state heat distribution in a medium going an auto-catalytic reaction. The physics is that heat is generated in the medium due to a reaction, with more heat where the temperature is higher. The heat then diffuses through the medium, and the outside edges of the domain are kept at the ambient temperature. The PDE and boundary conditions are

$$\begin{aligned} v_{,xx} + \exp(v) = 0, \quad & x \in (0,1) \\ v(0) = v(1) = 0. & \end{aligned}$$

We discretize the PDE for computer solution by introducing a mesh $x_i = ih$ for $i = 0, \ldots, N+2$ and $h = 1/(N+2)$; the solution is approximated by $v(x_i) \approx v_i$. We set $v_0 = v_{N+2} = 0$; when we refer to v without subscripts, we mean the vector of entries v_1 through v_N. This discretization leads to the nonlinear system

$$f_i(v) \equiv \frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + \exp(v_i) = 0.$$

for $i = 1, \ldots, N$. This equation has two solutions, shown in Figure 1 for $N = 50$; physically, these correspond to stable and unstable steady-state solutions of the time-dependent version of the model.

[1] You will see the method referred to as Newton's method or Newton-Raphson. A recent short article gives an interesting history:
`http://www.math.uiuc.edu/documenta/vol-ismp/13_deuflhard-peter.pdf`

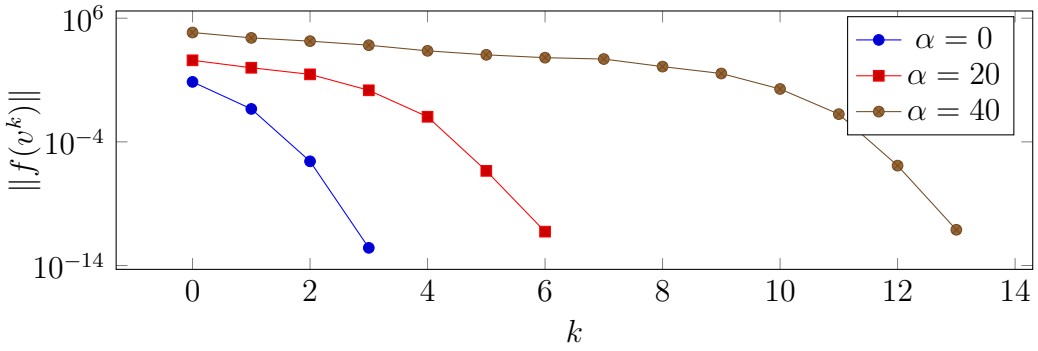

Figure 2: Newton convergence for different starting guesses.

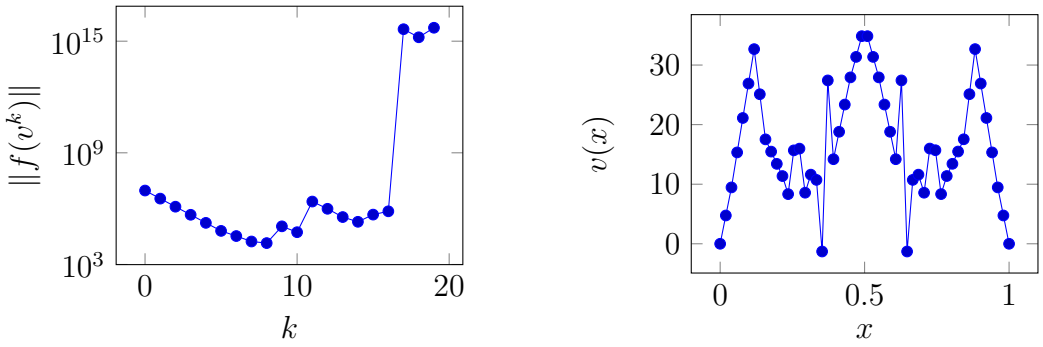

Figure 3: Residual history and final "solution" for $\alpha = 60$.

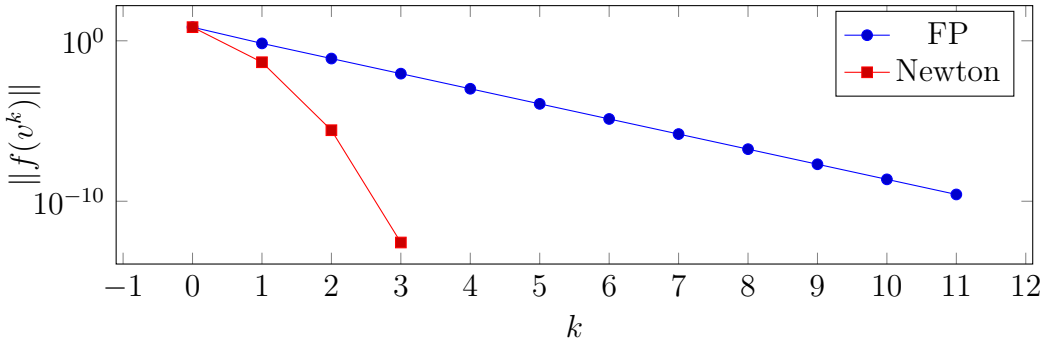

Figure 4: Convergence for fixed point iteration vs Newton for $\alpha = 0$.

To solve for a Newton step, we need the Jacobian of f with respect to the variables v_j is the tridiagonal matrix. We write this as

$$J(v) = -h^{-2}T_N + \text{diag}(\exp(v))$$

where

$$T_N = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{N \times N}.$$

For an initial guess, we use $v_i = \alpha x_i(1 - x_i)$ for different values of α. For $\alpha = 0$, we converge to the stable solution; for $\alpha = 20$ and $\alpha = 40$, we converge to the unstable solution. We eventually see quadratic convergence in all cases, but for $\alpha = 40$ there is a longer period before convergence sets in (Figure 2). For $\alpha = 60$, the method does not converge at all (Figure 3).

We can derive a Newton-like fixed point iteration from the observation that if v remains modest, the Jacobian is pretty close to $-h^2T_N$. This gives us the iteration

$$h^{-2}T_N v^{k+1} = \exp(v^k).$$

In Figure 4, we compare the convergence of this fixed point iteration to Newton's method. The fixed point iteration does converge, but it shows the usual linear convergence, while Newton's method converges quadratically.

Beyond asymptotic convergence

We saw in our example that Newton's method may indeed fail if the initial guess is not sufficiently good. Let's now be a little more precise about our error analysis in order to see where Newton gets into trouble.

Let M be a Lipschitz constant on f' in some ball around x^*, i.e. $\|f'(x) - f'(y)\| \leq M\|x-y\|$ for any points in the ball. Inside the ball, we have

$$\|r(x)\| \leq \frac{M}{2}\|x-x^*\|^2$$

and

$$\|f'(x) - f'(x^*)\| \leq M.$$

Recall from earlier in the semester that a Neumann bound says if A is non-singular and $\|A^{-1}\|\|E\| < 1$, then $\|(A+E)^{-1}\| \leq \|A^{-1}\|/(1-\|A^{-1}\|\|E\|)$. Applying this to $f'(x)$ and $f'(x^*)$, we have that if

$$M\|f'(x^*)^{-1}\|\|x-x^*\| < 1$$

then

$$\|f'(x)^{-1}\| \leq \frac{\|f'(x^*)\|}{1 - M\|f'(x^*)^{-1}\|\|x-x^*\|}.$$

Plugging this into our error iteration, we have that if

$$\gamma_k = M\|f'(x^*)^{-1}\|\|e^k\| < 1$$

then

$$\|e^{k+1}\| \leq \frac{\gamma_k}{2(1-\gamma_k)}\|e^k\|$$

Putting everything together, we have that if $\gamma_k < 2/3$, then $\|e^{k+1}\| < \|e^k\|$ (and $\gamma_{k+1} < \gamma_k$). Therefore, we can guarantee convergence of Newton if the initial error satisfies

$$\|e^0\| < \frac{2}{3M\|f'(x^*)^{-1}\|}.$$

The radius of guaranteed convergence will be small if $f'(x^*)$ is very close to singular or if $f'(x)$ can change very quickly (M large). Of course, this is only a bound! In practice, the quickest way to see if Newton converges for a particular problem is to try and see what happens.

The bound we gave is somewhat unsatisfactory in that it involves the norm of $f'(x^*)^{-1}$, and we do not know $f'(x^*)$ a priori. A slightly more subtle

argument due to Kantorovich shows that Newton converges to a zero of f close to the initial guess under some conditions on $f'(x^0)$ and $f(x^0)$, along with a Lipschitz condition on $f'(x)$.

Newton's method for optimization

We now turn to using Newton's method to solve optimization problems. We can approach this in two ways: either by applying Newton's method to solve the critical point equations or by developing the method directly from the optimization problem. We will follow the latter approach.

Suppose $f : \mathbb{R}^n \to \mathbb{R}$ is C^2, and x^* is a (strong) local minimizer of f. We want a sequence of points x^k that converge to x^*, assuming the initial guess x^0 is good enough. As with solving nonlinear equations, our approach will be to approximate f locally by a Taylor expansion:

$$f(x^k + z) \approx f(x^k) + f'(x^k)z + \frac{1}{2} z^T H(x^k) z$$

Assuming $H(x^k)$ is positive definite, the quadratic approximation has a minimum at $z = -H(x^k)^{-1}\nabla f(x^k)$. This gives us the Newton update

$$x^{k+1} = x^k - H(x^k)^{-1}\nabla f(x^k).$$

This is the same update we would get by applying Newton to the critical point equations.

Newton's method for optimization is special in that there is more structure in the critical point equations than there is for a nonlinear system of equations. For example, if x^k is sufficiently close to a strong local minimum, then $H(x^k)$ is guaranteed to be positive definite. Not only does this mean we can use Cholesky (rather than LU) to solve the Newton update system, but the update

$$p = -H(x^k)^{-1}\nabla f(x^k)$$

has the property that

$$\nabla f(x^k)^T p = -p^T H(x^k) p < 0,$$

i.e. moving in the direction of p decreases the objective function. That is, if $H(x^k)$ is positive definite, then p is a *descent direction*. Of course, just

because p points in a downhill direction does not necessarily mean that the Newton step will reduce the objective function value – if we take a full Newton step, we may move so far that the Taylor expansion ceases to give good information

Some practical issues

In general, there is no guarantee that a given solution of nonlinear equations will have a solution; and if there is a solution, there is no guarantee of uniqueness. This has a practical implication: many incautious computationalists have been embarrassed to find that they have "solved" a problem that was later proved to have no solution!

When we have reason to believe that a given system of equations has a solution — whether through mathematical argument or physical intuition — we still have the issue of finding a good enough initial estimate that Newton's method will converge. In coming lectures, we will discuss "globalization" methods that expand the set of initial guesses for which Newton's method converges; but globalization does not free us from the responsibility of trying for a good guess. Finding a good guess helps ensure that our methods will converge quickly, and to the "correct" solution (particularly when there are multiple possible solutions).

We saw one explicit example of the role of the initial guess in our analysis of the discretized blowup PDE problem. Another example occurs when we use unguarded Newton iteration for optimization. Given a poor choice of initial guess, we are as likely to converge to a saddle point or a local maximum as to a minimum! But we will address this pathology in our discussion of globalization methods.

If we have a nice problem and an adequate initial guess, Newton's iteration can converge quite quickly. But even then, we still have to think about when we will be satisfied with an approximate solution. A robust solver should check a few possible termination criteria:

- **Iteration count**: It makes sense to terminate (possibly with a diagnostic message) whenever an iteration starts to take more steps than one expects — or perhaps more steps than one can afford. If nothing else, this is necessary to deal with issues like poor initial guesses.

- **Residual check**: We often declare completion when $\|f(x^k)\|$ is suffi-

ciently close to zero. What is "close to zero" depends on the scaling of the problem, so users of black box solvers are well advised to check that any default residual checks make sense for their problems.

- **Update check**: Once Newton starts converging, a good estimate for the error at step x^k is $x^{k+1} - x^k$. A natural test is then to make sure that $\|x^{k+1} - x^k\|/\|x^{k+1}\| < \tau$ for some tolerance τ. Of course, this is really an estimate of the relative error at step k, but we will report the (presumably better) answer x^{k+1} — like anyone else who can manage it, numerical analysts like to have their cake and eat it, too.

A common problem with many solvers is to make the termination criteria too lax, so that a bad solution is accepted; or too conservative, so that good solutions are never accepted.

One common mistake in coding Newton's method is to goof in computing the Jacobian matrix. This error is not only very common, but also very often overlooked. After all, a good approximation to the Jacobian often still produces a convergent iteration; and when iterations diverge, it is hard to distinguish between problems due to a bad Jacobian and problems due to a bad initial guess. However, there is a simple clue to watch for that can help alert the user to a bad Jacobian calculation. In most cases, Newton converges quadratically, while "almost Newton" iterations converge quadratically. If you think you have coded Newton's method and a plot of the residuals shows linear behavior, look for issues with your Jacobian code!

Addendum: blow-up solver codes

Newton code

```
function [v,x,rhist] = blowup(alpha)

  % -- Number of (interior) mesh points and mesh spacing
  N = 50;                  % Number of (interior) mesh points
  h = 1/(N+1);             % Mesh spacing
  x = linspace(0,1,N+2)'; % Mesh points

  % -- Initial guess
  v = alpha .* x .* (1-x);

  % -- Set up some things for solver
  e = ones(N,1);           % Vector of ones
  rhist = [];              % Residual history

  % Newton loop (fixed number of steps)
  for k = 1:20

    % Compute residual vector and record norm
    r = ( v(1:N)-2*v(2:N+1)+v(3:N+2) )/h^2 + exp(v(2:N+1));
    rhist(k) = norm(r);

    % Form (sparse) Jacobian
    J = spdiags([e/h^2, exp(v(2:N+1))-2/h^2, e/h^2], -1:1, N,N);

    % Compute Newton update
    p = J\r;
    v(2:N+1) = v(2:N+1)-p;

    % Quit if we are making small changes
    if norm(p) / norm(v) < 1e-10, break; end
  end
```

Fixed point code

```
function [v,x,rhist] = blowupfp(alpha)

  % -- Number of (interior) mesh points and mesh spacing
  N = 50;                  % Number of (interior) mesh points
  h = 1/(N+1);             % Mesh spacing
  x = linspace(0,1,N+2)'; % Mesh points

  % -- Initial guess
  v = alpha .* x .* (1-x);

  % -- Set up some things for solver
  e = ones(N,1);           % Vector of ones
  rhist = [];              % Residual history

  L = spdiags([-e/h^2, 2*e/h^2, -e/h^2], -1:1, N,N);

  % Newton loop (fixed number of steps)
  for k = 1:12

    % Compute residual vector and record norm
    r = ( v(1:N)-2*v(2:N+1)+v(3:N+2) )/h^2 + exp(v(2:N+1));
    rhist(k) = norm(r);

    % Compute fixed point update
    v(2:N+1) = L\exp(v(2:N+1));
  end
```