Chapter 7

Nonlinear Systems

Try as we might, it simply is not possible to express all systems of equations in the linear framework we have developed over the last several chapters. It is hardly necessary to motivate the usage of logarithms, exponentials, trigonometric functions, absolute values, polynomials, and so on in practical problems, but except in a few special cases none of these functions is linear. When these functions appear, we must employ a more general if less efficient set of machinery.

7.1 Single-Variable Problems

We begin our discussion by considering problems of a single scalar variable. In particular, given a function $f(x) : \mathbb{R} \to \mathbb{R}$, we wish to develop strategies for finding points $x^* \in \mathbb{R}$ such that $f(x^*) = 0$; we call x^* a *root* of f. Single-variable problems in linear algebra are not particularly interesting; after all we can solve the equation $ax - b = 0$ in closed form as $x^* = {}^b/_a$. Solving a nonlinear equation like $y^2 + e^{\cos y} - 3 = 0$, however, is far less obvious (incidentally, the solution is $y^* = \pm 1.30246\ldots$).

7.1.1 Characterizing Problems

We no longer can assume f is linear, but without any assumption on its structure we are unlikely to make headway on solving single-variable systems. For instance, a solver is guaranteed to fail finding zeros of $f(x)$ given by

$$f(x) = \begin{cases} -1 & x \leq 0 \\ 1 & x > 0 \end{cases}$$

Or worse:

$$f(x) = \begin{cases} -1 & x \in \mathbb{Q} \\ 1 & \text{otherwise} \end{cases}$$

These examples are trivial in the sense that a rational client of root-finding software would be unlikely to expect it to succeed in this case, but far less obvious cases are not much more difficult to construct.

For this reason, we must add some "regularizing" assumptions about f providing a toehold into the possibility of designing root-finding techniques. Typical such assumptions are below, listed in increasing order of strength:

- *Continuity:* A function f is *continuous* if it can be drawn without lifting up a pen; more formally, f is continuous if the difference $f(x) - f(y)$ vanishes as $x \to y$.
- *Lipschitz:* A function f is *Lipschitz continuous* if there exists a constant C such that $|f(x) - f(y)| \leq C|x - y|$; Lipschitz functions need not be differentiable but are limited in their rates of change.
- *Differentiability:* A function f is *differentiable* if its derivative f' exists for all x.
- *C^k:* A function is C^k if it is differentiable k times and each of those k derivatives is continuous; C^∞ indicates that all derivatives of f exist and are continuous.

As we add stronger and stronger assumptions about f, we can design more effective algorithms to solve $f(x^*) = 0$. We will illustrate this effect by considering a few algorithms below.

7.1.2 Continuity and Bisection

Suppose all we know about f is that it is continuous. In this case, we can state an intuitive theorem from standard single variable calculus:

Theorem 7.1 (Intermediate Value Theorem). *Suppose $f : [a, b] \to \mathbb{R}$ is continuous. Suppose $f(x) < u < f(y)$. Then, there exists z between x and y such that $f(z) = u$.*

In other words, the function f must achieve every value in between $f(x)$ and $f(y)$.

Suppose we are given as input the function f as well as two values ℓ and r such that $f(\ell) \cdot f(r) < 0$; notice this means that $f(\ell)$ and $f(r)$ have opposite sign. Then, by the Intermediate Value Theorem we know that somewhere between ℓ and r there is a root of f! This provides an obvious bisection strategy for finding x^*:

1. Compute $c = {}^{\ell+r}/_{2}$.
2. If $f(c) = 0$, return $x^* = c$.
3. If $f(\ell) \cdot f(c) < 0$, take $r \leftarrow c$. Otherwise take $\ell \leftarrow c$.
4. If $|r - \ell| < \varepsilon$, return $x^* \approx c$.
5. Go back to step 1

This strategy simply divides the interval $[\ell, r]$ in half iteratively, each time keeping the side in which a root is known to exist. Clearly by the Intermediate Value Theorem it converges *unconditionally*, in the sense that so long as $f(\ell) \cdot f(r) < 0$ eventually both ℓ and r are guaranteed converge to a valid root x^*.

7.1.3 Analysis of Root-Finding

Bisection is the simplest but not necessarily the most effective technique for root-finding. As with most eigenvalue methods, bisection inherently is iterative and may never provide an *exact* solution x^*. We can ask, however, how close the value c_k of c in the k-th iteration is to the root x^* that we hope to compute. This analysis will provide a baseline for comparison to other methods.

In general, suppose we can establish an error bound E_k such that the estimate x_k of the root x^* during the k-th iteration of a root-finding method satisfies $|x_k - x^*| < E_k$. Obviously any algorithm with $E_k \to 0$ represents a *convergent* scheme; the speed of convergence, however, can be characterized by the rate at which E_k approaches 0.

For example, in bisection since both c_k and x^* are in the interval $[\ell_k, r_k]$, an upper bound of error is given by $E_k \equiv |r_k - \ell_k|$. Since we divide the interval in half each iteration, we know $E_{k+1} = 1/2 E_k$. Since E_{k+1} is linear in E_k, we say that bisection exhibits *linear* convergence.

7.1.4 Fixed Point Iteration

Bisection is guaranteed to converge to a root for any continuous function f, but if we know more about f we can formulate algorithms that can converge more quickly.

As an example, suppose we wish to find x^* satisfying $g(x^*) = x^*$; of course, this setup is equivalent to the root-finding problem since solving $f(x) = 0$ is the same as solving $f(x) + x = x$. As additional piece of information, however, we also might know that g is Lipschitz with constant $C < 1$.

The system $g(x) = x$ suggests a potential strategy we might hypothesize:

1. Take x_0 to be an initial guess of a root.

2. Iterate $x_k = g(x_{k-1})$.

If this strategy converges, clearly the result is a *fixed point* of g satisfying the criteria above.

Thankfully, the Lipschitz property ensures that this strategy converges to a root if one exists. If we take $E_k = |x_k - x^*|$, then we have the following property:

$$\begin{aligned} E_k &= |x_k - x^*| \\ &= |g(x_{k-1}) - g(x^*)| \text{ by design of the iterative scheme and definition of } x^* \\ &\leq C|x_{k-1} - x^*| \text{ since } g \text{ is Lipschitz} \\ &= CE_{k-1} \end{aligned}$$

Applying this statement inductively shows $E_k \leq C^k |E_0| \to 0$ as $k \to \infty$. Thus, fixed point iteration converges to the desired x^*!

In fact, if g is Lipschitz with constant $C < 1$ in a *neighborhood* $[x^* - \delta, x^* + \delta]$, then so long as x_0 is chosen in this interval fixed point iteration will converge. This is true since our expression for E_k above shows that it shrinks each iteration.

One important case occurs when g is C^1 and $|g'(x^*)| < 1$. By continuity of g' in this case, we know that there is some neighborhood $N = [x^* - \delta, x^* + \delta]$ in which $|g'(x)| < 1 - \varepsilon$ for any $x \in N$, for some choice of a sufficiently small $\varepsilon > 0$.[1] Take any $x, y \in N$. Then, we have

$$\begin{aligned} |g(x) - g(y)| &= |g'(\theta)| \cdot |x - y| \text{ by the Mean Value Theorem of basic calculus, for some } \theta \in [x, y] \\ &< (1 - \varepsilon)|x - y| \end{aligned}$$

This shows that g is Lipschitz with constant $1 - \varepsilon < 1$ in N. Thus, when g is continuously differentiable and $g'(x^*) < 1$, fixed point iteration will converge to x^* when the initial guess x_0 is close by.

[1]This statement is hard to parse: Make sure you understand it!

So far we have little reason to use fixed point iteration: We have shown it is guaranteed to converge only when g is Lipschitz, and our argument about the E_k's shows linear convergence like bisection. There is one case, however, in which fixed point iteration provides an advantage.

Suppose g is differentiable with $g'(x^*) = 0$. Then, the first-order term vanishes in the Taylor series for g, leaving behind:

$$g(x_k) = g(x^*) + \frac{1}{2}g''(x^*)(x_k - x^*)^2 + O\left((x_k - x^*)^3\right).$$

Thus, in this case we have:

$$\begin{aligned}
E_k &= |x_k - x^*| \\
&= |g(x_{k-1}) - g(x^*)| \text{ as before} \\
&= \frac{1}{2}|g''(x^*)|(x_{k-1} - x^*)^2 + O((x_{k-1} - x^*)^3) \text{ from the Taylor argument} \\
&\leq \frac{1}{2}(|g''(x^*)| + \varepsilon)|(x_{k-1} - x^*)^2 \text{ for some } \varepsilon \text{ so long as } x_{k-1} \text{ is close to } x^* \\
&= \frac{1}{2}(|g''(x^*)| + \varepsilon)E_{k-1}^2
\end{aligned}$$

Thus, in this case E_k is *quadratic* in E_{k-1}, so we say fixed point iteration can have *quadratic convergence*; notice this proof of quadratic convergence only holds because we already know $E_k \to 0$ from our more general convergence proof. This implies that $E_k \to 0$ much faster, so we will need fewer iterations to reach a reasonable root.

Example 7.1 (Convergence of fixed-point iteration)**.**

7.1.5 Newton's Method

We tighten our class of functions once more to derive a method that has more consistent quadratic convergence. Now, suppose again we wish to solve $f(x^*) = 0$, but now we assume that f is C^1, a slightly tighter condition than Lipschitz.

At a point $x_k \in \mathbb{R}$, since f now is differentiable we can approximate it using a tangent line:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

Solving this approximation for $f(x) \approx 0$ yields a root

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Iterating this formula is known as *Newton's method* for root-finding, and it amounts to iteratively solving a linear approximation of the nonlinear problem.

Notice that if we define

$$g(x) = x - \frac{f(x)}{f'(x)},$$

then Newton's method amounts to fixed point iteration on g. Differentiating, we find:

$$\begin{aligned}
g'(x) &= 1 - \frac{f'(x)^2 - f(x)f''(x)}{f'(x)^2} \text{ by the quotient rule} \\
&= \frac{f(x)f''(x)}{f'(x)^2}
\end{aligned}$$

Suppose x^* is a *simple* root, meaning $f'(x^*) \neq 0$. Then, $g'(x^*) = 0$, and by our derivation of fixed-point iteration above we know that Newton's method converges quadratically to x^* for a sufficiently close initial guess. Thus, when f is differentiable with a simple root Newton's method provides a fixed-point iteration formula that is guaranteed to converge quadratically; when x^* is not simple, however, convergence can be linear or worse.

The derivation of Newton's method suggests other methods derived by using more terms in the Taylor series. For instance, "Halley's method" adds terms involving f'' to the iterations, and a class of "Householder methods" takes an arbitrary number of derivatives. These techniques offer even higher-order convergence at the cost of having to evaluate more complex iterations and the possibility of more exotic failure modes. Other methods replace Taylor series with other basic forms; for example, linear fractional interpolation uses rational functions to better approximate functions with asymptote structure.

7.1.6 Secant Method

One efficiency concern we have not addressed yet is the cost of evaluating f and its derivatives. If f is a very complicated function, we may wish to minimize the number of times we have to compute f or worse f'. Higher orders of convergence help with this problem, but we also can design numerical methods that avoid evaluating costly derivatives.

Example 7.2 (Design). *Suppose we are designing a rocket and wish to know how much fuel to add to the engine. For a given number of gallons x, we can write a function $f(x)$ giving the maximum height of the rocket; our engineers have specified that we wish to the rocket to reach a height h, so we need to solve $f(x) = h$. Evaluating $f(x)$ involves simulating a rocket as it takes off and monitoring its fuel consumption, which is an expensive proposition, and although we might suspect that f is differentiable we might not be able to evaluate f' in a practical amount of time.*

One strategy for designing lower-impact methods is to reuse data as much as possible. For instance, we easily could approximate:

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}.$$

That is, since we had to compute $f(x_{k-1})$ in the previous iteration, we simply use the slope to $f(x_k)$ to approximate the derivative. Certainly this approximation works well, especially when x_k's are near convergence.

Plugging our approximation into Newton's method reveals a new iterative scheme:

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}$$

Notice that the user will have to provide two initial guesses x_0 and x_{-1} to start this scheme, or can run a single iteration of Newton to get it started.

Analyzing the secant method is somewhat more complicated than the other methods we consider because it uses both $f(x_k)$ and $f(x_{k-1})$; proof of its convergence is outside the scope of our discussion. Interestingly, error analysis reveals that error decreases at a rate of $1+\sqrt{5}/2$ (the "Golden Ratio"), between linear and quadratic; since convergence is *close* to that of Newton's method without the need for evaluating f', the secant method can provide a strong alternative.

7.1.7 Hybrid Techniques

Additional engineering can be carried out to attempt to combine the advantages of different root-finding algorithms. For instance, we might make the following observations about two methods we have discussed:

- *Bisection* is guaranteed to converge unconditionally but only does so at a linear rate.
- The *secant method* converges faster when it does reach a root, but in some cases it may not converge.

Suppose we have bracketed a root of $f(x)$ in an interval $[\ell_k, r_k]$ as in bisection. We can say that our current estimate of x^* is given by $x_k = \ell_k$ when $|f(\ell_k)| < |f(r_k)|$ and $x_k = r_k$ otherwise. If we keep track of x_k and x_{k-1}, then we could take x_{k+1} to be the next estimate of the root given by the secant method. If x_k is *outside* the interval $[\ell_k, r_k]$, however, we can replace it with $\ell_k+r_k/2$. This correction guarantees that $x_{k+1} \in [\ell_k, r_k]$, and regardless of the choice we can update to a valid bracket $[\ell_{k+1}, r_{k+1}]$ as in bisection by examining the sign of $f(x_{k+1})$. This algorithm is known as "Dekker's method."

The strategy above strategy attempts to combine the unconditional convergence of bisection with the stronger root estimates of the secant method. In many cases it is successful, but its convergence rate is somewhat difficult to analyze; specialized failure modes can reduce this method to linear convergence or worse–in fact, in some cases bisection surprisingly can converge more quickly! Other techniques, e.g. "Brent's method," make bisection steps more often to avoid this case and can exhibit guaranteed behavior at the cost of a somewhat more complex implementation.

7.1.8 Single-Variable Case: Summary

We now have presented and analyzed a number of methods for solving $f(x^*) = 0$ in the single-variable case. It is probably obvious at this point that we only have scraped the surface of such techniques; many iterative schemes for root-finding exist, all with different guarantees, convergence rates, and caveats. Regardless, through our experiences we can make a number of observations:

- Due to the possible generic form of f, we are unlikely to be able to find roots x^* exactly and instead settle for iterative schemes.
- We wish for the sequence x_k of root estimates to reach x^* as quickly as possible. If E_k is an error bound, then we can characterize a number of convergence situations assuming $E_k \to 0$ as $k \to \infty$. A complete list of conditions that must hold when k is large enough is below:
  1. Linear convergence: $E_{k+1} \le CE_k$ for some $C < 1$
  2. Superlinear convergence: $E_{k+1} \le CE_k^r$ for $r > 1$ (now we do not require $C < 1$ since if E_k is small enough, the r power can cancel the effects of C)
  3. Quadratic convergence: $E_{k+1} \le CE_k^2$
  4. Cubic convergence: $E_{k+1} \le CE_k^3$ (and so on)
- A method might converge more quickly but during each individual iteration require additional computation; for this reason, it may be preferable to do more iterations of a simpler method than fewer iterations of a more complex one.

7.2 Multivariable Problems

Some applications may require solving a more general problem $f(\vec{x}) = \vec{0}$ for a function $f : \mathbb{R}^n \to \mathbb{R}^m$. We have already seen one instance of this problem when solving $A\vec{x} = \vec{b}$, which is equivalent to finding roots of $f(\vec{x}) \equiv A\vec{x} - \vec{b}$, but the general case is considerably more difficult. In particular, strategies like bisection are difficult to extend since we now much guarantee that m different values are all zero *simultaneously*.

7.2.1 Newton's Method

Thankfully, one of our strategies extends in a straightforward way. Recall that for $f : \mathbb{R}^n \to \mathbb{R}^m$ we can write the *Jacobian* matrix, which gives the derivative of each component of f in each of the coordinate directions:

$$(Df)_{ij} \equiv \frac{df_i}{dx_j}$$

We can use the Jacobian of f to extend our derivation of Newton's method to multiple dimensions. In particular, the first-order approximation of f is given by:

$$f(\vec{x}) \approx f(\vec{x}_k) + Df(\vec{x}_k) \cdot (\vec{x} - \vec{x}_k).$$

Substituting the desired $f(\vec{x}) = \vec{0}$ yields the following linear system for the next iterate $\vec{x}_{k+1}$:

$$Df(\vec{x}_k) \cdot (\vec{x}_{k+1} - \vec{x}_k) = -f(\vec{x}_k)$$

This equation can be solved using the pseudoinverse when $m < n$; when $m > n$ one can attempt least-squares but the existence of a root and convergence of this technique are both unlikely. When Df is square, however, corresponding to $f : \mathbb{R}^n \to \mathbb{R}^n$, we obtain the typical iteration for Newton's method:

$$\vec{x}_{k+1} = \vec{x}_k - [Df(\vec{x}_k)]^{-1} f(\vec{x}_k),$$

where as always we do not explicitly compute the matrix $[Df(\vec{x}_k)]^{-1}$ but rather use it to signal solving a linear system.

Convergence of fixed-point methods like Newton's method that iterate $\vec{x}_{k+1} = g(\vec{x}_k)$ requires that the maximum-magnitude eigenvalue of the Jacobian Dg be less than 1. After verifying that assumption, an argument similar to the one-dimensional case shows that Newton's method can have quadratic convergence near roots $\vec{x}^*$ for which $Df(\vec{x}^*)$ is nonsingular.

7.2.2 Making Newton Faster: Quasi-Newton and Broyen

As m and n increase, Newton's method becomes very expensive. For each iteration, a *different* matrix $Df(\vec{x}_k)$ must be inverted; because it changes so often, pre-factoring $Df(\vec{x}_k) = L_k U_k$ does not help.

Some *quasi-Newton* strategies attempt to apply different approximation strategies to simplify individual iterations. For instance, one straightforward approach might reuse Df from previous iterations while recomputing $f(\vec{x}_k)$ under the assumption that the derivative does not change very quickly. We will return to these strategies when we discuss the application of Newton's method to optimization.

Another option is to attempt to parallel our derivation of the secant method. Just as the secant method still contains division, such approximations will not necessarily alleviate the need to invert a matrix, but they do make it possible to carry out optimization without explicitly calculating the Jacobian Df. Such extensions are not totally obvious, since divided differences do not yield a full approximate Jacobian matrix.

Recall, however, that the *directional derivative* of f in the direction $\vec{v}$ is given by $D_{\vec{v}}f = Df \cdot \vec{v}$. As with the secant method, we can use this observation to our advantage by asking that our approximation J of a Jacobian satisfy

$$J \cdot (\vec{x}_k - \vec{x}_{k-1}) \approx f(\vec{x}_k) - f(\vec{x}_{k-1}).$$

Broyden's method is one such extension of the secant method that keeps track not only of an estimate $\vec{x}_k$ of $\vec{x}^*$ but also a matrix J_k estimating the Jacobian; initial estimates J_0 and $\vec{x}_0$ both must be supplied. Suppose we have a previous estimate J_{k-1} of the Jacobian from the previous iteration. We now have a new data point $\vec{x}_k$ at which we have evaluated $f(\vec{x}_k)$, so we would like to update J_{k-1} to a new Jacobian J_k taking into account this new observation. One reasonable model is to ask that the new approximation be as similar to the old approximation except in the $\vec{x}_k - \vec{x}_{k-1}$ direction:

$$\begin{aligned} \text{minimize}_{J_k} \quad & \|J_k - J_{k-1}\|^2_{\text{Fro}} \\ \text{such that} \quad & J_k \cdot (\vec{x}_k - \vec{x}_{k-1}) = f(\vec{x}_k) - f(\vec{x}_{k-1}) \end{aligned}$$

To solve this problem, define $\Delta J \equiv J_k - J_{k-1}$, $\Delta\vec{x} \equiv \vec{x}_k - \vec{x}_{k-1}$, and $\vec{d} \equiv f(\vec{x}_k) - f(\vec{x}_{k-1}) - J_{k-1} \cdot \Delta\vec{x}$. Making these substitutions yields the following form:

$$\begin{aligned} \text{minimize}_{\Delta J} \quad & \|\Delta J\|^2_{\text{Fro}} \\ \text{such that} \quad & \Delta J \cdot \Delta\vec{x} = \vec{d} \end{aligned}$$

If we take $\vec{\lambda}$ to be a Lagrange multiplier, this minimization is equivalent to finding critical points of the Lagrangian Λ:

$$\Lambda = \|\Delta J\|^2_{\text{Fro}} + \vec{\lambda}^\top(\Delta J \cdot \Delta\vec{x} - \vec{d})$$

Differentiating with respect to $(\Delta J)_{ij}$ shows:

$$0 = \frac{\partial \Lambda}{\partial(\Delta J)_{ij}} = 2(\Delta J)_{ij} + \lambda_i(\Delta\vec{x})_j \implies \Delta J = -\frac{1}{2}\vec{\lambda}(\Delta\vec{x})^\top$$

Substituting into $\Delta J \cdot \Delta\vec{x} = \vec{d}$ shows $\vec{\lambda}(\Delta\vec{x})^\top(\Delta\vec{x}) = -2\vec{d}$, or equivalently $\vec{\lambda} = {-2\vec{d}}/{\|\Delta\vec{x}\|^2}$. Finally, we can substitute to find:

$$\Delta J = -\frac{1}{2}\vec{\lambda}(\Delta\vec{x})^\top = \frac{\vec{d}(\Delta\vec{x})^\top}{\|\Delta x\|^2}$$

Expanding out our substitution shows:

$$\begin{aligned} J_k &= J_{k-1} + \Delta J \\ &= J_{k-1} + \frac{\vec{d}(\Delta\vec{x})^\top}{\|\Delta x\|^2} \\ &= J_{k-1} + \frac{(f(\vec{x}_k) - f(\vec{x}_{k-1}) - J_{k-1} \cdot \Delta\vec{x})}{\|\vec{x}_k - \vec{x}_{k-1}\|^2}(\vec{x}_k - \vec{x}_{k-1})^\top \end{aligned}$$

Thus, Broyden's method simply alternates between this update and the corresponding Newton step $\vec{x}_{k+1} = \vec{x}_k - J_k^{-1} f(\vec{x}_k)$. Additional efficiency in some cases can be gained by keeping track of the matrix J_k^{-1} explicitly rather than the matrix J_k, which can be updated using a similar formula.

7.3 Conditioning

We already showed in Example 1.7 that the condition number of root-finding in a single variable is:

$$\text{cond}_{x^*} f = \frac{1}{|f'(x^*)|}$$

As illustrated in Figure NUMBER, this condition number shows that the best possible situation for root-finding occurs when f is changing rapidly near x^*, since in this case perturbing x^* will make f take values far from 0.

Applying an identical argument when f is multidimensional shows a condition number of $\|Df(\vec{x}^*)\|^{-1}$. Notice that when Df is not invertible, the condition number is *infinite*. This oddity occurs because to first order perturbing $\vec{x}^*$ preserves $f(\vec{x}) = \vec{0}$, and indeed such a condition can create challenging root-finding cases like that shown in Figure NUMBER.

7.4 Problems

Many possibilities, including:

- Many possible fixed point iteration schemes for a given root-finding problem, graphical version of fixed point iteration
- Mean field iteration in ML
- Muller's method – complex roots
- Higher-order iterative methods – Householder methods
- Interpretation of eigenstuff as root-finding
- Convergence of secant method
- Roots of polynomials
- Newton-Fourier method (!)
- "Modified Newton's method in case of non-quadratic convergence"
- Convergence –¿ spectral radius for multidimensional Newton; quadratic convergence
- Sherman-Morrison update for Broyden