Nonlinear Systems

Quantitative Macroeconomics

Raül Santaeulàlia-Llopis

MOVE-UAB and Barcelona GSE

Fall 2017

Introduction

- Equilibria are often expressed as nonlinear systems,
  - A Root (a zero) of $f : \Re^n \to \Re^n$ is,
$$f(x) = 0$$
  - A fixed point of $f : \Re^n \to \Re^n$ is
$$f(x) = x$$
    But see the fixed point of $f(x)$ is the root of $f(x) - x = 0$.
- We examine numerical methods for solving these nonlinear equations in 1-dimensional and n-dimensional problems , $n < \infty$.

Univariate Problems

- Goal: Solve $f(x) = 0$ with $f : \Re \to \Re$ [1]

  This is 1 equation and 1 unknown.

- We want to learn how to solve that nonlinear equation.

[1] These slides rely heavily on Chapter 5 in Judd (1998).

Bisection Method

- **Intermediate Value Theorem (IVT)**: if a continuous real-valued f that is defined on an interval assumes two distinct values, then it must assume all values in between.

  In particular, if f is continuous, and $f(a)$ and $f(b)$ have different signs, then f must have *at least one* root $x \in [a, b]$.

- The bisection method uses this result repeatedly to compute a zero.

- The workings of the **iteration scheme**,

  Consider the interval $[x^L, x^R]$, with $f(x^L) < 0$ and $f(x^R) > 0$. To bisect $[x^L, x^R]$ means to compute $x^M = \frac{x^L + x^R}{2}$, the midpoint of $[x^L, x^R]$, and evaluate f at x^M.

  - If $f(x^M) = 0$ we are done.
  - If $f(x^M) < 0$, the IVT says there is a zero of f in (x^M, x^R). We then bisect the interval (x^M, x^R).
  - If $f(x^M) > 0$, the IVT says there is a zero of f in (x^L, x^M). We then bisect the interval (x^L, x^M).

- Bisection applies this procedure repeatedly, constructing successively smaller intervals containing zero.

- Note: There could be zeros in (x^L, x^M) and (x^M, x^R), but our objective is to find a zero, not all zeros.

- **Stopping Rules:**

  A stopping rule computes some distance to a solution, then stops the iteration when that distance is small.

  In the case of bisection, is $x^R - x^L$ a good estimate of that distance?. It will overestimate the distance because the zero is bracketed.

We stop the iteration scheme in either of two cases:

- First, when the bracketing interval is so small that we do not care about any further precision.

  This is controlled by our choice of ϵ:

  - $\epsilon = 0$ is nonsense.
  - $\epsilon = 10^n$ is nonsense in machines with at most $n-1$ digits of accuracy.
  - If x^R and x^L are of the order 10^n, demanding a $x^R - x^L < \epsilon = 10^{n-k}$ with k bigger than the machine precision is nonsense. For example, given $n = 10$, choosing $k = 15$ if the machine has a 12-digit precision is nonsense. If our preferred choice for ϵ is that small, 10^{n-k}, we can call for a relative change test of the form $x^R - x^L < \epsilon(|x^R| + |x^L|)$ to adjust for the machine's precision.
  - If the solution is close to $x = 0$, and x^L and x^R converges to zero then $x^R - x^L < \epsilon(|x^R| + |x^L|)$ would have problems. We can solve this by adding a 1, hence, choosing $x^R - x^L < \epsilon(1 + |x^R| + |x^L|)$.

- Second, we stop when $f(x^M)$ is less than the expected error in calculating f.

  This is controlled by δ:

  - Some times the evaluation of f at x^M may be subject to a variety of numerical errors, then δ may be larger than the machine precision.
  - Some times we may want an x that makes $f(x)$ small, not necessarily zero, then we choose δ to be the maximal value that satisfies that purposes.

- **Convergence:**

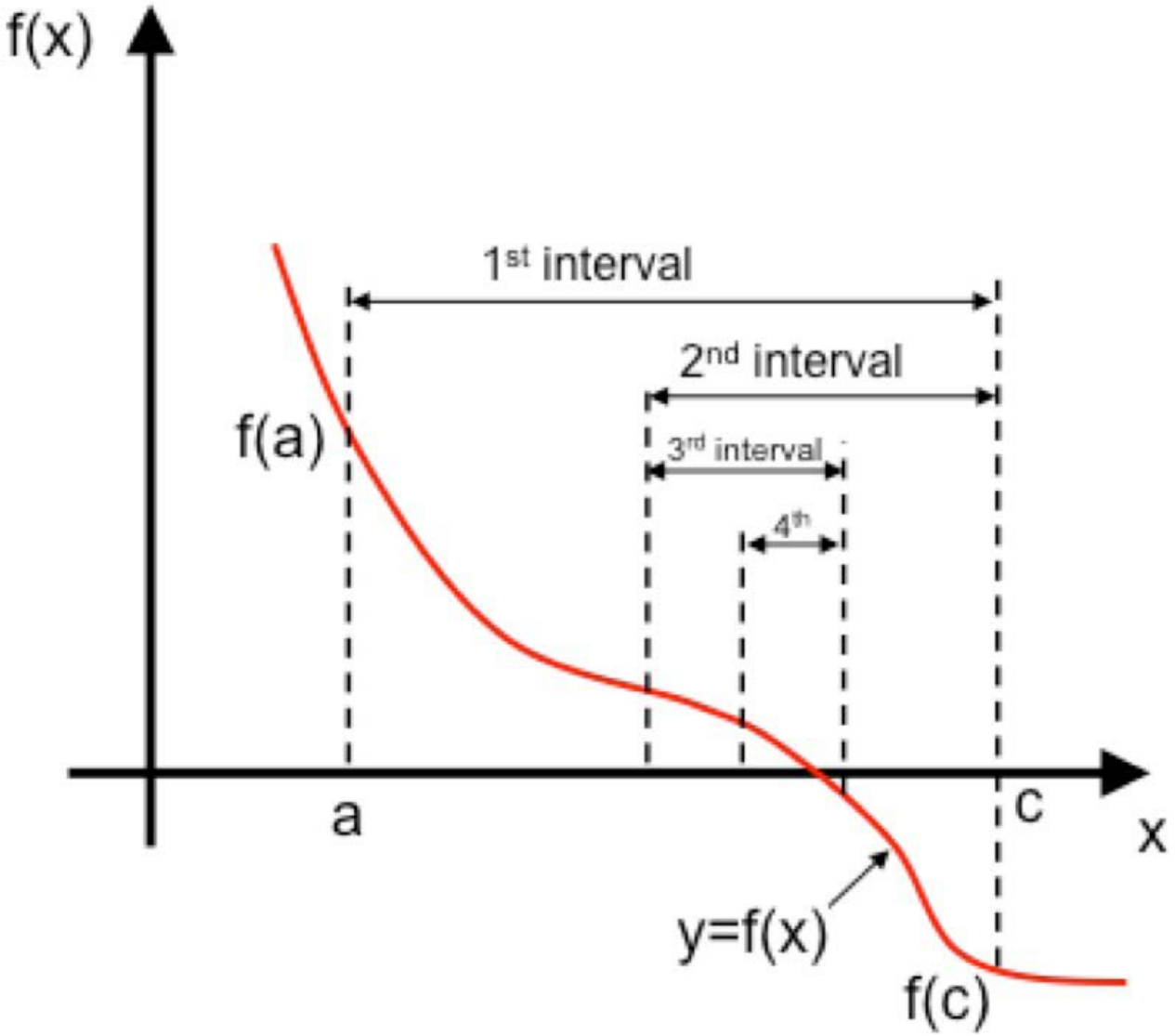
  - Bisection, have we found a pair x^L and x^R that bracket a zero, *always* converges.

  - It does so, but *slow*. It is a **linear convergent iteration**. Each iteration reduces the error by only 50%, that is, it takes more than three iterations to add a decimal digit of accuracy.

- **Bisection Algorithm:**

  - Step 0. Initialize and bracket a zero: find $x^L < x^R$ such that $f(x^L)f(x^R) < 0$, and choose stopping rule parameters, $\epsilon, \delta > 0$
  - Step 1. Compute midpoint: $x^M = \frac{x^L + x^R}{2}$ and $f(x^M)$
  - Step 2. Change the bounds: if $f(x^L)f(x^M) < 0$, then $x^R = x^M$ and do not change x^L; else then $x^L = x^M$ and do not change x^R.
  - Step 3. Check stopping rule: if $x^R - x^L \leq \epsilon(1 + |x^L| + |x^R|)$ or if $f(x^M) < \delta$, then STOP and report solution at x^M; else go to step 1.

A graphical representation of the bisection method (univariate)



[All credit for this figure goes to `https://calldutyblack.ru/method-of-bisection.php`]

Newton's Method

- Goal: Solve $f(x) = 0$ with $f : \Re \to \Re$ and $f \in C^1$.
- Newton's method reduces a nonlinear problem to a sequence of linear problems, where the roots are easy to compute.
    - Uses smoothness properties of f to approximate f locally in a linear fashion.
    - Approximates a root of f with the zeros of the linear approximations.

- The workings of the **iteration scheme**,
  - Suppose our current guess is x_k. Then we construct a linear approximation to f at x_k, $g(x)$,

$$g(x) \equiv f(x_k) + f'(x_k)(x - x_k))$$

  - Instead of solving for a zero of f, we solve for a zero of g, hoping that the two functions have similar zeros.
  - Our new guess, x_{k+1}, will be the zero of $g(x)$, that is,

$$0 \equiv f(x_k) + f'(x_k)(x - x_k)), \qquad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{1}$$

    The new guess will likely not be a zero of f, but the sequence x_k my converge to a zero of f. Sufficient conditions for convergence established in the next theorem.

- **Convergence:**

  - Theorem 2.1 Judd (1998): Suppose that f is $\mathcal{C}^2$ and that $f(x^*) = 0$. If x_0 is sufficiently close to x^*, $f'(x^*) \neq 0$ and $\left|\frac{f''(x^*)}{f'(x^*)}\right| < \infty$, the Newton sequence x_k defined by (1) converges to x^* and it is **quadratically convergent**, that is,

$$\lim_{k\to\infty} \sup \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2}.$$

- Trade-off between Bisection and Newton:
    - Bisection is safe, always converges, but it is slow. Newton gains in speed but may not converge.
    - We must search for the right balance between these 2 features, convergence and speed.

- **Stopping Rules:**

  We typically apply a 2-stage test,

  - First, we check if the last few iterations have moved much, and we conclude we have converged if $|x_k - x_{k-l}| < \epsilon(1 + |x_k|)$ for $l = 1, 2, ..., L$. Usually L=1.

  - Second, we check if $f(x_k)$ is 'nearly' zero. We stop if $f(x_k) \leq \delta$ for some prespecified δ.

- **Newton's Algorithm:**

  - Step 0. Initialize: Choose a starting point x_0 and set $k = 0$.
  - Step 1. Compute next iterate:

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

  - Step 2. Check stopping criterion: if $|x_k - x_{k-1}| < \epsilon(1 + |x_k|)$ go to step 3, else go to step 1.
  - Step 3. Report results and STOP: if $f(x_{k+1}) < \delta$ report success in finding a zero, otherwise report failure.

- **Problem with a Loose Stopping Rule**

  - Even if we find a point that satisfies the ϵ and δ stopping rules, it may still not be a zero, or close to one.

  - See $f(x) = x^6$ plotted in Figure 1.

  - The problem is that $f(x) = x^6$ is flat at its zero. If a function is nearly flat at a zero, convergence can be quite slow, and loose stopping rules may stop far from the true zero.

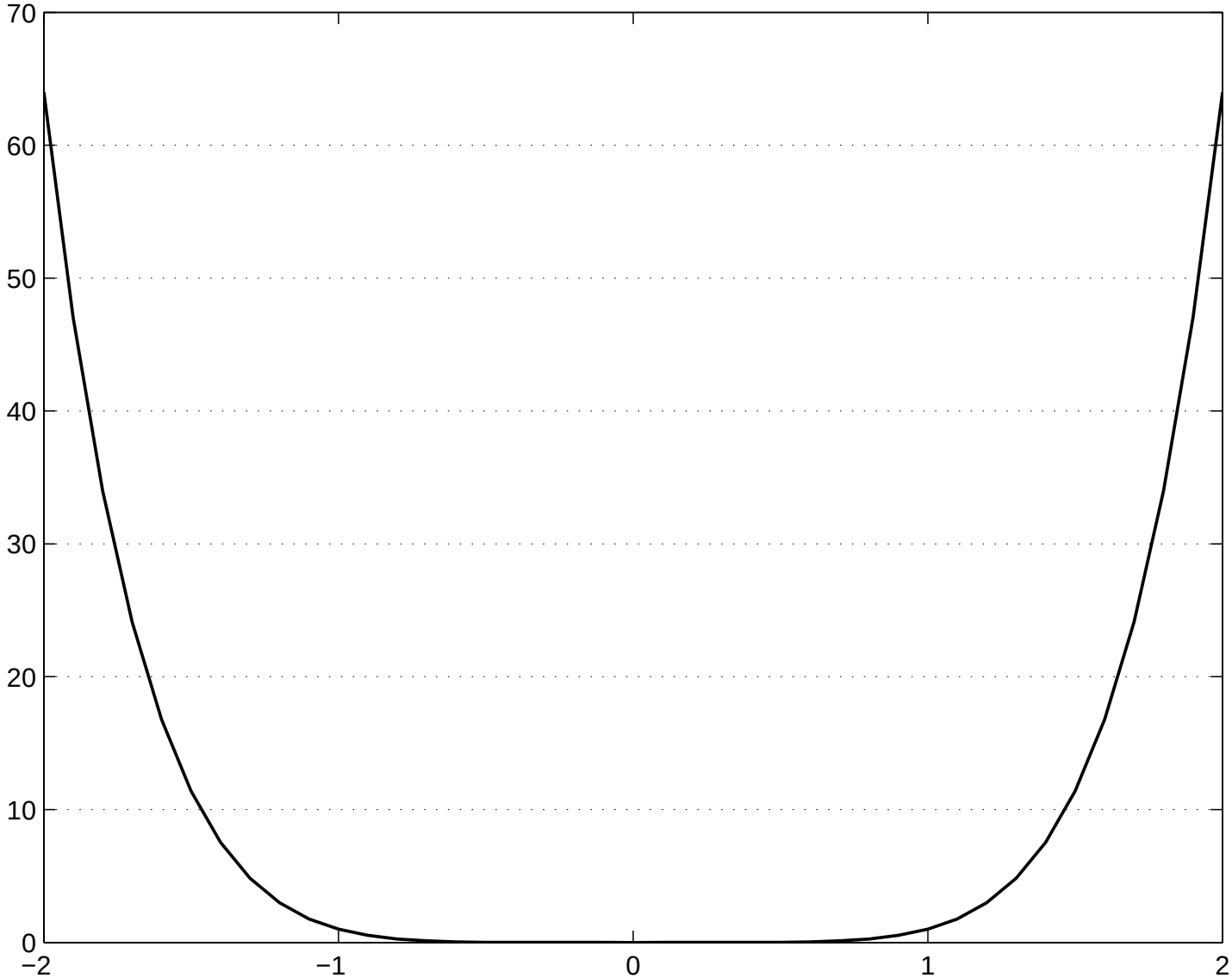


Figure: [Example 1] $f(x) = x^6$

- **Pathological Examples I**

  - See $f(x) = x^{\frac{1}{3}} e^{-(x^2)}$ plotted in Figure 2.
  - The first pathology strives from its iterative scheme,

$$x_{k+1} = x_k \left(1 - \frac{3}{1 - 6x_k^2}\right) \tag{2}$$

    - First, for x_k small, (2) is $x_{k+1} = -2x_k$, see Figure 3.
    - Second, for x_k large, (2) is $x_{k+1} = x_k \left(1 + \frac{2}{x_k^2}\right)$, which diverges but slowly, Figure 3.
    - That is, only if $x_0 = 0$, (2) converges to zero.

  - The second pathology derives from the fact that $e^{-(x^2)}$ factor squashes $f(x)$ at large $|x|$, leading Newton's method to believe that it is getting close to zero, while it does not (though the limit of is).

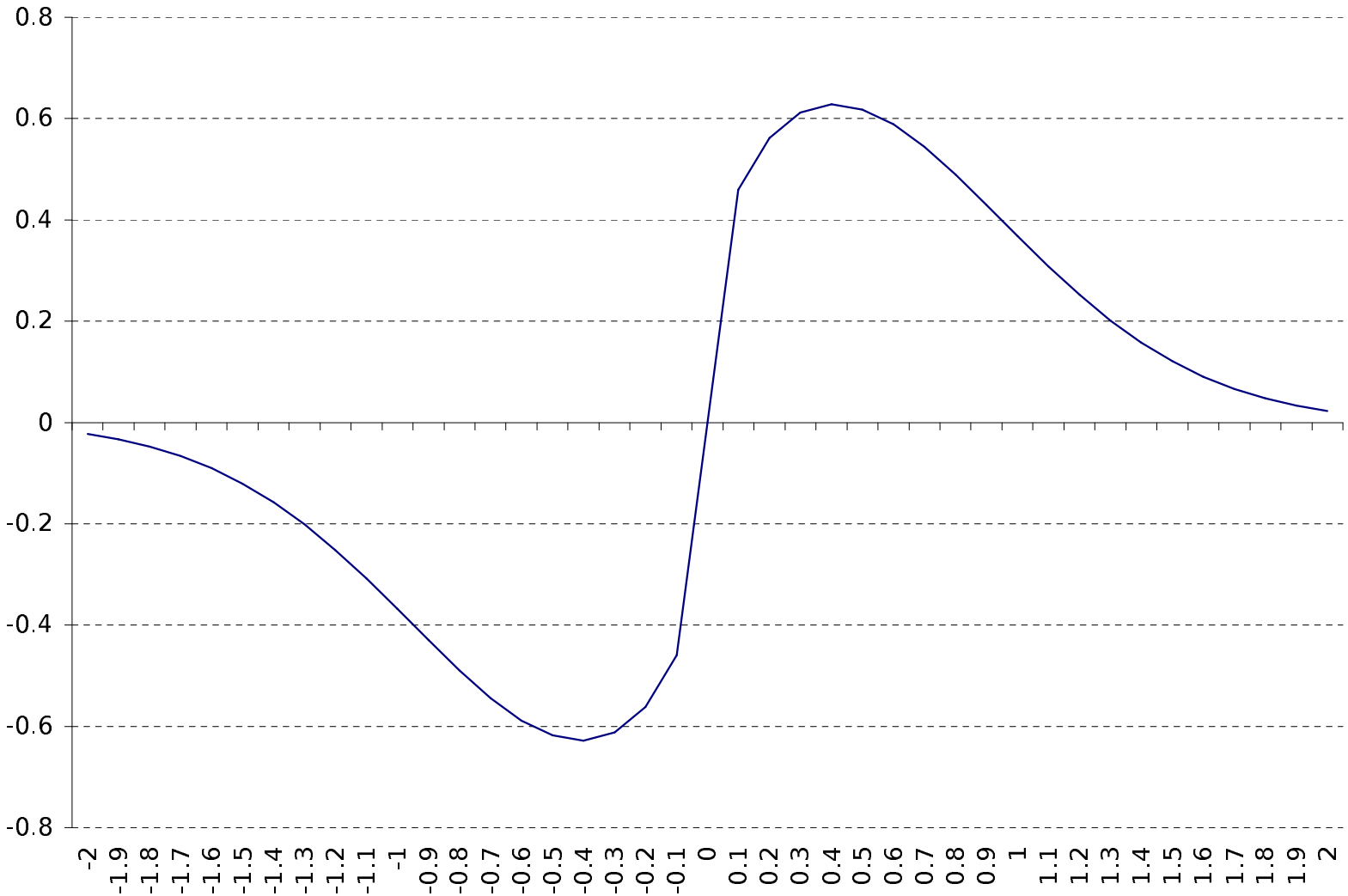


Figure: [Example 2] $f(x) = x^{\frac{1}{3}} e^{-(x^2)}$

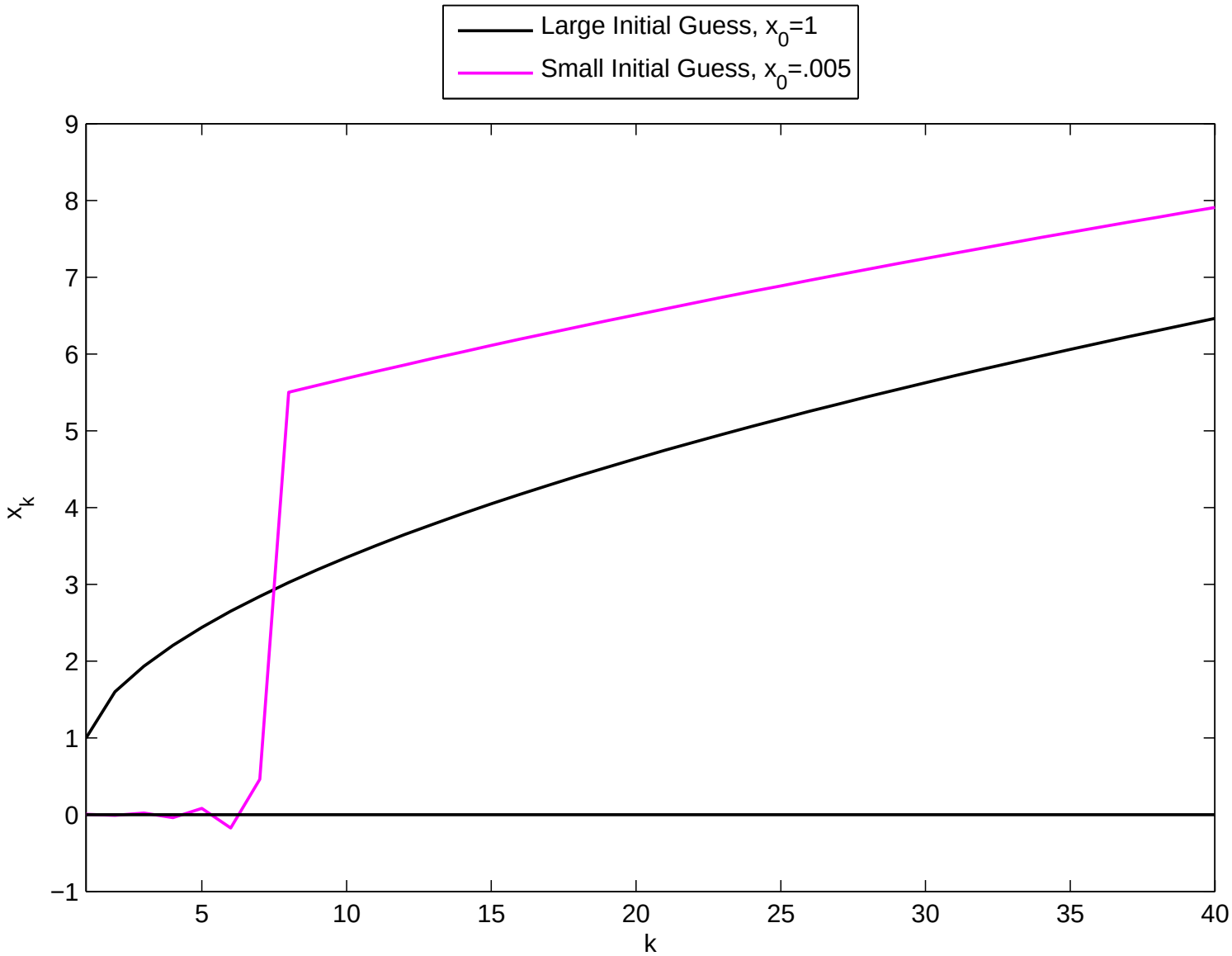
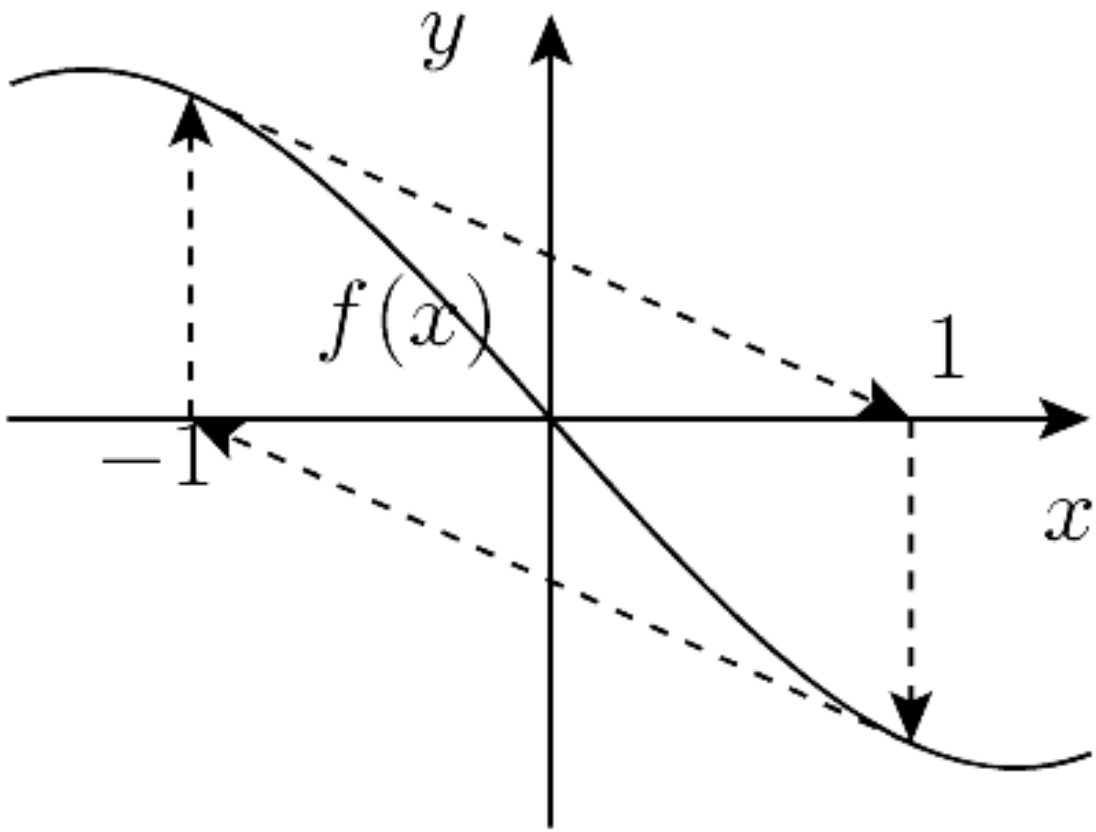


Figure: [Example 3] x_k path from Iteration Scheme in Equation (2)

- **Pathological Examples II**

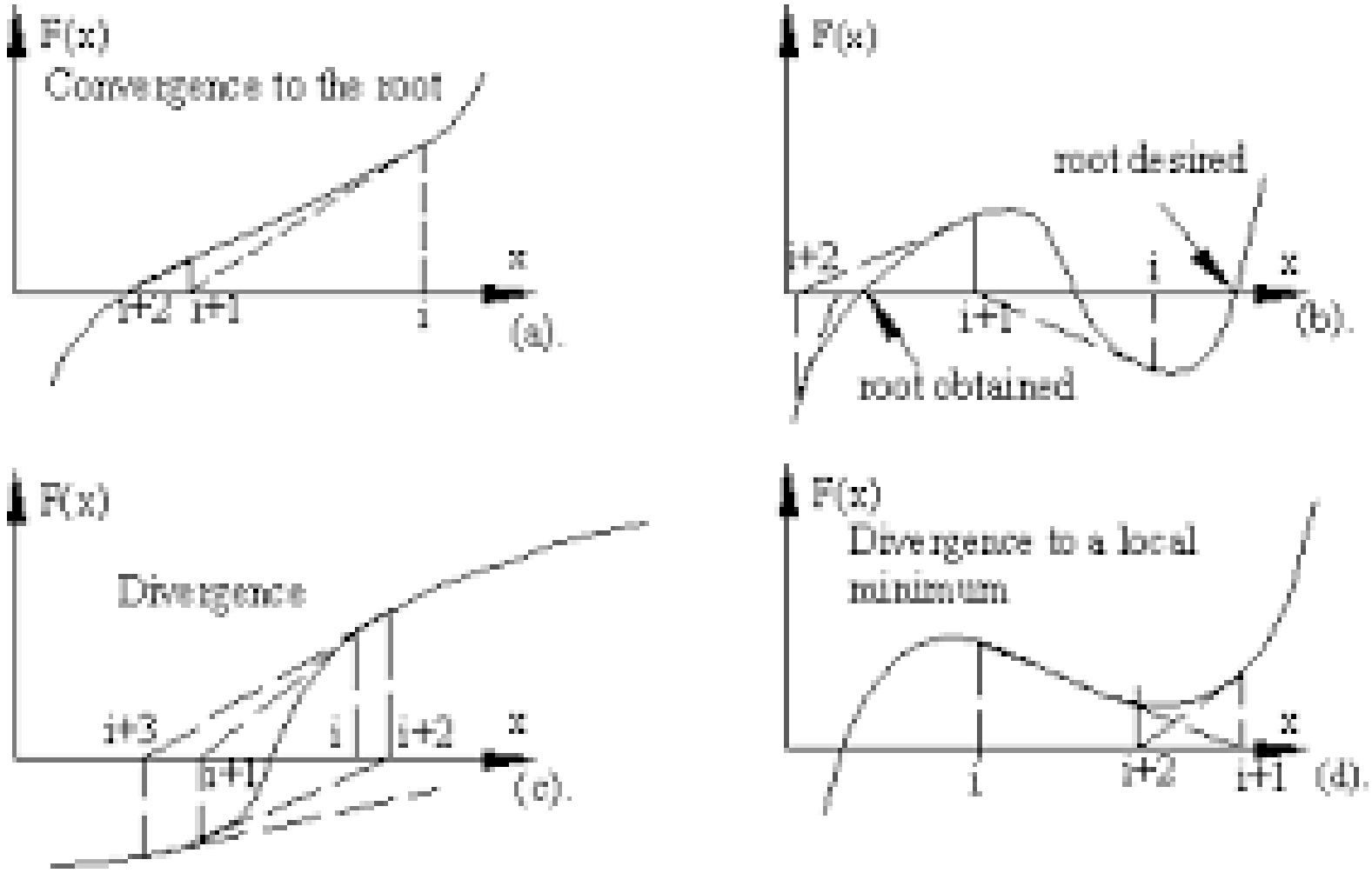
  - Newton's method can also converge to a cycle.



[Figure straight from Wiki]

  - In this cycle, Newton method would provide a root if it the initial guess is close enough to the root. This shows the importance of a good initial guess.

Other well- and ill-behaved examples:



[All credit for this figure goes to
`http://ocw.metu.edu.tr/pluginfile.php/3960/mod_resource/content/9/ch3/3-9.htm`]

Secant Method

- The computation of $f'(x)$ is necessary in the Newton method, but it may be costly.

- Substitute $f'(x)$ with an approximate of it, the slope of the secant of between x_{k-1} and x_k.

- The iteration scheme is then,

$$x_{k+1} = x_k - \frac{f(x_k)}{\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}} \tag{3}$$

- The algorithm for the secant method is, otherwise, identical to the Newton method.

- Convergence is slower in terms of the # of required evaluations of f because of the secant approximation to the derivative.

- However, the running time can be much less because the secant method never evaluates f'.

- The convergence rate is between linear and quadratic.

- Theorem 3.1 Judd (1998). If $f(x^*) = 0$, $f'(x^*) \neq 0$ and $f'(x)$ and $f''(x)$ are continuous near x^*, then the secant method converges at the rate $\frac{(1+5^{.5})}{2}$, that is

$$\lim_{k \to \infty} \sup \frac{|x_{k+1} - x^*|}{|x_k - x^*|^{\frac{(1+5^{.5})}{2}}}.$$

Fixed-Point Iteration

- Recall we can always rewrite root finding problems as fixed-point problems.
- Fixed-point problems, $f(x) = x$, suggests the iteration

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

- Consider equation

$$x^3 - x - 1 = 0 \tag{4}$$

- Equation (4) can be rewritten in fixed-point form as:
  - either $x = (x+1)^{\frac{1}{3}}$

$$x_{k+1} = (x_k + 1)^{\frac{1}{3}} \tag{5}$$

  - or $x = x^3 - 1$

$$x_{k+1} = x_k^3 - 1 \tag{6}$$

- If we take $x_0 = 1$, then (5) converges to a solution while (6) diverges to $-\infty$. That is, some transformations can turn unstable schemes into stable schemes.

- Remark: The secant method and the fixed-point iteration method do not directly generalize to multivariate problems.

Multivariate Problems

- Goal: Solve $f(x) = 0$ with $f : \Re^n \to \Re^n$,

$$\begin{aligned} f^1(x_1, x_2, ..., x_n) &= 0, \\ f^2(x_1, x_2, ..., x_n) &= 0, \\ &\vdots \\ f^n(x_1, x_2, ..., x_n) &= 0 \end{aligned}$$

  This is a list of n equations and n unknowns.

- We want to learn how to solve that nonlinear system of equations.

Gauss-Jacobi

- Given a known value of x^k(i.e., a vector), we use the ith equation to compute the ith component of unknown x^{k+1}, the next iterate.

- Formally x^{k+1} is defined in terms of x^k by

$$\begin{aligned} f^1(x_1^{k+1}, x_2^k, \ldots x_{n-1}^k, x_n^k) &= 0, \\ f^2(x_1^k, x_2^{k+1}, \ldots x_{n-1}^k, x_n^k) &= 0, \\ &\vdots \\ f^n(x_1^k, x_2^k, \ldots x_{n-1}^k, x_n^{k+1}) &= 0 \end{aligned}$$

- Each equation above is a single nonlinear equation with one unknown x_i^{k+1}.
- That is, Gauss-Jacobi algorithm reduces the problem of solving for n unknowns simultaneously in n equations to that of repeatedly solving n equations with one unknown $\rightarrow$ we can apply the univariate methods learned in the previous section.

- *Indexing*

  - Gauss-Jacobi is affected by the indexing scheme for the variables and equations. Which equation should we use for which variable and in which order?.

  - There are $n!$ possible combinations and no natural choice.

  - It may help to choose an indexing that resembles back-substitution. That is, if some equation depends on only one unknown, then that equation should be equation 1 and that variable should be variable 1.

- *Linear Gauss-Jacobi method*

  - There is little point in solving each x_i^{k+1} precisely, since we must solve each equation again in the next iteration.
  - We could solve for each x_i^{k+1} a bit loosely in order to save steps.
  - One way is to take a single Newton step to approximate each x_i^{k+1}.
  - The resulting scheme, known as the *linear Gauss-Jacobi method* is,

$$x_i^{k+1} = x_i^k - \frac{f^i(x^k)}{f^i_{x_i}(x^k)}, \qquad i = 1, ..., n$$

Gauss-Seidel

- In Gauss-Jacobi we have used the new guess of x_i, x_i^{k+1}, only after we have computed the entire vector of new values x^{k+1}.

- In Gauss-Seidel, however, we use the new guess of x_i, x_i^{k+1}, as soon as it becomes available

- Formally x^{k+1} is defined in terms of x^k (and x_i^{k+1}) by

$$\begin{aligned} f^1(x_1^{k+1}, x_2^k, ..., x_{n-1}^k, x_n^k) &= 0, \\ f^2(x_1^{k+1}, x_2^{k+1}, ..., x_{n-1}^k, x_n^k) &= 0, \\ &\vdots \\ f^n(x_1^{k+1}, x_2^{k+1}, ..., x_{n-1}^k, x_n^{k+1}) &= 0 \end{aligned}$$

- Again, we solve for x_i^{k+1} sequentially, but we immediately use each new component.

- *Indexing*

    - Gauss-Seidel is affected by the indexing scheme for the variables and equations even more than in Gauss-Jacobi, because now indexing also affects the way in which later results depend on earlier ones.

- *Linear Gauss-Seidel method*

  - Again, there is little point in solving each x_i^{k+1} precisely, since we must solve each equation again in the next iteration.
  - Again, we could solve for each x_i^{k+1} a bit loosely in order to save steps.
  - Again, one way is to take a single Newton step to approximate each x_i^{k+1}.
  - The resulting scheme, known as the *linear Gauss-Seidel method* is,

$$x_i^{k+1} = x_i^k - \frac{f^i}{f^i_{x_i}}(x_i^{k+1}, ..., x_{i-1}^{k+1}, x_i^k, ..., x_n^k), \qquad i = 1, ..., n$$

- Few remarks on Gaussian methods,

  - Gaussian methods (either Jacobi or Seidel), while often used, pose some problems.

  - Methods are most useful when the system is diagonally dominant — non-convergence may arise otherwise.

  - We can apply extrapolation and acceleration methods to attain or accelerate convergence. However, convergence is at best linear.

- **Stopping Rules for Multivariate Systems**

  - If we are using an iterative scheme $x^{k+1} = G(x^k)$ (as GJacobi) and we want to stop when $||x^k - x^*|| < \epsilon$, we must at least continue until $||x^k - x^*|| < (1-\beta)\epsilon$ where we can approximate $\beta = \max\{\frac{||x^{k-j+1} - x^k||}{||x^{k-j} - x^k||}, \quad j = 1, ..., L\}$

  - We also want to check that $f(x^k)$ is close to zero, that is, $||f(x^k)|| \leq \delta$ for some small δ. This $||f(x^k)|| \leq \delta$ implies trade-offs across different equations, that is, some equations in f may be closer to zero at x^k than they are at x^{k+1}, but a stopping rule will choose x^{k+1} over x^k if $||f(x^{k+1})|| \leq \delta \leq ||f(x^k)||$. To avoid this we can use the supnorm instead of the euclidean norm.

Fixed-Point Iteration

- We have strong constructive existence theory for the fixed points of contraction mappings.

- Check our Dynamic Programming Slides.

Newton's Method

- We start supplying a guess x^0 for the root of f.

- Given x^k, the subsequent iterate x^{k+1} is computed by solving the linear rootfinding problem obtained by replacing f with its first-order Taylor approximation about x^k:

$$f(x) \approx f(x^k) + f'(x^k)\left(x - x^k\right) = 0$$

  where $f'(x^k)$ is the Jacobian.

- This approach yields the Newton iteration scheme,

$$x^{k+1} \leftarrow x^k - \left[f'(x^k)\right]^{-1} f(x^k)$$

- Newton's method converges if f is continuously differentiable and if the initial value x^0 is 'sufficiently' close to ta root of f at with f' is invertible.

- There is however, no generally practical formula for determining what 'sufficiently' close is.

- Theorem 5.5.1. If $f(x^*) = 0$, $det(f'(x^*)) \neq 0$ and $f'(x^*)$ is Lipschitz near x^*, then for x^0 near x^*, the sequence defined above satisfies

$$\lim_{k \to \infty} \frac{||x^{k+1} - x^*||}{||x^k - x^*||^2} < \infty$$

- As in univariate cases, Newton method is quadratically convergent.

- **Multivariate Newton's Algorithm:**

  - Step 0. Initialize: Choose a starting point x_0 and set $k = 0$.

  - Step 1. Compute next iterate. Compute Jacobian $A_k = f'(x^k)$, solve $A_k s^k = -f(x^k)$ for s^k, and set $x^{k+1} = x^k + s^k$

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

  - Step 2. Check stopping criterion: if $||x_k - x_{k-1}|| < \epsilon(1 + ||x_k||)$ go to step 3, else go to step 1.

  - Step 3. Report results and STOP: if $||f(x_{k+1})|| < \delta$ report success in finding a zero, otherwise report failure.

- Newton's method can be robust to the starting value if f is well behaved.

- Newton's method can be very sensitive to starting value, however, if the function behaves erratically.

- Finally, in practice it is not sufficient for f' to be merely invertible at the root. If f' is invertible but ill conditioned, then rounding errors in the vicitinty of the root can make it difficult to compute a precise approximation to the root using Newton's method.

Secant (Broyden) Method

- This is the most popular quasi-Newton method and multivariate generalization of the univariate secant method.

- Broyden's method generates a sequence of vectors x^k and matrices A^k that approximate the root of f and the Jacobian f' at the root, respectively.

- **Iteration Scheme**. We begin guessing x^0 for the root of the function and a guess A^0 for the Jacobian of the function at the root.

- How do we choose A^0?

  - A^0 can be set equal to the numerial Jacobian of f at x^0.
  - A^0 can be set to a rescaled identity matrix. This approach typically will require more iterations to obtain a solution.

- Given x^k and A^k, one updates the root approximation by solving the linear rootfinding problem obtained by replacing f with its first-order Taylor approximation about x^k:

$$f(x) \approx f(x^k) + A^k(x - x^k) = 0$$

This step yields the root approximation iteration rule:

$$x^{k+1} \leftarrow x^k - \left(A^k\right)^{-1} f(x^k)$$

- Broyden's method then updates the A^k by making the smallest possible change, measured in the Frobenius matrix norm, that is consistent with the *secant condition*, a condition that any reasonable Jacobian estimate should satisfy:

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

This condition yields the iteration rule:

$$A^{k+1} \leftarrow A^k - \left[f(x^{k+1}) - f(x^k) - A^k d^k\right] \frac{(d^k)^T}{(d^k)^T d^k}$$

where $d^k = x^{k+1} - x^k$.

Speeding it up.

- We can accelerate this method by avoiding the linear solve. We do so by retaining and updating the Broyden estimate of the inverse of the Jacobian, rather than that of the Jacobian itself.

- Broyden's method with the inverse update generates a sequence of vectors x^k and matrices B^k that approximate the root of f and the inverse Jacobian f'^{-1} at the root, respectively. It uses the iteration rule

$$x^{k+1} \leftarrow x^k - B^k f(x^k)$$

  and the inverse update rule,

$$B^{k+1} \leftarrow B^k + \frac{(d^k - u^k)(d^k)^T B^k}{(d^k)^T u^k}$$

  where $u^k = B^k \left[f(x^{k+1}) - f(x^k) \right]$.

- Most implementations of Broyden's methods employ the inverse update rule because of its speed advantage.

Convergence.

- Boryden's method converges if f is continuously differentiable, if x^0 is 'sufficiently' close to a root of f at which f' is invertible, and if A^0 and B^0 are 'sufficiently' close to the Jacobian or the inverse Jacobian of f at that root.

- Like Newton's method, the robustness of Broyden's depends on the regularity of f and its derivative.

- Broyden's method may also have difficulty computing a precise root estimate if f' is ill conditioned near the root.

- It is also important to note that the sequence approximants A^k and B^k need not, and typically do not, converge to the Jacobian and inverse Jacobian, even if x^k converge to a root of f.

- **Broyden's Algorithm:**

  - Step 0. Initialize: Choose a starting point x_0, an initial Jacobian guess $A_0 = I$ and set $k = 0$.

  - Step 1. Compute next iterate: Solve $A_k s^k = -f(x^k)$ for s^k, and set $x^{k+1} = x^k + s^k$

  - Step 2. Update Jacobian guess: Set $y_k = f(x^{k+1}) - f(x^k)$ and

$$A_{k+1} = A_k + \frac{(y_k - A_k s^k)(s^k)^T}{(s^k)^T s^k}$$

  - Step 3: Check stopping criterion: if $||x_k - x_{k-1}|| < \epsilon(1 + ||x_k||)$ go to step 4, else go to step 1.

  - Step 4. Report results and STOP: if $||f(x_{k+1})|| < \delta$ report success in finding a zero, otherwise report failure.

- Important remarks:
  - Outside of the special case of contraction mappings, none of these methods is globally convergent (we could easily construct cases where Newton's method cycles).
  - There are systematic ways to enhance convergence to a zero that combine minimization ideas with the nonlinear equation approach.

Enhancing Global Convergence

- There are strong connections between nonlinear equations and optimization problems.
- If $f(x)$ is $\mathcal{C}^2$, then the solution to $\min_x f(x)$ is also a solution to the system of first-order conditions $f'(x) = 0$
- The converse is sometimes true as well. We may find a function $F(x)$ such that $f(x) = F'(x)$, in which case the zeros of f are exactly the local minima of F. Such systems $f(x)$ are called *integrable*.
- Then, since we have globally convergent schemes for minimizing F we can use them to compute the zeros of f.
- Importantly, this approach has limited applicability because few systems $f(x)$ are integrable.

- In one general sense, nonlinear equation problems can be converted to optimization problems. Any solution to the system $f(x) = 0$ is also a global solution to

$$\min_x \sum_i^n f^i(x)^2$$

and any global minimum of $\sum_i^n f^i(x)^2$ is a solution to $f(x) = 0$.

Powell's Hybrid

- On the one hand, Newton's method will converge rapidly if it converges but it may diverge.
- On the other hand, the minimization idea above converges to something, but it may do so slowly, and it may converge to a point other than a solution to the nonlinear system.
- Since these approaches have complementary strengths and weaknesses, we are tempted to develop a hybrid algorithm combining their ideas.

- If we define $SSR(x) = f^i(x)^2$, then the solutions to the nonlinear equation $f(x) = 0$ are exactly the global solutions to the minimization problem above.

- Perhaps, we can use values of $SSR(x)$ to indicate how ell we are doing and help restrain Newton's method when it does not appear to be working.

- **Powell's method** modifies Newton's method by checking if Newton step reduces the value of SSR.

  - Suppose that the current guess is x^k.
  - The Newton step is then $s^k = -f'(x^k)^{-1} f(x^k)$.
  - Newton's method takes $x^{k+1} = x^k + s^k$ without hesitation.
  - Powell's method checks $x^k + s^k$ before accepting it has the next iteration.
  - In particular, Powell's method will accept $x^k + s^k$ as the next iteration only if $SSR(x^k + s^k) < SSR(x^k)$, that is, only if there is some improvement in SSR.

- These methods will converge to a solution $f(x) = 0$ or will stop if they come too near a local minimum of SSR.

- If there is no such local minimum then we will get global convergence.

- If we do get stuck near a local minimum x^m, we will know that we are not at the global solution, since we will know that $f(x^m)$ is not zero and we can continue by choosing a new starting point.

Homotopy Continuation Methods

- Homotopy methods formalize the notion of deforming a simple problem into a hard problem, computing a series of zeros of the intervening problems in order to end with a zero to the problem of interest.

- This yields a globally convergent way to find the zeros in a multivariate problem.

- *Homotopy functions*, $H(x,t)$, $H:\Re^{n+1}\to\Re^n$, $H\in\mathcal{C}^0(\Re^{n+1})$.

- A *homotopy function* H that continuously deforms g into f is any continuous function H where

$$H(x,0)=g(x),\quad H(x,1)=f(x).$$

- In practice we take $H(x,0)$ to be a simple function with easily calculated zeros, and $H(x,1)$ to be the function whose zeros we want.

- For example

  - The *Newton homotopy* is $H(x,t) = f(x) - (1-t)f(x^0)$ for some x^0.
    - At $t = 0$, $H = f(x) - f(x^0)$ which has a zero at $x = x^0$.
    - At $t = 1$, $H = f(x)$.

    This is a simple homotopy, since the difference between $H(x,t)$ and $H(s,t)$ is proportional to $t - s$.

  - The *fixed-point homotopy* is $H(x,t) = (1-t)(x - x^0) + tf(x)$ for some x^0. It transforms the function $x - x^0$ into $f(x)$.

  - More generally, the *linear homotopy* is $H(x,t) = tf(x) + (1-t)g(x)$ which transforms g into f, since at $t = 0$, $H = g$ and at $t = 1$, $H = f$.

- **Generically Convergent Euler Homotopy Method:**

  - Step 0. Choose an $a \in \Re^n$ and form the homotopy, $H(a, t, x) = (1 - t)(x - a) + t(x - f(x))$. Let $x_0 = a, s_0 = 0$ and $t_0 = 0$. Choose step size ϵ.

  - Step 1. Given x_i, s_i and t_i compute x_{i+1} and t_{i+1}. Let $s_{i+1} = s_i + \epsilon$.

  - Step 2. If $t_{i+1} > 1$ go to Step 3; else go to Step 1.

  - Step 3. Stop and report the last iterate of x as the solution.

Remarks,

- Homotopy methods have good convergence properties, but they may perform very slowly. In many homotopy methods, each step involves computing a Jacobian; in such cases each step is as costly as a Newton step.

- Then, we can first use a homotopy method to compute a rough solution and then use it as the initial guess for Newton's method.

- Slow convergence of homotopy methods also implies it is not easy to satisfy their convergence criterion. It seems useful to apply Newton's method to the final iterate of a homotopy method as a natural step to improve on the homotopy result.

Judd, K. L. (1998): *Numerical Methods in Economics*. MIT Press.