1 Solving Equations

This chapter would benefit from an introduction, explaining what a system of equations is, what it means to solve one, why (in broad terms) it's important to know this for ML, and what concrete skills and knowledge the reader will gain from the chapter.

In this chapter we consider the methods for solving a set of M simultaneous equations of N variables:

If readers are unfamiliar with linear algebra, they may need the concept of functions on vectors explained.

$$\begin{cases} f_1(x_1, \ ) = f_1(\mathbf{x}) = 0 \\ \cdots\cdots\cdots \\ f_M(x_1, \cdots, x_N)) = f_M(\mathbf{x}) = 0 \end{cases}$$

add space

Readers will probably understand what a column vector is, but "N-D" should be defined as N-dimensional.

where$\mathbf{x} = [x_1, \cdots, x_N]^T \in \mathbb{R}^N$ is an N-D column vector, a point in the N-D vector space $\mathbb{R}^N$ spanned by $\{x_1, \cdots, x_N\}$. We further define $\mathbf{f} = [f_1, \cdots, f_M]^T$ as an M-D column vector, and represent this equation system more concisely as $\mathbf{f}(\mathbf{x}) = \mathbf{0}$.

Some students have trouble understanding the geometric applications of vectors and vector functions until they fully understand what a function of a vector is. Consider defining vector functions mathematically as well as geometrically. (E.g., a vector is a set of variables, and a function of the vector is a mathematical expression that assigns each combination of values of those variables to an outcome value. Then give an example!)

"Hypersurface" should be defined.

Geometrically, a function $f(\mathbf{x})$ defined over this space $\mathbb{R}^N$ is a hypersurface in the N+1 dimensional space, of which the N+1st dimension represents the function value $y = f(\mathbf{x})$ corresponding to a point $\mathbf{x}$ in the N-D space. The *roots* or *zeros* of function $f(\mathbf{x})$, i.e., the solutions of equation $f(\mathbf{x}) = 0$, are all the points on a hypercurve in the N-D space, as the intersection of the hypersurface $y = f(\mathbf{x})$ and the hyperplane $y = 0$, if they do intersect.

The concept of a function of a vector should be explained before the concept of a system of equations of vectors.

In general, no closed-form solution exists for a nonlinear equation system and the number of solutions is unknown. We therefore have to rely on numerical methods to find iteratively one or more such solutions if they do exist from an initial guess of the solution.

This paragraph belongs at the end of the section.

For example, in the special case of $M = N = 2$, functions $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ are two surfaces defined over the 2-D space spanned by x_1 and x_2, and the roots of each of the two functions are on a curve as the intersection of the corresponding surface and the 2-D space. The solutions of the system composed of the two equations $f_1 = 0$ and $f_2 = 0$ are the intersection of these two curves, if they do intersect, otherwise no solution exists.

Example: Consider a simultaneous equation system:

Keep notation consistent—y is an outcome variable, and entries in a vector are x_i.

$$\begin{cases} f_1(x, y) = x^2 + y^2 - 2 = 0 \\ f_2(x, y) = x + y - C = 0 \end{cases}$$

The first function $f_1(x, y)$ is a parabolic cone centrally symmetric to the vertical axis, and its roots form a circle $x^2 + y^2 = 2$ on the x-y plane centered at the origin (0, 0) with radius 1; the second function $f_2(x, y)$ is a plane through

the origin, and its roots form a straight line $y = C - x$ on the x-y plane. The solutions of the equation system are where the two curves intersect:

- If $C = 0$, there are two solutions $(1, -1)$ and $(-1, 1)$;
- If $C = 2$, there is only one solution $(1, 1)$;
- If $C = 3$, the two curves do not intersect, i.e., no solution exists.

1.1 Linear Equation Systems

Consider a linear equation system of M equations and N variables:

$$\begin{cases} \sum_{j=1}^{N} a_{1j}x_j - b_1 = 0 \\ \cdots \\ \sum_{j=1}^{N} a_{Mj}x_j - b_M = 0 \end{cases} \tag{1.2}$$

which can be expressed in vector form:

$$\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0} \tag{1.3}$$

where

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{M1} & \cdots & a_{MN} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_M \end{bmatrix} \tag{1.4}$$

In general, the existence and uniqueness of the solution $\mathbf{x}$ can be determined by the *fundamental theorem of linear algebra*, (see Table 3 in Section A.19), based on the rank R of the coefficient matrix $\mathbf{A}$ and whether the system is underdetermined (underconstrained) if $M < N$, or overdetermined (overcostrained) if $M > N$.

Specifically if $R = M = N$, $\mathbf{A}$ is a full rank square matrix and its inverse $\mathbf{A}^{-1}$ exists. Then the system has a unique solution $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

But if $M > N = R$ and $\mathbf{b}$ is not in the column space of $\mathbf{A}$, the system is overconstrained without a solution. In this case we can still find an optimal approximated solution. We first define the *residual* of the system as $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$, and then define the *sum-of-squares error (SSE)* as:

Before getting into the math, describe in plain language what you're going to do: create an equation that, when minimized, will provide values for the parameters that result in the minimum difference between **b** and **Ax** (since, in the true answer, we would expect that difference to be 0).

$$\begin{aligned} \varepsilon(\mathbf{x}) &= \frac{1}{2}||\mathbf{r}||^2 = \frac{1}{2}\mathbf{r}^T\mathbf{r} = \frac{1}{2}(\mathbf{b} - \mathbf{A}\mathbf{x})^T(\mathbf{b} - \mathbf{A}\mathbf{x}) \\ &= \frac{1}{2}\left(\mathbf{b}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}\right) \end{aligned} \tag{1.5}$$

To find the optimal solution $\mathbf{x}^*$ that minimizes ε, we set its derivative with respect to $\mathbf{x}$ to zero (see section A.18 for differentiation with respect to a vector variable):

$$\begin{aligned} \frac{d}{d\mathbf{x}}\varepsilon(\mathbf{x}) &= \frac{1}{2}\frac{d}{d\mathbf{x}}||\mathbf{r}||^2 = \frac{1}{2}\frac{d}{d\mathbf{x}}\left(\mathbf{b}^T\mathbf{b} - \mathbf{x}^T\mathbf{A}^T\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}\right) \\ &= -\mathbf{A}^T\mathbf{b} + \mathbf{A}^T\mathbf{A}\mathbf{x} = 0 \end{aligned} \tag{1.6}$$

Solving this matrix equation we get

$$\mathbf{x} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^-\mathbf{b} \tag{1.7}$$

where $\mathbf{A}^- = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T$ is the *pseudo-inverse* (Section A.15) of the non-square matrix $\mathbf{A}$.

If the M equations are barely independent, is the $N \times N$ matrix $\mathbf{A}^T\mathbf{A}$ it still has a full rank $R = N$ and is invertible, but it is a near-singular matrix. In this case some of its eigenvalue[illegible] very close to zero, and correspondingly some of the eigenvalues of its inverse $(\mathbf{A}^T\mathbf{A})^{-1}$ may be huge. Consequentl[illegible]e system becomes ill-conditione[illegible] ill-posed, in the sense that a small change in the system (in either $\mathbf{A}$ or $\mathbf{b}$) due to noise may cause a large change in the solution $\mathbf{x}$.

This il[illegible]sted problem can be addressed by *regularization*, [illegible]which the solution $\mathbf{x}$ is unde[illegible]ntrol so that it will not take unreasonably high values. Specifically we construct an *objective function* that contains a penalty term for large $\mathbf{x}$, as well as the error term $\varepsilon = ||\mathbf{r}||^2/2$:

$$J(\mathbf{x}) = \frac{1}{2}||\mathbf{r}||^2 + \frac{\lambda}{2}||\mathbf{x}||^2 \tag{1.8}$$

By minimizing $J(\mathbf{x})$, we can obtain a solution $\mathbf{x}$ of small norm $||\mathbf{x}||$ as well as low error $\varepsilon(\mathbf{x})$. S[illegible]e as before, the solution $\mathbf{x}$ can be obtained by setting the derivative of $J(\mathbf{x})$ to zero

$$\frac{d}{d\mathbf{x}}J(\mathbf{x}) = -\mathbf{A}^T\mathbf{b} + \mathbf{A}^T\mathbf{A}\mathbf{x} + \lambda\mathbf{x} = \mathbf{0} \tag{1.9}$$

and solving the resulting equation to get:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b} \tag{1.10}$$

We see that even if $\mathbf{A}^T\mathbf{A}$ is near singular, matrix $\mathbf{A}^T\mathbf{A} + \lambda\mathbf{I}$ is not due to the additional term $\lambda\mathbf{I}$.

By adjusting the hyperparameter λ, we can make a proper tradeoff between accuracy and stability.

- Small λ: the solution is more accurate but less stable as it is more prone to noise, i.e., the variance error may be large. This is called *overfitting*.
- Large λ: the solution is less accurate but more stable as it is less affe[illegible]l by noise. This is called *underfitting*.

The issue of overfitting versus underfitting is of central importance in ma[illegible] learning in general, and will be more formally addressed while discussing va[illegible] regression and classification algorithms in some later Chapters.

I'm not familiar with the term "barely independent." If it's a technical term, it should be defined; if not, consider rephrasing to make your meaning clearer.

"Specifically" is not necessary here.

"Objective function" should be defined.

Before discussing what the solution is, describe in plain language what lambda is. (This would be a good place to define "hyperparameter.")

"Hyperparameter" should be defined.

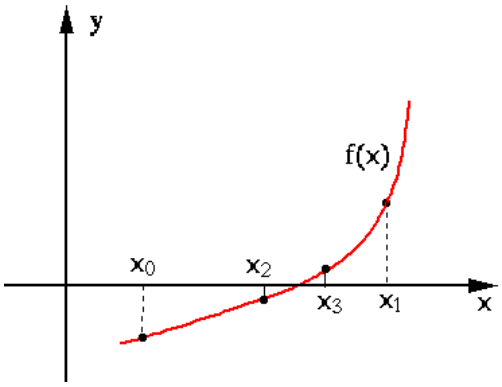


Figure 1.1 Bisection Search

Inconsistent capitalization. This should be either "The Bisection and Secant Methods" or "The bisection and secant methods."

1.2 The Bisection and Secant methods

The shift from linear equation systems to nonlinear ones is big, and it's easy to miss in this introduction. Adding some introductory text about nonlinear equations and why the problem of solving them is different than solving linear equation systems would help to ground the reader in what they're about to learn.

Here we consider a set of methods that find the solution x^* of a single-variable nonlinear equation $f(x) = 0$, by searching iteratively through a neighborhood of the domain, in which x^* is known to be located.

The bisection search

Starting out with a brief, broad description of what the bisection method does would help readers to understand the more detailed explanation.

This method requires two initial guesses $x_0 < x_1$ satisfying $f(x_0)f(x_1) < 0$. As $f(x_0)$ and $f(x_1)$ are on opposite sides of the x-axis $y = 0$, the solution x^* at which $f(x^*) = 0$ must reside somewhere in between of these two guesses, i.e., $x_0 < x^* < x_1$.

This explanation would be clearer if it broke down the method into smaller steps: Start with two end points; find the midpoint; check the sign of the midpoint; replace the endpoint with the same sign as the midpoint with the midpoint.

Given such two end points x_0 and x_1, we find another point x_2 in between and use it to replace one of the end points at which the function value $f(x)$ has the same sign as that of $f(x_2)$. The new search interval is $a = x_2 - x_0$ if $f(x_2)f(x_1) > 0$, or $b = x_1 - x_2$ if $f(x_0)f(x_2) > 0$, either of which is smaller than the previous interval $a + b = x_1 - x_0$. This process is then carried out iteratively until the solution is eventually approached, with a guaranteed convergence.

Define "convergence."

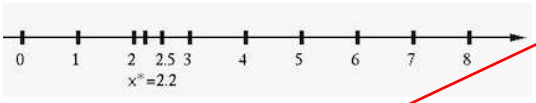
While any point between the two end points can be chosen for the next iteration, we want to avoid the worst possible case in which the solution always happens to be in the larger of the two sections a and b. We can therefore choose the middle point between the two end points, i.e., $x_2 = (x_0 + x_1)/2$, so that $a = b = (x_1 - x_0)/2$, i.e., the new search interval is always halved in each step of the iteration:

The method will be clearer if the final version of the algorithm is described, so instead of "another point," we find the midpoint. Then in the next paragraph, you can explain why the midpoint is used.

$$x_{n+1} = \frac{x_n + x_{n-1}}{2} \tag{1.11}$$

Here, without loss of generality, we have assumed $f(x_{n-1})f(x_{n+1}) > 0$ and x_{n-1} is replaced by x_{n+1}.

For example, we assume the root is located at $x^* = 2.2$ between the two initial values are $x_0 = 0$ and $x_1 = 8$, then we get



This figure would be clearer if the endpoints and midpoints used by the search were labeled.

Figure 1.2

n	0	1	2	3	4	5	6	7	8
x_n	0	8	4	2	3	2.5	2.25	2.125	$\cdots$
$\lvert e_n\rvert$	2.2	5.8	1.8	0.2	0.8	0.3	0.05	0.075	$\cdots$

(1.12)

Note that the error $|e_n| = |x_n - x^*|$ does not necessarily always reduce monotonically. However, as in each iteration the search interval is always halved, the worst possible error is also halved:

$$|e_{n+1}| = |x_{n+1} - x^*| \leq \frac{|x_n - x_{n-1}|}{2} = \frac{|x_0 - x_1|}{2^n} \approx \frac{|e_n|}{2} \tag{1.13}$$

i.e., the convergence of the iteration is linear (of order $p = 1$) and the rate of convergence is $1/2$:

"Order" and "rate" of convergence should be defined.

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = \lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} \leq \mu = \frac{1}{2} \tag{1.14}$$

Compared to other methods to be considered later, the bisection method converges rather slowly, but one of the advantages of the bisection method is that no derivative of the given function is needed. This means the given function $f(x)$ does not need to be differentiable.

The Secant method

Like as in the bisection method, here again we assume there are two initial values x_0 and x_1 available, but they do not have to satisfy $f(x_0)f(x_1) < 0$. Here the iteration is based on the zero-crossing of the secant line passing through the two points $f(x_0)$ and $f(x_1)$, instead of their middle point. The equation for the secant is:

This is confusingly worded. Instead of relying on the algorithm description from the bisection method, describe the steps of the secant method.

Undergraduate students are used to seeing equations for lines in point-slope format. Consider first defining the equation for the secant with y alone on the left side, and then explaining the traditional representation.

$$\frac{y - f(x_0)}{x - x_0} = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{1.15}$$

At the zero crossing, $y = 0$ and the equation above can be solved for x, the position of the zero-crossing:

$$x = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0) = x_0 - \frac{f(x_0)}{\hat{f}'(x_0)} \tag{1.16}$$

where $\hat{f}'(x_0)$ is the finite difference that approximates the derivative $f'(x_0)$:

$$\hat{f}'(x_0) = \frac{f(x_1) - f(x_0)}{x_1 - x_0} = \frac{\Delta f(x_0)}{\Delta x_0} \overset{\Delta x_0 \to 0}{\Longrightarrow} f'(x_0) \tag{1.17}$$

The new value x obtained above, as a better estimate of the root than either x_0 or x_1, is used to replace one of them:

Why is the new value of x a better estimate of the root? This is the heart of the method, and won't be obvious to an undergraduate reader. An explanation, after the algorithm has been outlined, would help the reader to understand why the method works.

This would be clearer to an undergraduate reader if instead of "x_i (i = 0, 1) satisfying f(x_i)f(x) > 0" were instead written in plain language describing which endpoint is replaced.

- If $f(x_0)f(x_1) < 0$, then x_i $(i = 0, 1)$ satisfying $f(x_i)f(x) > 0$ is replaced by $x_2 = x$ (same as the bisection method);
- If $f(x_0)f(x_1) > 0$, then x_i $(i = 0, 1)$ with greater $|f(x_i)|$ is replaced by $x_2 = x$.

This process is then carried out iteratively to approach the root:

$$x_{n+1} = x_n - \frac{f(x_n)}{\hat{f}'(x_n)} \tag{1.18}$$

In case $f(x_n) = f(x_{n-1})$, the approximated derivative is zero $\hat{f}'(x_n) = 0$, and x_{n+1} cannot be found. In such a case, a root may exist between x_n and x_{n-1}. Therefore the way to resolve this problem is to combine the bisection search with the secant method so that when $\hat{f}' = 0$, the algorithm will switch to bisection search. This is Dekker's method:

$$x_{n+1} = \begin{cases} x_n - f(x_n)/\hat{f}'(x_n) & \text{if } f(x_n) \neq f(x_{n-1}) \\ (x_n + x_{n-1})/2 & \text{otherwise} \end{cases} \tag{1.19}$$

These calculations are lengthy and complex, and may be intimidating to an undergraduate who is unfamiliar with orders of convergence and with the styles of proof used in academic articles. Consider moving the calculations to an appendix, and just providing the readers with the order and rate of convergence.

We now consider the order of convergence of the secant method. Let x^* be the root at which $f(x^*) = 0$. The error of x_{n+1} is:

$$\begin{aligned} e_{n+1} = x_{n+1} - x^* &= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) - x^* \\ &= \frac{(x_{n-1} - x^*)f(x_n) - (x_n - x^*)f(x_{n-1})}{f(x_n) - f(x_{n-1})} = \frac{e_{n-1}f(x_n) - e_n f(x_{n-1})}{f(x_n) - f(x_{n-1})} \end{aligned} \tag{1.20}$$

Consider the Taylor expansion of $f(x_n)$:

$$f(x_n) = f(x^*+e_n) = f(x^*)+f'(x^*)e_n+\frac{1}{2}f''(x^*)e_n^2+O(e_n^3) = f'(x^*)e_n+\frac{1}{2}f''(x^*)e_n^2+O(e_n^3) \tag{1.21}$$

(as $f(x^*) = 0$) and similarly

$$f(x_{n-1}) = f'(x^*)e_{n-1} + \frac{1}{2}f''(x^*)e_{n-1}^2 + O(e_{n-1}^3) \tag{1.22}$$

Substituting these into the expression for e_{n+1} above we get

$$\begin{aligned} e_{n+1} &= \frac{e_{n-1}[f'(x^*)e_n + \frac{1}{2}f''(x^*)e_n^2 + O(e_n^3)] - e_n[f'(x^*)e_{n-1} + \frac{1}{2}f''(x^*)e_{n-1}^2 + O(e_{n-1}^3)]}{[f'(x^*)e_n + \frac{1}{2}f''(x^*)e_n^2 + O(e_n^3)] - [f'(x^*)e_{n-1} + \frac{1}{2}f''(x^*)e_{n-1}^2 + O(e_{n-1}^3)]} \\ &= \frac{e_{n-1}e_n f''(x^*)(e_n - e_{n-1})/2 + O(e_{n-1}^4)}{(e_n - e_{n-1})f'(x^*) + (e_n^2 - e_{n+1}^2)f''(x^*)/2 + O(e_{n-1}^3)} \\ &= \frac{e_{n-1}e_n f''(x^*)/2 + O(e_{n-1}^3)}{f'(x^*) + (e_n + e_{n+1})f''(x^*)/2 + O(e_{n-1}^2)} \\ &= \frac{e_{n-1}e_n f''(x^*)/2 + O(e_{n-1}^3)}{f'(x^*) + O(e_n) + O(e_{n-1}^2)} \end{aligned} \tag{1.23}$$

When $n \to \infty$, the lowest order terms in both the numerator and denominator

become the dominant terms as all other higher order terms approach to zero, and we have

$$e_{n+1} = \frac{e_{n-1}e_n f''(x^*)}{2f'(x^*)} = C\, e_n e_{n-1} \tag{1.24}$$

where we have defined a constant $C = f''(x^*)/2f'(x^*)$. To find the order of convergence, we need to find p in

$$|e_{n+1}| \leq |C|\ |e_{n-1}|\ |e_n| = \mu |e_n|^p \tag{1.25}$$

Solving this equation for $|e_n|$ we get

$$|e_n| = \left(\frac{|C|}{\mu}|e_{n-1}|\right)^{1/(p-1)} = \left(\frac{|C|}{\mu}\right)^{1/(p-1)} |e_{n-1}|^{1/(p-1)} \tag{1.26}$$

On the other hand, when $n \to \infty$ we also have

$$|e_n| = \mu |e_{n-1}|^p \tag{1.27}$$

Equating the right-hand sides of the two equations above we get

$$\left(\frac{|C|}{\mu}\right)^{1/(p-1)} |e_{n-1}|^{1/(p-1)} = \mu |e_{n-1}|^p \tag{1.28}$$

which requires the following two equations to hold:

$$p = \frac{1}{p-1}, \qquad \mu = \left(\frac{|C|}{\mu}\right)^{1/(p-1)} = \left(\frac{|C|}{\mu}\right)^{p} \tag{1.29}$$

These two equations can be solved separately to get

$$p = \frac{1+\sqrt{5}}{2} = 1.618, \qquad \mu = |C|^{p/(p+1)} = C^{(\sqrt{5}-1)/2} = |C|^{0.618} \tag{1.30}$$

i.e.,

$$|e_{n+1}| = \mu |e_n|^p = |C|^{0.618} |e_n|^{1.618} = \left|\frac{f''(x)}{2f'(x)}\right|^{0.618} |e_n|^{1.618} \tag{1.31}$$

or

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^{1.618}} = \mu = |C|^{0.618} \tag{1.32}$$

We see that the rate of convergence for the secant method is $\mu = |C|^{0.618}$ and the order of convergence is $p = 1.618$, which is better than the linear convergence of the bisection search (which does not use the information of the specific function $f(x)$), but worse than quadratic convergence of the Newton-Raphson method based on the function derivative $f'(x)$ instead of the approximation $\hat{f}'(x)$, to be considered in the next section.

The Newton-Raphson method hasn't been introduced yet, so mentioning it here is confusing to the reader. Save the comparison of this new method's convergence for after you've introduced it.

The inverse quadratic interpolation

Similar to the secant method that approximates the given function $f(x)$ by a straight line that goes through two consecutive points $\{x_n,\ f(x_n)\}$ and $\{x_{n-1},\ f(x_{n-1})\}$, the inverse quadratic interpolation method approximates the function by a quadratic

curve that goes through three consecutive points $\{x_i,\ f(x_i)),\ \ (i = n,\ n-1,\ n-2)\}$. As the function may be better approximated by a quadratic curve rather than a straight line, the iteration is likely to converge more quickly.

In general, any function $y = f(x)$ can be approximated by a quadratic function $q(x)$ based on three points at $y_0 = f(x_0)$, $y_1 = f(x_1)$, and $y_2 = f(x_2)$. This is the *Lagrange interpolation*:

$$y = q(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}y_0 + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}y_1 + \frac{(x-y_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}y_2 \tag{1.33}$$

"Horizontal function" is an unusual term. If this is meant to be a reference to the horizontal line test for an invertible function, that should be explained in greater detail.

However, here we use the interpolation $x = q^{-1}(y)$ to approximate the inverse (horizontal) function $x = f^{-1}(y)$:

$$x = q^{-1}(y) = \frac{(y-y_1)(y-y_2)}{(y_0-y_1)(y_0-y_2)}x_0 + \frac{(y-y_0)(y-y_2)}{(y_1-y_0)(y_1-y_2)}x_1 + \frac{(y-y_0)(y-y_1)}{(y_2-y_0)(y_2-y_1)}x_2 \tag{1.34}$$

This won't necessarily be obvious to undergraduate readers. Cut "obviously."

Obviously when $y = 0$, the corresponding x is the zero-crossing of $x = q^{-1}(y)$:

$$x_3 = q^{-1}(y)\big|_{y=0} = \frac{y_1y_2}{(y_0-y_1)(y_0-y_2)}x_0 + \frac{y_0y_2}{(y_1-y_0)(y_1-y_2)}x_1 + \frac{y_0y_1}{(y_2-y_0)(y_2-y_1)}x_2 \tag{1.35}$$

which can be used as an estimate of the root x^* of $y = f(x)$. This expression can then be converted into an iteration by which the next root estimate x_{n+1} is computed based on the previous three estimates at x_n, x_{n-1}, and x_{n-2}.

It's clearer to describe the algorithm step-by-step. First determine the initial three root estimates (how?); then estimate the root using the inverse Langrange function on those three values; then subsitute the new root estimate for the oldest of the three points; iterate until convergence.

1.3 Fixed point iteration

We first consider the special case of solving the equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ when $M = N = 1$. To find a solution x^* that satisfies the quation $f(x) = 0$, we can first convert it into an equivalent equation $g(x) = x$, in the sense that an x satisfying one of the equations will also satisfy the other, and then carry out an iteration $x_{n+1} = g(x_n)$ from some initial value x_0. If the iteration converges at a point x^*, i.e., $g(x^*) = x^*$, then we also have $f(x^*) = 0$, i.e., x^* is a solution of the equation $f(x) = 0$. Consider the following examples:

You're defining "equivalent" as it applies to equations, so the word should italicized. The definition would also be clearer as its own sentence: "we can first convert it into an equivalent equation g(x)=x. An equivalent equation is..."

Example 1

To solve the equation

$$f(x) = \log(x) - 0.5 = 0$$

Line breaks before and after examples, and/or boxes around them, will make the text easier to read.

we can first construct another function

$$g(x) = x - f(x) = x - \log(x) + 0.5$$

so that the equation $g(x) = x$ is indeed equivalent to the given equation $f(x) = 0$. We then carry out the iteration $x_{n+1} = g(x_n)$ from some initial value, such as $x_0 = 0.5$, and get:

This would be clearer if phrased in the order in which the actions take place: We set x_0 = 0.5, and then iterate.

n	x_n	$g(x_n)$	$f(x_n)$
0	0.50000	1.69315	−1.19315
1	1.69315	1.66656	0.02659
2	1.66656	1.65580	0.01076
3	1.65580	1.65152	0.00428
4	1.65152	1.64982	0.00169
5	1.64982	1.64915	0.00067
6	1.64916	1.64889	0.00026
7	1.64889	1.64879	0.00010
8	1.64879	1.64875	0.00004
9	1.64875	1.64873	0.00002
10	1.64873	1.64873	0.00001
11	1.64873	1.64872	0.00000
12	1.64872	1.64872	0.00000

This table should be numbered.

We see that the iteration converges to $x^* = \lim_{n\to\infty} g(x_n) = 1.64872$ satisfying $g(x^*) = x^* - f(x^*) = x^*$, and, equivalently, x^* is also the solution of the given equation $f(x) = 0.5 - \log(x) = 0$:

$$0.5 - log(1.64872) = 0, \qquad \text{i.e.,} \qquad e^{0.5} = \sqrt{e} = 1.64872$$

Alternatively, we could construct a different function also equivalent to $f(x) = 0$:

$$g(x) = x + f(x) = x + \log(x) - 0.5$$

But this iteration no longer converges. Why does it not work?

Example 2

The placement of this example makes it seem like it's going to answer the question posed in the previous line. Since it doesn't (and also doesn't provide any other new information) consider cutting this example and going directly to the answer.

$$f(x) = e^x - 1/x = 0$$

We first convert this equation into an equivalent form

$$g(x) = x = e^{-x}$$

and then carry out the iteration:

$$x_{n+1} = g(x_n) = e^{-x_n} \overset{n\to\infty}{\Longrightarrow} x^* = 0.5671$$

which is the root of the given equation $f(x) = e^x - 1/x = 0$, i.e., $e^{-0.5671} = 0.5671$.

Alternatively, the given equation can also be converted into a different form $g(x) = x = -\ln(x)$. However, the iteration based on this function no longer converges.

Why does this iterative method work in some cases but fails in others? To answer these questions we need to understand the theory behind the method, the *fixed point* of a contraction function.

If a single variable function $g(x)$ satisfies

$$|g(x_1) - g(x_2)| \le k|x_1 - x_2|, \qquad k \ge 0 \tag{1.36}$$

it is *Lipschitz continuous*, and k is a *Lipschitz constant*. Specially, if $k = 1$, then $g(x)$ is a *non-expansive* function, if $0 \le k < 1$, then $g(x)$ is a *contraction function* or simply a *contraction*. These concepts can be generated to functions of multivariables $\mathbf{x} = [x_1, \cdots, x_N]^T \in \mathbb{R}^N$ (a point in an N-D metric space V):

$$g_n(\mathbf{x}) = g_n(x_1, \cdots, x_N), \qquad (n = 1, \cdots, N) \tag{1.37}$$

which can also be expressed in vector form:

$$\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \cdots, g_N(\mathbf{x})]^T \tag{1.38}$$

Definition: In a metric space V with certain distance $d(\mathbf{x}, \mathbf{y})$ defined between any two points $\mathbf{x}, \mathbf{y} \in V$, a function $\mathbf{g} : V \to V$ is a *contraction* if

$$d(\mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{y})) \le k\, d(\mathbf{x}, \mathbf{y}) \tag{1.39}$$

The smallest k value that satisfies the above is called the *(best) Lipschitz constant*.

Intuitively, a contraction reduces the distance between points in the space, i.e., it brings them closer together. A function $\mathbf{g}(\mathbf{x})$ may not be a contraction through out its entire domain, but it can be a contraction in the neighborhood of a certain point $\mathbf{x}^* \in V$, in which any $\mathbf{x}$ is sufficiently close to $\mathbf{x}^*$ so that

$$d(\mathbf{g}(\mathbf{x}), \mathbf{g}(\mathbf{x}^*)) \le k\, d(\mathbf{x}, \mathbf{x}^*), \qquad 0 \le k < 1, \quad \text{when } d(\mathbf{x}, \mathbf{x}^*) < \epsilon > 0 \tag{1.40}$$

Definition: A *fixed point* $\mathbf{x}^*$ of a function $\mathbf{g}(\mathbf{x})$ is a point in its domain that is mapped to itself:

$$\mathbf{x}^* = \mathbf{g}(\mathbf{x}^*) \tag{1.41}$$

We immediately have

$$\mathbf{g}(\mathbf{g}(\mathbf{x}^*)) = \mathbf{g}^2(\mathbf{x}^*) = \mathbf{x}^*, \cdots\cdots, \mathbf{g}(\mathbf{g}(\mathbf{g}(\cdots \mathbf{g}(\mathbf{x}^*) \cdots))) = \mathbf{g}^n(\mathbf{x}^*) = \mathbf{x}^* \tag{1.42}$$

A fixed point $\mathbf{x}^*$ is an *attractive fixed point* if any point x in its neighborhood converges to $\mathbf{x}^*$, i.e., $\lim_{n\to\infty} \mathbf{g}^n(\mathbf{x}) = \mathbf{x}^*$.

Fixed Point Theorem : Let $\mathbf{g}(\mathbf{x})$ be a contraction function satisfying

$$d(\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{y})) \le k\, d(\mathbf{x} - \mathbf{y}) \qquad (0 \le k < 1) \tag{1.43}$$

then there exists a unique fixed point $\mathbf{x}^* = \mathbf{g}(\mathbf{x}^*)$, which can be found by an iteration from an arbitrary initial point $\mathbf{x}_0$:

$$\lim_{n\to\infty} \mathbf{x}_n = \lim_{n\to\infty} \mathbf{g}(\mathbf{x}_{n-1}) = \lim_{n\to\infty} \mathbf{g}^2(\mathbf{x}_{n-2}) = \cdots = \lim_{n\to\infty} \mathbf{g}^n(\mathbf{x}_0) = \mathbf{x}^* \tag{1.44}$$

Proof

Cut "specially."

"Multivariables" is an unusual term that is not defined in the text. Substitute "vectors."

The definition of a function on a vector should be placed before the first time functions of vectors are discussed, earlier in the chapter.

"Best" is part of the term and should not be in parentheses.

If this is a definition of "neighborhood," the term should be italicized, and the definition should be rephrased to make that clearer.

This definition would be better placed at the beginning of the chapter, so that readers understand why the method is called "fixed point interation."

What is sigma in this expression?

Overly formal language. Also, undergraduate readers may not immediately make this leap. An explanation of what's happening (the function can be applied to its own output any number of times, and will have the same result) will make things clearer. Since it's novel notation, $g^n(x)$ should also be explicitly defined.

This proof is long, complex, and unnecessary to understand why some fixed-point iterations don't converge. Consider moving to an appendix.

Here the distance $d(\mathbf{x}-\mathbf{y})$ is specifically defined as the *p-norm* (section A.16) of the vector $\mathbf{x}-\mathbf{y}$:

$$d(\mathbf{x},\,\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||_p = \left(\sum_{n=1}^{N}(x_n - y_n)^p\right)^{1/p} \tag{1.45}$$

where $p \geq 1$, e.g., $p = 1, 2, \infty$. Also, for conveninece, we can drop p so that $d(\mathbf{x},\,\mathbf{y}) = ||\mathbf{x}-\mathbf{y}||$.

- We first prove constructively the existence of a fixed point. As $\mathbf{g}(\mathbf{x})$ is a contraction, we have

  $$||\mathbf{x}_{n+1}-\mathbf{x}_n|| = ||\mathbf{g}(\mathbf{x}_n)-\mathbf{g}(\mathbf{x}_{n-1})|| \leq k||\mathbf{x}_n-\mathbf{x}_{n-1}|| \leq k^2||\mathbf{x}_{n-1}-\mathbf{x}_{n-2}|| \leq \cdots \leq k^n||\mathbf{x}_1-\mathbf{x}_0|| \tag{1.46}$$

  As $0 \leq k < 1$, we have $\lim_{n\to\infty} ||\mathbf{x}_{n+1}-\mathbf{x}_n|| = 0$. This is a *Cauchy sequence* (Section A.1) that converges to some point $\lim_{n\to\infty}\mathbf{x}_n = \mathbf{x}^*$ also in the space. We further have

  $$\mathbf{g}(\mathbf{x}^*) = \mathbf{g}(\lim_{n\to\infty}\mathbf{x}_n) = \lim_{n\to\infty}\mathbf{g}(\mathbf{x}_n) = \lim_{n\to\infty}\mathbf{x}_{n+1} = \mathbf{x}^* \tag{1.47}$$

  i.e., the limit of the Cauchy sequence $\lim_{n\to\infty}\mathbf{x}_n = x^*$ is a fixed point.
- We next prove the uniqueness of the fixed point. Let $\mathbf{x}_1^*$ and $\mathbf{x}_2^*$ be two fixed points of $\mathbf{g}(\mathbf{x})$, then we have

  $$||\mathbf{g}(\mathbf{x}_1^*) - \mathbf{g}(\mathbf{x}_2^*)|| \leq k||\mathbf{x}_1^* - \mathbf{x}_2^*|| = k||\mathbf{g}(\mathbf{x}_1^*) - \mathbf{g}(\mathbf{x}_2^*)|| \tag{1.48}$$

  For any $k \neq 0$, the above holds only if $||\mathbf{x}_1^* - \mathbf{x}_2^*|| = ||\mathbf{g}(\mathbf{x}_1^*) - f(\mathbf{x}_2^*)|| = 0$, i.e., $\mathbf{x}_1^* = \mathbf{x}_2^*$ is the unique fixed point.

QED

Theorem: Let $\mathbf{x}^* = \mathbf{g}(\mathbf{x}^*)$ be a fixed point of a differentiable function $\mathbf{g}(\mathbf{x})$, i.e, $\partial g_i/\partial x_j$ exists for any $1 \leq i, j \leq N$. If the norm of the Jacobian matrix is smaller than 1, $||\mathbf{J}_\mathbf{g}(\mathbf{x}^*)|| < 1$, then $\mathbf{g}(\mathbf{x})$ is a contraction at $\mathbf{x}^*$.

The Jacobian matrix of $\mathbf{g}(\mathbf{x})$ is defined as

$$\mathbf{g}'(\mathbf{x}) = \mathbf{J}_\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_N}{\partial x_1} & \cdots & \frac{\partial g_N}{\partial x_N} \end{bmatrix} \tag{1.49}$$

Similarly, this proof is complex and not necessary to understand the point. Explaining the conditions under which g(x) is a contraction and moving the proof to an appendix would be clearer.

Proof:

Consider the Taylor expansion (Section A.22) of the function $\mathbf{g}(\mathbf{x})$ in the neighborhood of $\mathbf{x}^*$:

$$\mathbf{g}(\mathbf{x}) = \mathbf{g}(\mathbf{x}^*) + \mathbf{g}'(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) + R(\mathbf{x}-\mathbf{x}^*) = \mathbf{g}(\mathbf{x}^*) + \mathbf{J}_\mathbf{g}(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) + R(\mathbf{x}-\mathbf{x}^*) \tag{1.50}$$

where $R(\mathbf{x}-\mathbf{x}^*)$ is the remainder composed of second and higher order terms of $\boldsymbol{\delta} = \mathbf{x}-\mathbf{x}^*$. Subtracting $\mathbf{g}(\mathbf{x}^*)$ and taking any p-norm on both sides, we get

$$||\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)||_p = ||\mathbf{J}_\mathbf{g}(\mathbf{x}^*)(\mathbf{x}-\mathbf{x}^*) + R(\mathbf{x}-\mathbf{x}^*)||_p \tag{1.51}$$

When $\mathbf{x} \to \mathbf{x}^*$, the second and higher order terms of $\mathbf{x} - \mathbf{x}^*$ disappear and $R(\mathbf{x} - \mathbf{x}^*) \to 0$, we have

$$||\mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}^*)||_p = ||\mathbf{J}_\mathbf{g}(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)||_p \leq ||\mathbf{J}_\mathbf{g}(\mathbf{x}^*)||_p \cdot ||\mathbf{x} - \mathbf{x}^*||_p, \quad (1.52)$$

The inequality is due to the *Cauchy-Schwarz inequality* (Section A.1) if $||\mathbf{J}_\mathbf{g}(\mathbf{x}^*)|| < 1$, the function $\mathbf{g}(\mathbf{x})$ is a contraction at $\mathbf{x}^*$.

QED

In particular, for a single-variable function in an $N = 1$ dimensional space, we have

$$g(x) - g(x^*) = g'(x^*)(x - x^*) + \frac{1}{2}g''(x^*)(x - x^*)^2 + R(x - x^*) \quad (1.53)$$

and

$$|g(x) - g(x^*)| \leq |g'(x^*)(x - x^*)| \leq |g'(x^*)|\,|(x - x^*)| \quad (1.54)$$

Do you ever prove that g(x) is a contraction at x* if and *only* if |g'(x)| < 1?

If $|g'(x)| < 1$, then $g(x)$ is a contraction at x^*.

Now we understand why in the examples of the previous section the iteration leads to convergence in some cases but divergence in other cases: if $|g'(x)| < 1$, the iteration will converge to the root x^* of $f(x) = 0$, but if $|g'(x)| > 1$, it never will never converge.

If this is referring to an unincluded figure, reference the specific figure number.

The iterative process $x_{n+1} = g(x_n)$ for finding the fixed point of a single-variable function $g(x)$ can be shown graphically as the intersections of the function $y = g(x)$ and the identity function $y = x$, as shown below. The iteration converges in the first two cases as $|g'(x)| < 1$, but it diverges in the last two cases as $|g'(x)| > 1$.

We next find the order of convergence of the fixed point iteration. Consider

Cut "Consider," and move this above expression 1.55.

Rather than going through all of the math, it would be clearer to spend the end of this section explaining the conditions under which fixed-point iteration has various rates of convergence, and why that matters.

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} = \lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|} = \lim_{n\to\infty} \frac{|g(x_n) - g(x^*)|}{|x_n - x^*|} = |g'(x^*)| = \mu < 1 \quad (1.55)$$

We see that in general the fixed point iteration converges linearly. However, if the iteration function $g(x)$ has zero derivative at the fixed point $g'(x^*) = 0$, we have

$$g(x) - g(x^*) = g'(x^*)(x - x^*) + \frac{1}{2}g''(x^*)(x - x^*)^2 + R(x - x^*) = \frac{1}{2}g''(x^*)(x - x^*)^2 + R(x - x^*) \quad (1.56)$$

and the iteration converges quadratically:

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^2} = \lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^2} = \lim_{n\to\infty} \frac{|g(x_n) - g(x^*)|}{|x_n - x^*|^2} = \frac{1}{2}|g''(x^*)| = \text{constant}$$

What is ϕ(x) in this context?

Moreover, if $g''(x^*) = 0$, then the iteration converges cubically.

Consider a specific iteration function in the form of $g(x) = x - f(x)\phi(x)$, which is equivalent to $f(x) = 0$, as if x^* is the root of $f(x)$ satisfying $f(x^*) = 0$, it also satisfies $g(x^*) = x^*$, which is indeed a fixed point of $g(x)$. The derivative of this function is

$$g'(x) = (x - f(x)\phi(x))' = 1 - f'(x)\phi(x) - f(x)\phi'(x) = 1 - f'(x)\phi(x) \quad (1.58)$$

If $f'(x) \neq 0$, we can define $\phi(x) = 1/f'(x)$ so that $g'(x) = 1 - f'(x)/f'(x) = 0$, then the convergence becomes quadratic. This is actually the Newton-Raphson method discussed in the following section.

Save this observation for after the Newton-Raphson method has been introduced.

This section ends with a lot of examples, and doesn't explain clearly what they're illustrating. Is it differing rates of convergence? Circumstances under which the method doesn't converge? Or how to use the method? Choose one or two examples that demonstrate the relevant principle, and explain what they are demonstrating. (If you're trying to demonstrate how to use the method, or the circumstances under which it doesn't converge, those examples would be better placed earlier, when those concepts are introduced.)

Example 1

Find the solution of the following equation:

$$f(x) = x^3 - x - 2 = 0$$

This equation can converted into an equivalent form of $g(x) = x$ in two different ways:

$$g_1(x) = x = \sqrt[3]{x+2}, \qquad \text{and} \qquad g_2(x) = x = x^3 - 2$$

In the figure below, the two functions $g_1(x)$ (left) and $g_2(x)$ (right) together with $f(x)$ and the identity function are plotted:

The iteration based on $g_1(x)$ converges to the solution $x^* = 1.5214$ for any initial guess $-\infty < x_0 < \infty$, as $g_1(x)$ is a contraction. However, the iteration based on $g_2(x)$ does not converge to x^* as it is not a contraction in the neighborhood of x^*. In fact, the iteration will diverge towards either $-\infty$ if $x_0 < x^*$ or ∞ if $x_0 > x^*$.

Example 2

To solve the following equation

$$f(x) = x^3 - 3x + 1 = 0$$

we first convert it into the form of $g(x) = x$ in two different ways:

$$g_1(x) = x = \sqrt[3]{3x-1}, \qquad g_2(x) = x = \frac{1}{3}(x^3 - 1)$$

As can be seen in the plots, this equation has three solutions,

$$x_1 = -1.8794, \qquad x_2 = 0.3473, \qquad x_3 = 1.5321$$

of which x_1 and x_3 can be obtained by the iteration based on $g_1(x)$ and x_2 can be obtained by the iteration based on $g_2(x)$. But neither of them can find all three roots.

- As shown in the plot on the left, $|g_1'(x)| < 1$ for all $-\infty < x < \infty$ except in the neighborhood of $x_2 = 0.3473$, i.e., $g_1(x)$ is a contraction mapping everywhere except around x_2. Therefore the iteration starting from any initial guess x_0 will converge to either $x_1 = -1.8749$ if $x_0 < x_2$, or $x_3 = 1.5321$ if $x_0 > x_2$.

$$x_{n+1} = g_1(x_n) \overset{n\to\infty}{\Longrightarrow} \begin{cases} x_1 = -1.8749 & x_0 < x_2 \\ x_3 = 1.5321 & x_0 > x_2 \end{cases}$$

  However, as $g_1(x)$ is not a contraction mapping around $x = x_2$, the iteration will never converge to x_2.

- As shown in the plot on the right, $|g_2'(x)| > 1$ for all $-\infty < x < \infty$ except in the neighborhood of x_2, i.e., $g_2(x)$ is not a contraction mapping around either x_1 or x_3. Therefore the iteration based on $g_2(x)$ will not converge to either x_1 or x_3, but it may converge to x_2, if the initial guess x_0 is in the range $x_1 < x_0 < x_3$. However, if x_0 is outside this range the iteration will diverge toward either $-\infty$ if $x_0 < x_1$ of ∞ if $x_0 > x_3$.

$$x_{n+1} = g_2(x_n) \overset{n\to\infty}{\Longrightarrow} \begin{cases} -\infty & x_0 < x_1 \\ x_2 = 0.3473 & x_1 < x_0 < x_3 \\ \infty & x_0 > x_3 \end{cases}$$

n	x_n	$f(x_n)$
1	$-2.0408275e+00$	$-1.3775173e+00$
2	$-1.9240239e+00$	$-3.5041089e-01$
3	$-1.8919392e+00$	$-9.6254050e-02$
4	$-1.8829328e+00$	$-2.7019276e-02$
5	$-1.8803890e+00$	$-7.6311660e-03$
6	$-1.8796694e+00$	$-2.1590449e-03$
7	$-1.8794656e+00$	$-6.1114717e-04$
8	$-1.8794080e+00$	$-1.7301761e-04$
9	$-1.8793916e+00$	$-4.8983741e-05$
10	$-1.8793870e+00$	$-1.3868146e-05$
11	$-1.8793857e+00$	$-3.9263250e-06$
12	$-1.8793853e+00$	$-1.1116151e-06$

Example 3

$$f(x) = \sin(x) - x^3 = 0$$

This equation can be converted into the form $g(x) = x$ in different ways:

- $$g_0(x) = x = \sqrt[3]{\sin(x)}, \quad g_0'(x) = \frac{\cos(x)}{3\,\sin(x)^{2/3}}$$

n	x_n	$f(x_n)$
0	$2.000000e+00$	-7.0907
1	$9.688027e-01$	-0.085089
2	$9.375886e-01$	-0.018075
3	$9.306842e-01$	-0.0041048
4	$9.291018e-01$	-0.00094611
5	$9.287364e-01$	-0.00021881
6	$9.286518e-01$	$-5.0647e-05$
7	$9.286322e-01$	$-1.1725e-05$
8	$9.286277e-01$	$-2.7144e-06$

- $$g_1(x) = x = \sin(x)/x^2, \quad g_1'(x) = \frac{\cos(x)}{x^2} - \frac{2\,\sin(x)}{x^3}$$

- $$g_2(x) = x + \sin(x) - x^3, \quad g_2'(x) = \cos(x) - 3\,x^2 + 1$$

- $$g_3(x) = x - \frac{\sin(x) - x^3}{\cos(x) - 3x^2}, \quad g_3'(x) = -\frac{(6x + \sin(x))(\sin(x) - x^3)}{(\cos(x) - 3x^2)^2}$$

n	x_n	$f(x_n)$
0	$2.000000e+00$	-7.0907
1	$1.428913e+00$	-1.9276
2	$1.106787e+00$	-0.46152
3	$9.637850e-01$	-0.073886
4	$9.304466e-01$	-0.0036294
5	$9.286316e-01$	$-1.0509e-05$
6	$9.286263e-01$	$-8.9029e-11$

Example 4

$$f(x) = e^x - \frac{1}{x}$$

- $$g_0(x) = x = e^{-x}, \quad g_0'(x) = e^{-x}$$

  The iteration from any initial guess $x_0 > 0$ will converge to $x^* = g_1(x^*) = 0.56714$.

- $$g_1(x) = x = -\log(x), \quad g_1'(x) = 1/x$$

  Around $x = x^* = 0.56714$, $g_1'(x) = 1/x > 1$, the iteration does not converge.

- $$g_2(x) = x - \frac{e^x - 1/x}{e^x + 1/x^2}, \quad g_2'(x) = \frac{(e^x - 1/x)(e^x - 2/x^3)}{(e^x + 1/x^2)^2}$$

Example 5
Consider a 3-variable linear vector function $\mathbf{f}(\mathbf{x})$ of arguments $\mathbf{x} = [x,\ y,\ z]^T$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \begin{cases} f_1(\mathbf{x}) = 6x + 3y + 2z - 18 = 0 \\ f_2(\mathbf{x}) = 2x + 7y + 3z - 25 = 0 \\ f_3(\mathbf{x}) = x + 3y + 5z - 22 = 0 \end{cases}$$

from which the g-function can be obtained:

$$\mathbf{g}(\mathbf{x}) = \mathbf{x}, \qquad \begin{cases} g_1(\mathbf{x}) = x = -(3y + 2z - 18)/6 \\ g_2(\mathbf{x}) = y = -(2x + 3z - 25)/7 \\ g_3(\mathbf{x}) = z = -(x + 3y - 22)/5 \end{cases}$$

The Jacobian $\mathbf{g}(\mathbf{x})$ of this linear system is a constant matrix

$$\mathbf{J}_g = \begin{bmatrix} 0 & -1/2 & -1/3 \\ -2/7 & 0 & -3/7 \\ -1/5 & -3/5 & 0 \end{bmatrix}$$

with the induced p=2 norm (maximum singular value) $||\mathbf{J}_g|| = 0.851 < 1$. Consequently, the iteration $\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n)$ converges from an initial guess $\mathbf{x}_0 = [1,\ 1,\ 1]^T$ to the solution $\mathbf{x} = [1,\ 2,\ 3]^T$.

Alternatively, the g-function can also be obtained as

$$\mathbf{g}'(\mathbf{x}) = \mathbf{x}, \qquad \begin{cases} g_1'(\mathbf{x}) = x = -(3y + 5z - 22) \\ g_2'(\mathbf{x}) = y = -(2x + 3z - 25)/7 \\ g_3'(\mathbf{x}) = z = -(6x + 3y - 18)/2 \end{cases}$$

The Jacobian is

$$\mathbf{J}_{g'} = \begin{bmatrix} 0 & -3 & -5 \\ -2/7 & 0 & -3/7 \\ -3 & -3/2 & 0 \end{bmatrix}$$

with the induced p=2 norm $||\mathbf{J}_g|| = 5.917 > 1$. The iteration does not converge.

Example 6

Consider a 3-variable nonlinear function $\mathbf{f}(\mathbf{x})$ of arguments $\mathbf{x} = [x,\ y,\ z]^T$:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \begin{cases} f_1(\mathbf{x}) = x^2 - 2x + y^2 - z + 1 = 0 \\ f_2(\mathbf{x}) = xy^2 - x - 3y + yz + 2 = 0 \\ f_3(\mathbf{x}) = xz^2 - 3z + yz^2 + xy = 0 \end{cases}$$

The g-function can be obtained as

$$\mathbf{g}(\mathbf{x}) = \mathbf{x}, \qquad \begin{cases} g_1(\mathbf{x}) = x = (x^2 + y^2 - z + 1)/2 \\ g_2(\mathbf{x}) = y = (xy^2 - x + yz + 2)/3 \\ g_3(\mathbf{x}) = z = (xz^2 + yz^2 + xy)/3 \end{cases}$$

With $\mathbf{x}_0 = [0,\ 0,\ 0]^T$ and after $n > 170$ iterations $\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n)$ converges to $\mathbf{x}_n = [1.098933,\ 0.367621,\ 0.144932]^T$, with error $||\mathbf{f}(\mathbf{x}_n)|| < 10^{-7}$. However, the iteration may not converge from other possible initial guesses.

Aitken's method is a significant new concept that should have its own headed subsection. As it is currently worded, it reads as though this is part of Example 6.

By Aitken's method, the iteration $x_{n+1} = g(x_n)$ can be accelerated based on two consecutive points x_0 and x_1, as shown in the figure below.

The secant line of $g(x)$ that goes through the two points $P_0 = (x_0, g(x_0) = x_1)$ and $P_1 = (x_1, g(x_1) = x_2)$ is represented by the equation in terms of its slope:

$$\frac{y - g(x_0)}{x - x_0} = \frac{g(x_1) - g(x_0)}{x_1 - x_0} \tag{1.59}$$

The concept of accelerating iteration should be explicitly defined.

Solving for y, we get

$$y = g(x_0) + (x - x_0)\frac{g(x_1) - g(x_0)}{x_1 - x_0} = x_1 + (x - x_0)\frac{x_2 - x_1}{x_1 - x_0} \tag{1.60}$$

Unclear. What do you mean by "moving from x_0 to x_1"?

To accelerate, instead of moving from x_0 to x_1, we move to the point x at which this secant line intersects with the identity function $y = x$. We can therefore replace y in the equation above by x and solve the resulting equation

$$x = x_1 + (x - x_0)\frac{x_2 - x_1}{x_1 - x_0} \tag{1.61}$$

to get

$$x = \frac{x_1(x_1 - x_0) - x_0(x_2 - x_1)}{(x_1 - x_0) - (x_2 - x_1)} = x_0 - \frac{(x_1 - x_0)^2}{x_2 - 2x_1 + x_0} = x_0 - \frac{(\Delta x_0)^2}{\Delta^2 x_0} \tag{1.62}$$

where Δx_0 and $\Delta^2 x_0$ are respectively the first and second order differences defined below:

$$\Delta x_0 = x_1 - x_0, \qquad \Delta x_1 = x_2 - x_1 \tag{1.63}$$

$$\Delta^2 x_0 = \Delta x_1 - \Delta x_0 = (x_2 - x_1) - (x_1 - x_0) = x_2 - 2x_1 + x_0 \tag{1.64}$$

This result can then be converted into an iterative process

How does this accelerate convergence, if x_n+1 and x_n+2 still have to be computed each time? This section needs more explanation.

$$\begin{cases} x_{n+1} = g(x_n) \\ x_{n+2} = g(x_{n+1}) \\ x_{n+3} = x_n - (x_{n+1} - x_n)^2/(x_{n+2} - 2x_{n+1} + x_n) \end{cases} \tag{1.65}$$

Given x_n, we skip $x_{n+1} = g(x_n)$ and $x_{n+2} = g(x_{n+1})$ but directly move to x_{n+3} computed based on x_{n+1} and x_{n+2}, thereby making a greater step towards the solution.

This example would benefit from actually working through a couple of iterations of each method.

Example

Solve $x^3 - 3x + 1 = 0$. Construct $g(x) = (3x-1)^{1/3}$. It takes 18 iterations for the regular fixed point algorithm with initial guess $x_0 = 1$, to get $x_{18} = 1.5320887$ that satisfies $|f(x_n)| < 10^{-6}$, but it only three iterations for Aitken's method to converge to the same result:

n	x_n	$f(x_n)$
0	1.0000	-1.0000
1	1.259921	$-7.797631e-01$
2	1.406056	$-4.384047e-01$
3	1.476396	$-2.110206e-01$
4	1.507985	$-9.476760e-02$
5	1.521751	$-4.129594e-02$
6	1.527672	$1.776362e-02$
7	1.530205	$-7.598914e-03$
8	1.531286	$3.242990e-03$
9	1.531747	$-1.382619e-03$
10	1.531943	$-5.892137e-04$
11	1.532027	$-2.510521e-04$
12	1.532062	$-1.069599e-04$
13	1.532078	$-4.556840e-05$
14	1.532084	$-1.941335e-05$
15	1.532087	$-8.270551e-06$
16	1.532088	$-3.523444e-06$
17	1.532089	$-1.501066e-06$
18	1.532089	$-6.394874e-07$

n	x_n	$f(x_n)$
0	1.0000	-1.0000
1	1.5937361	$2.668730e-01$
2	1.5323992	$1.254749e-03$
3	1.5320889	$3.421160e-08$

1.4 Newton-Raphson Method (Univariate)

To solve equation $f(x) = 0$, we first consider the Taylor series expansion of $f(x)$ at any point x_0:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!}f''(x)(x - x_0)^2 + \cdots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n + \cdots \tag{1.66}$$

If $f(x)$ is linear, i.e., its slope $f'(x)$ is a constant for any x, then the second and higher order terms are all zero, and the equation $f(x) = 0$ becomes

$$f(x) = f(x_0) + f'(x)(x - x_0) = 0 \tag{1.67}$$

Solving this equation we get the root x^* at which $f(x^*) = 0$:

$$x^* = x = x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 + \Delta x_0 \tag{1.68}$$

n	x_n	$f(x_n)$
0	1.0000	-1.0000
1	1.259921	$-7.797631e-01$
2	1.406056	$-4.384047e-01$
3	1.476396	$-2.110206e-01$
4	1.507985	$-9.476760e-02$
5	1.521751	$-4.129594e-02$
6	1.527672	$1.776362e-02$
7	1.530205	$-7.598914e-03$
8	1.531286	$3.242990e-03$
9	1.531747	$-1.382619e-03$
10	1.531943	$-5.892137e-04$
11	1.532027	$-2.510521e-04$
12	1.532062	$-1.069599e-04$
13	1.532078	$-4.556840e-05$
14	1.532084	$-1.941335e-05$
15	1.532087	$-8.270551e-06$
16	1.532088	$-3.523444e-06$
17	1.532089	$-1.501066e-06$
18	1.532089	$-6.394874e-07$

n	x_n	$f(x_n)$
0	1.0000	-1.0000
1	1.5937361	$2.668730e-01$
2	1.5323992	$1.254749e-03$
3	1.5320889	$3.421160e-08$

1.4 Newton-Raphson Method (Univariate)

An introduction explaining what the Newton-Raphson method is would make this section a lot easier to understand.

To solve equation $f(x) = 0$, we first consider the Taylor series expansion of $f(x)$ at any point x_0:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2!}f''(x)(x - x_0)^2 + \cdots + \frac{1}{n!}f^{(n)}(x_0)(x - x_0)^n + \cdots \tag{1.66}$$

If $f(x)$ is linear, i.e., its slope $f'(x)$ is a constant for any x, then the second and higher order terms are all zero, and the equation $f(x) = 0$ becomes

This will be clearer if first the revised Taylor series expansion is given, and then the equation is set to 0.

$$f(x) = f(x_0) + f'(x)(x - x_0) = 0 \tag{1.67}$$

Solving this equation we get the root x^* at which $f(x^*) = 0$:

$$x^* = x = x_0 - \frac{f(x_0)}{f'(x_0)} = x_0 + \Delta x_0 \tag{1.68}$$

where $\Delta x_0 = -f(x_0)/f'(x_0)$ is the step we need to take to go from the initial point x_0 to the root x^*:

$$\Delta x_0 = -\frac{f(x_0)}{f'(x_0)} \begin{cases} < 0 & \text{if } f(x_0) \text{ and } f'(x_0) \text{ are of the same sign} \\ > 0 & \text{if } f(x_0) \text{ and } f'(x_0) \text{ are of different signs} \end{cases} \tag{1.69}$$

If $f(x)$ is nonlinear, the sum of the first two terms of the Taylor expansion is only an approximation of $f(x)$, and the resulting x found above can be treated as an approximation of the root, which can be improved iteratively to move from the initial point x_0 towards the root, as illustrated in the figure below:

$$x_{n+1} = x_n + \Delta x_n = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad n = 0,\ 1,\ 2, \cdots \tag{1.70}$$

The Newton-Raphson method can be considered as the fixed point iteration $x_{n+1} = g(x_n)$ based on

$$g(x) = x - f(x)/f'(x) \tag{1.71}$$

The root x^* at which $f(x^*) = 0$ is also the fixed point of $g(x)$, i.e., $g(x^*) = x^*$. For the iteration to converge, $g(x)$ needs to be a contraction with $|g'(x)| < 1$. Consider

$$g'(x) = \left(x - \frac{f(x)}{f'(x)}\right)' = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2} \tag{1.72}$$

We make the following comments:

- At the root $x = x^*$ where $f(x^*) = 0$, if $f'(x^*) \neq 0$, then $g'(x^*) = 0 < 1$, i.e., $g(x)$ is a contraction and the iteration $x_{n+1} = g(x_n)$ converges quadratically when x_n is close to x^*.
- If $f'(x_n) = 0$ (the tangent is horizontal), the iteration cannot proceed, but we can modify x_n by adding a small value ϵ to x_n so that $f'(x + \epsilon) \neq 0$.
- It is difficult to know the number of roots of a nonlinear equation (unlike the case of a linear equation), which can vary from zero (e.g., $f(x) = x^2 + 1$) to infinity (e.g., $f(x) = \cos(x)$). One can try different initial guesses in the range of interest to see if different roots can be found.
- Sometime a parameter δ can be used to control the step size of the iteration:

$$x_{n+1} = x_n - \delta \frac{f(x_n)}{f'(x_n)} \tag{1.73}$$

  - If $\delta < 1$, the iteration is de-accelerated. Although the convergence becomes slower, this may be desirable if the function $f(x)$ is not smooth with many local variations.
  - If $\delta > 1$, the iteration is accelerated. The convergence may or may not be accelerated. Due to the greater step size, the root may be skipped and missed. Sometimes the convergence may become significantly slowed or even oscillate around the true root, such as the example shown in the figure below with $\delta = 2$.

I think that what you mean by this is that the Newton-Raphson method works iteratively in the same way that the previous two methods do, with the new point at each iteration chosen by the expression given above. But that is not clear. Give an algorithmic explanation of how the method works.

Overly formal language.

These bullet points would be better organized into paragraphs that explain each point and why it's important.

The initial explanation of what the Newton-Raphson method is used a specifically linear equation. If it's going to be applied to non-linear cases, it needs to be addressed in detail how that does or does not change the equation and the algorithm.

Acceleration of iteration should be defined, especially with regards to how it is distinct from acceleration of convergence.

This is a long and complicated derivation that is unnecessary to understand the Newton-Raphson method. Consider moving it to an appendix and just telling the reader what the order of convergence is.

The order of convergence of the Newton-Raphson iteration can be found based on the Taylor expansion of $f(x)$ at the neighborhood of the root $x^* = x_n + e_n$:

$$0 = f(x^*) = f(x_n) + f'(x_n)e_n + \frac{f''(x_n)}{2}e_n^2 + O(e_n^3) \tag{1.74}$$

What is $O(e^3_n)$ in this context?

where $e_n = x^* - x_n$ is the error at the nth step. Substituting the Newton-Raphson's iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad \text{i.e.} \quad f(x_n) = f'(x_n)(x_n - x_{n+1}) \tag{1.75}$$

into the equation above, we get

$$\begin{aligned} 0 &= f'(x_n)(x_n - x_{n+1}) + f'(x_n)(x^* - x_n) + \frac{f''(x_n)}{2}e_n^2 + O(e_n^3) \\ &= f'(x_n)(x^* - x_{n+1}) + \frac{f''(x)}{2}e_n^2 + O(e_n^3) \\ &= f'(x_n)e_{n+1} + \frac{f''(x_n)}{2}e_n^2 + O(e_n^3) \end{aligned} \tag{1.76}$$

i.e.

$$e_{n+1} = -\frac{f''(x_n)}{2f'(x_n)}e_n^2 + O(e_n^3) \tag{1.77}$$

When $n \to \infty$ all the higher order terms disappear, and the above can be written as

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^2} \le \frac{|f''(x^*)|}{2|f'(x^*)|} = \mu \tag{1.78}$$

Alternatively, we can get the Taylor expansion in terms of $g(x)$:

$$x_{n+1} = x^* - e_{n+1} = g(x_n) = g(x^* - e_n) = g(x^*) - g'(x^*)e_n + \frac{g''(x^*)}{2}e_n^2 + O(e_n^3) \tag{1.79}$$

Subtracting $g(x^*) = x^*$ from both sides we get:

$$e_{n+1} = g'(x^*)e_n - \frac{g''(x^*)}{2}e_n^2 + O(e_n^3) \tag{1.80}$$

Now we find $g'(x)$ and $g''(x)$:

$$g'(x) = \left(x - \frac{f(x)}{f'(x)}\right)' = 1 - \frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = \frac{f(x)f''(x)}{(f'(x))^2} \tag{1.81}$$

and

$$\begin{aligned} g''(x) &= \left(\frac{f(x)f''(x)}{(f'(x))^2}\right)' = \frac{(f'(x)f''(x) + f(x)f'''(x))(f'(x))^2 - f(x)f''(x)2f'(x)f''(x)}{(f'(x))^4} \\ &= \frac{(f'(x))^3 f''(x)}{(f'(x))^4} = \frac{f''(x)}{f'(x)} \end{aligned} \tag{1.82}$$

Evaluating these at $x = x^*$ at which $f(x^*) = 0$, and substituting them back

into the expression for e_{n+1} above, the linear term is zero as $g'(x^*) = 0$, i.e., the convergence is quadratic, and we get the same result:

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^2} \leq \frac{|f''(x^*)|}{2|f'(x^*)|} = \frac{|g''(x^*)|}{2} \tag{1.83}$$

We see that, if $f'(x^*) \neq 0$, then the order of convergence of the Newton-Raphson method is $p = 2$, and the rate of convergence is $\mu = |f''(x)|/2|f'(x)|$. However, if $f'(x^*) = 0$, the convergence is linear rather than quadratic, as shown in the example below.

Example: Consider solving the equation

$$f(x) = x^3 - 4x^2 + 5x - 2 = (x-1)^2(x-2) = 0 \tag{1.84}$$

which has a repeated root $x = 1$ as well as a single root $x = 2$. We have

$$f'(x) = 3x^2 - 8x + 5 = (3x-5)(x-1), \qquad f''(x) = 6x - 8 \tag{1.85}$$

Note that at the root $x^* = 1$ we have $f'(x^*) = f'(1) = 0$. We further find:

$$g(x) = x - \frac{f(x)}{f'(x)} = x - \frac{(x-1)^2(x-2)}{(3x-5)(x-1)} = x - \frac{(x-1)(x-2)}{3x-5} \tag{1.86}$$

and

$$\begin{aligned} g'(x) &= \frac{f(x)f''(x)}{(f'(x))^2} = \frac{(x-1)^2(x-2)(6x-8)}{(3x-5)^2(x-1)^2} \\ &= \frac{(x-2)(6x-8)}{(3x-5)^2} = \begin{cases} 1/2 & x = 1 \\ 0 & x = 2 \end{cases} \end{aligned}$$

We therefore have

$$e_{n+1} = g'(x)e_n - \frac{1}{2}g''(x)e_n^2 + O(e_n^3) \overset{n\to\infty}{\Longrightarrow} \begin{cases} e_n/2 & x = 1 \\ -g''(x)e_n^2/2 & x = 2 \end{cases} \tag{1.87}$$

We see the iteration converges quadratically to the single root $x = 2$, but only linearly to the repeated root $x = 1$.

We consider in general a function with a repeated root at $x = a$ of multiplicity k:

$$f(x) = (x-a)^k h(x) \tag{1.88}$$

its derivative is

$$f'(x) = k(x-a)^{k-1}h(x) + (x-a)^k h'(x) = (x-a)^{k-1}[kh(x) + (x-a)h'(x)] \tag{1.89}$$

As $f'(a) = 0$, the convergence of the Newton-Raphson method to $x = a$ is linear, rather than quadratic.

In such case, we can accelerate the iteration by using a step size $\delta = k > 1$:

$$g(x) = x - k\,\frac{f(x)}{f'(x)} \tag{1.90}$$

and

$$g'(x) = 1 - k\,\frac{(f'(x))^2 - f(x)f''(x)}{(f'(x))^2} = 1 - k + k\,\frac{f(x)f''(x)}{(f'(x))^2} \tag{1.91}$$

Now we show that this $g'(x)$ is zero at the repeated root $x = a$, therefore the convergence to this root is still quadratic.

We substitute $f(x) = (x-a)^k h(x)$, $f'(x) = k(x-a)^{k-1}[kh + (x-a)h']$, and

$$\begin{aligned} f''(x) &= [(x-a)^{k-1}[kh + (x-a)h']]' \\ &= 1 - k + k\,(k-1)(x-a)^{k-2}[kh + (x-a)h'] \\ &\quad +(x-a)^{k-1}[(k+1)h' + (x-a)h''] \end{aligned} \tag{1.92}$$

into the expression for $g'(x)$ above to get (after some algebra):

$$\begin{aligned} g'(x) &= 1 - k + k\frac{f(x)f''(x)}{(f'(x))^2} \\ &= 1 - k + k\,\frac{h(k-1)(kh + (x-a)h') + h(x-a)((k+1)h' + (x-a)h')}{(kh + (x-a)h')^2} \end{aligned} \tag{1.93}$$

At $x = a$, we get

$$g'(x)\Big|_{x=a} = 1 - k + k\frac{(k-1)kh^2}{(kh)^2} = 0 \tag{1.94}$$

i.e., the convergence to the repeated root at $x = a$ is no longer linear but quadratic.

The difficulty, however, is that the multiplicity k of a root is unknown ahead of time. If $\delta > 1$ is used blindly some root may be skipped, and the iteration may oscillate around the real root.

Example: Consider solving $f(x) = x^3 - 4x^2 + 5x - 2 = (x-1)^2(x-2) = 0$, with a double root $x = 1$ and a single root $x = 2$. In the following, we compare the performance of both $g_1(x) = x - f(x)/f'(x)$ and $g_2(x) = x - 2f(x)/f'(x)$.

- First use an initial guess $x_0 = 0.3$.
  - When $g_1(x)$ is used, the convergence is linear around the double root $x = 1$. It takes 16 iterations to get $x_{16} = 0.999983$ with $|e_{16}| < 10^{-9}$:

n	x_n	$f(x_n)$	$f'(x_n)$
0	0.300000	−0.833000	2.870000
1	0.590244	−0.236698	1.323212
2	0.769125	−0.065609	0.621659
3	0.874665	−0.017678	0.297797
4	0.934027	−0.004640	0.145004
5	0.966023	−0.001194	0.071417
6	0.982737	−0.000303	0.035420
7	0.991296	−0.000076	0.017636
8	0.995629	−0.000019	0.008799
9	0.997810	−0.000005	0.004395
10	0.998904	−0.000001	0.002196
11	0.999452	−0.000000	0.001098
12	0.999726	−0.000000	0.000549
13	0.999863	−0.000000	0.000274
14	0.999931	−0.000000	0.000137
15	0.999966	−0.000000	0.000069
16	0.999983	−0.000000	

  - When $g_2(x)$ is used, the convergence is quadratic around the double root $x = 1$. It takes only 3 iterations to get $x_3 = 0.999982$ with $|e_3| < 10^{-9}$:

n	x_n	$f(x_n)$	$f'(x_n)$
0	0.300000	−0.833000	2.870000
1	0.880488	−0.015990	0.281874
2	0.993944	−0.000037	0.012222
3	0.999982	−0.000000	

- Next use a different initial guess $x_0 = 4$.
  - If g_1 is used, it takes 7 iterations to get $x_7 = 2.0$ with $|e_7| < 10^{-9}$, the convergence is quadratic.

n	x_n	$f(x_n)$	$f'(x_n)$
1	$x = 3.00000$	4.00000	8.00000
2	$x = 2.50000$	1.12500	3.75000
3	$x = 2.20000$	0.28800	1.92000
4	$x = 2.05000$	0.05513	1.20750
5	$x = 2.00435$	0.00439	1.01745
6	$x = 2.00004$	0.00004	1.00015
7	$x = 2.00000$	0.00000	1.00000
8	$x = 2.00000$	0.00000	

– If g_2 is used, oscillation happens as shown in the figure below. However, if a better initial guess $x_0 = 3.5$ is used instead of $x_0 = 4$, it takes only one step to get $x_1 = 2.0$ with $|e_1| < 10^{-9}$, the convergence is significantly accelerated.

n	x_n	$f(x_n)$	$f'(x_n)$
1	3.00000	4.00000	8.00000
2	2.00000	0.00000	

1.5 Newton-Raphson Method (Multivariate)

The Newton-Raphson method discussed above for solving a single-variable equation $f(x) = 0$ can be generalized to the case of multivariate equation systems containing M equations of N variables in $\mathbf{x} = [x_1, \cdots, x_N]^T$:

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} f_1(x_1, \cdots, x_N) \\ \vdots \\ f_M(x_1, \cdots, x_N) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \mathbf{0} \tag{1.95}$$

To solve the equation system, we first consider the Taylor series expansion of each of the M functions in the neighborhood of the initial point $\mathbf{x}_0 = [x_{01}, \cdots, x_{0N}]^T$:

$$f_m(\mathbf{x}) = f_m(\mathbf{x}_0) + \sum_{n=1}^{N} \frac{\partial f_m(\mathbf{x}_0)}{\partial x_n}(x_n - x_{0n}) + r_m(||\mathbf{x} - \mathbf{x}_0||^2), \qquad (m = 1, \cdots, M) \tag{1.96}$$

where $r_m(||\mathbf{x} - \mathbf{x}_0||^2)$ represents the second and higher order terms in the series beyond the linear term, which can be neglected if $||\mathbf{x} - \mathbf{x}_0||$ is small. These M equations can be expressed in matrix form

$$\begin{bmatrix} f_1(\mathbf{x}) \\ \vdots \\ f_M(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} f_1(\mathbf{x}_0) \\ \vdots \\ f_M(\mathbf{x}_0) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1(\mathbf{x}_0)}{\partial x_1} & \cdots & \frac{\partial f_1(\mathbf{x}_0)}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_M(\mathbf{x}_0)}{\partial x_1} & \cdots & \frac{\partial f_M(\mathbf{x}_0)}{\partial x_N} \end{bmatrix} \begin{bmatrix} x_1 - x_{01} \\ \vdots \\ x_N - x_{0N} \end{bmatrix} + \begin{bmatrix} r_1 \\ \vdots \\ r_M \end{bmatrix} \tag{1.97}$$

or more concisely

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\,(\mathbf{x} - \mathbf{x}_0) + \mathbf{r} \approx \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\,(\mathbf{x} - \mathbf{x}_0) = \mathbf{f}_0 + \mathbf{J}_0\,\Delta\mathbf{x} \tag{1.98}$$

where $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_0$, and $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0)$ and $\mathbf{J}_0 = \mathbf{J}(\mathbf{x}_0)$ are respectively the function $\mathbf{f}(\mathbf{x})$ and its *Jacobian matrix* $\mathbf{J}_f(\mathbf{x})$ both evaluated at $\mathbf{x}_0$. We further consider solving the equation system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in the following two cases:

These bullet points are very long. They might be better organized as subsections of some kind.

- $M = N$: The number of equations is the same as the number of unknowns, the Jacobian $\mathbf{J}(\mathbf{x})$ is a square matrix and its inverse $\mathbf{J}^{-1}$ exists in general. In the special case where $\mathbf{f}(\mathbf{x})$ is linear, the Taylor series contains only the first two terms while all higher order terms are zero, and the approximation

in Eq. (1.98) becomes exact. To find the root $\mathbf{x}^*$ satisfying $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, we set $\mathbf{f}(\mathbf{x})$ in Eq. (1.98) to zero and solve the resulting equation for $\mathbf{x}$ to get

$$\mathbf{x}^* = \mathbf{x} = \mathbf{x}_0 - \mathbf{J}_0^{-1}\mathbf{f}_0 = \mathbf{x}_0 - \mathbf{A}^{-1}\mathbf{f}_0 \tag{1.99}$$

As in general $\mathbf{f}(\mathbf{x})$ is nonlinear, it can only be approximated by the first two terms of the Taylor series, consequently the result above is only an approximation of the optimal solution. But this approximation can be improved iteratiively to approach the optimal solution $\mathbf{x}^*$:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n = \mathbf{x}_n - \mathbf{J}_n^{-1}\,\mathbf{f}_n \tag{1.100}$$

where we have defined $\Delta\mathbf{x}_n = -\mathbf{J}_n^{-1}\,\mathbf{f}_n$ as the increment, which can also be denoted by $\mathbf{d}_n = \Delta\mathbf{x}_n$ to represent the *search* or *Newton direction*. The iteration moves $\mathbf{x}_n$ in the N-D space spanned by $\{x_1, \cdots, x_N\}$ from some initial guess $\mathbf{x}_0$ along such a path that all function values $f_m(\mathbf{x}), \quad m = 1, \cdots, M)$ are reduced. Same as in the univariate case, a scaling factor δ_n can be used to control the step size of the iteration

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \delta_n\,\Delta\mathbf{x}_n = \mathbf{x}_n - \delta_n\,\mathbf{J}_n^{-1}\,\mathbf{f}_n \tag{1.101}$$

When $\delta < 1$, the step size becomes smaller and the convergence of the iteration is slower, however, we will have a better chance not to skip a solution, which may happen if $\mathbf{f}(\mathbf{x})$ is not smooth and the step size is too big.

Smoothness of functions should be defined.

The algorithm is listed below:

- Select $\mathbf{x}_0$
- Obtain $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0)$ and $\mathbf{J}_0$
- Obtain $\mathbf{J}_0^{-1}$
- $n = 0$
- While $||\mathbf{x}_{n+1} - \mathbf{x}_n|| > tol$ do
  - $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_n^{-1}\mathbf{f}_n$
  - Find $\mathbf{f}_{n+1}$ and $\mathbf{J}_{n+1}$
  - $n = n + 1$

Having a step by step description of the algorithm is very helpful! This, or something like it, should exist for every method and method variation that you describe.

tol should be defined somewhere as meaning "tolerance."

- $M > N$: There are more equations than unknowns, i.e., equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ is an over-constrained system, and the Jacobian $\mathbf{J}(\mathbf{x})$ is an $M \times N$ non-square matrix without an inverse, i.e., no solution exists for the equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ in general. But we can still seek to find an optimal solution $\mathbf{x}^*$ that minimizes the following sum-of-squares error:

$$\varepsilon(\mathbf{x}) = \frac{1}{2}||\mathbf{f}(\mathbf{x})||^2 = \frac{1}{2}\mathbf{f}(\mathbf{x})^T\mathbf{f}(\mathbf{x}) = \frac{1}{2}\sum_{m=1}^{M} f_m^2(\mathbf{x}) \tag{1.102}$$

Gradient is a key term, and should be italicized.

The gradient vector of $\varepsilon(\mathbf{x})$ is:

$$\mathbf{g}_\varepsilon(\mathbf{x}) = \frac{d}{d\mathbf{x}}\varepsilon(\mathbf{x}) = \frac{d}{d\mathbf{x}}\left(\frac{1}{2}\sum_{m=1}^{M} f_m^2(\mathbf{x})\right) = \sum_{m=1}^{M}\frac{d}{d\mathbf{x}}f_m(\mathbf{x})\;f_m(\mathbf{x}) \tag{1.103}$$

The nth component of $\mathbf{g}_\varepsilon(\mathbf{x})$ is

$$\frac{\partial \varepsilon(\mathbf{x})}{\partial x_n} = \sum_{m=1}^{M} \frac{\partial f_m(\mathbf{x})}{\partial x_n} f_m(\mathbf{x}) = \sum_{m=1}^{M} J_{mn} f_m(\mathbf{x}) \quad (n = 1, \cdots, N) \tag{1.104}$$

where $J_{mn} = \partial f_m(\mathbf{x})/\partial x_n$ is the component in the mth row and nth column of the Jacobian matrix $\mathbf{J}_f(\mathbf{x})$ of $\mathbf{f}(\mathbf{x})$. Now the gradient can be written as

$$\mathbf{g}_\varepsilon(\mathbf{x}) = \mathbf{J}_f^T(\mathbf{x})\,\mathbf{f}(\mathbf{x}) \tag{1.105}$$

If $\mathbf{f}(\mathbf{x})$ is linear and can therefore be represented as the sum of the first two terms of its Taylor series in Eq. (1.98), then the gradient is:

$$\mathbf{g}_\varepsilon(\mathbf{x}) = \mathbf{J}^T(\mathbf{x})\,\mathbf{f}(\mathbf{x}) = \mathbf{J}^T(\mathbf{x}_0)\,[\mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\Delta\mathbf{x}] = \mathbf{J}_0^T(\mathbf{f}_0 + \mathbf{J}_0\Delta\mathbf{x}) \tag{1.106}$$

where $\mathbf{x}_0$ is any chosen initial guess. If we assume $\mathbf{x}$ is the optimal solution at which $\varepsilon(\mathbf{x})$ is minimized and $\mathbf{g}_\varepsilon(\mathbf{x})$ is zero:

$$\mathbf{g}_\varepsilon(\mathbf{x}) = \mathbf{J}_0^T(\mathbf{f}_0 + \mathbf{J}_0\Delta\mathbf{x}) = \mathbf{0} \tag{1.107}$$

then the increment $\Delta\mathbf{x}$ can be found by solving the equation

$$\Delta\mathbf{x} = -(\mathbf{J}_0^T\mathbf{J}_0)^{-1}\mathbf{J}_0^T\mathbf{f}_0 = -\mathbf{J}_0^-\,\mathbf{f}_0 \tag{1.108}$$

Here $\mathbf{J}_0^- = (\mathbf{J}_0^T\mathbf{J}_0)^{-1}\mathbf{J}_0^T$ is the *pseudo-inverse* (Section A.15) of the non-square matrix $\mathbf{J}_0$. Now the optimal solution can be found as:

$$\mathbf{x}^* = \mathbf{x}_0 + \Delta\mathbf{x} = \mathbf{x}_0 - \mathbf{J}_0^-\,\mathbf{f}_0 \tag{1.109}$$

However, as $\mathbf{f}(\mathbf{x})$ is nonlinear in general, the sum of the first two terms of its Taylor series is only an approximation. Consequently the result $\mathbf{x} = \mathbf{x}_0 + \Delta\mathbf{x}$ above is not the optimal solution, but an estimate which can be improved by carrying out this step iteratively:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \Delta\mathbf{x}_n = \mathbf{x}_n - \mathbf{J}_n^-\,\mathbf{f}_n \tag{1.110}$$

This iteration will converge to $\mathbf{x}^*$ at which $\mathbf{g}_\varepsilon(\mathbf{x}^*) = \mathbf{0}$, and the squared error $\varepsilon(\mathbf{x})$ is minimized.

Especially, for a linear equation system $\mathbf{f}(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$, the Jacobian is simply $\mathbf{J}_f(\mathbf{x}_n) = \mathbf{A}$, and the optimal solution is

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{A}^-\mathbf{f}_n = \mathbf{x}_n - (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T(\mathbf{A}\mathbf{x}_n - \mathbf{b}) = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{A}^-\mathbf{b} \tag{1.111}$$

i.e., the optimal solution can be found from any initial guess in a single step. This result is the same as that in Eq. (1.7).

Comparing Eqs. (1.100) and (1.110), we see that the two algorithms are essentially the same, with the only difference that the regular inverse $\mathbf{J}^{-1}$ is used when $M = N$, but the pseudoinverse $\mathbf{J}^-$ is used when $M > N$ and $\mathbf{J}^{-1}$ does not exist.

The Newton-Raphson method assumes the availability of the analytical expressions of all partial derivatives $J_{mn} = \partial f_m(\mathbf{x})/\partial x_n$ $(m = 1, \cdots, M,\ n =$

$1, \cdots, N$) in the Jacobian matrix $\mathbf{J}$. However, when this is not the case, J_{mn} need to be approximated by forward or central difference (secant) method:

$$J_{mn} = \frac{\partial f_m(x_1, \cdots, x_N)}{\partial x_n} \approx \frac{f_m(x_1, \cdots, x_n + h, \cdots, x_N) - f_m(x_1, \cdots, x_n, \cdots, x_N)}{h}$$
$$\approx \frac{f_m(x_1, \cdots, x_n + h, \cdots, x_N) - f_m(x_1, \cdots, x_n - h, \cdots, x_N)}{2h} \tag{1.112}$$

where h is a small increment.

Example 1

$$\begin{cases} 3\,x_1 - \cos(x_2 x_3) - 3/2 = 0 \\ 4x_1^2 - 625\,x_2^2 + 2x_3 - 1 = 0 \\ 20\,x_3 + e^{-x_1 x_2} + 9 = 0 \end{cases}$$

$$\mathbf{J} = \begin{bmatrix} 3 & x_3 \sin(x_2 x_3) & x_2 \sin(x_2 x_3) \\ 8\,x_1 & -1250\,x_2 & 2 \\ -x_2\,e^{-x_1 x_2} & -x_1\,e^{-x_1 x_2} & 20 \end{bmatrix}$$

n	$\mathbf{x}$	$error$
0	$(1.000000, 1.000000, 1.000000)$	$6.207e+02$
1	$(1.232701, 0.503132, -0.473253)$	$1.541e+02$
2	$(0.832592, 0.251806, -0.490636)$	$3.884e+01$
3	$(0.833238, 0.128406, -0.494702)$	$9.517e+00$
4	$(0.833275, 0.069082, -0.497147)$	$2.200e+00$
5	$(0.833281, 0.043585, -0.498206)$	$4.063e-01$
6	$(0.833282, 0.036117, -0.498517)$	$3.486e-02$
7	$(0.833282, 0.035343, -0.498549)$	$3.741e-04$
8	$(0.833282, 0.035335, -0.498549)$	$4.498e-08$
9	$(0.833282, 0.035335, -0.498549)$	$5.551e-16$

Example 2

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \qquad \begin{cases} f_1(\mathbf{x}) = x_1^2 - 2x_1 + x_2^2 - x_3 + 1 = 0 \\ f_2(\mathbf{x}) = x_1 x_2^2 - x_1 - 3x_2 + x_2 x_3 + 2 = 0 \\ f_3(\mathbf{x}) = x_1 x_3^2 - 3x_3 + x_2 x_3^2 + x_1 x_2 = 0 \end{cases}$$

$$\mathbf{J} = \begin{bmatrix} 2x_1 - 2 & 2x_2 & -1 \\ x_2^2 - 1 & 2x_1 x_2 - 3 + x_3 & x_2 \\ x_3^2 + x_2 & x_3^2 + x_1 & 2x_1 x_3 - 3 + 2x_2 x_3 \end{bmatrix}$$

With $\mathbf{x}_0 = [1,\ 2,\ 3]^T$, we get a root:

n	$\mathbf{x}$	$error$
0	$(1.00000, 2.00000, 3.00000)$	$2.064e+01$
1	$(0.10256, 1.64103, 2.56410)$	$4.303e+00$
2	$(1.52062, 1.41113, 0.19859)$	$2.689e+00$
3	$(1.94123, 0.77134, 0.89465)$	$1.217e+00$
4	$(1.06737, 1.19117, 0.48353)$	$1.144e+00$
5	$(1.26825, 0.95182, 0.88028)$	$3.323e-01$
6	$(0.95899, 1.03384, 0.96813)$	$1.171e-01$
7	$(1.00171, 1.00007, 0.99718)$	$4.162e-03$
8	$(1.00000, 1.00000, 1.00000)$	$6.701e-06$

With $\mathbf{x}_0 = [0,\ 0,\ 0]^T$, we get another root:

n	$\mathbf{x}$	$error$
0	$(0.00000, 0.00000, 0.00000)$	$2.236e+00$
1	$(0.50000, 0.50000, 0.00000)$	$5.728e-01$
2	$(0.83951, 0.47531, 0.13580)$	$1.175e-01$
3	$(0.98582, 0.41849, 0.15069)$	$2.639e-02$
4	$(1.05417, 0.38715, 0.14717)$	$6.088e-03$
5	$(1.08565, 0.37339, 0.14558)$	$1.264e-03$
6	$(1.09693, 0.36849, 0.14503)$	$1.618e-04$
7	$(1.09888, 0.36764, 0.14494)$	$4.817e-06$

This is a nice, concise explanation of the bones of the method, and it should be used earlier when the basic method is being described to the reader.

Broyden's method

In the Newton-Raphson method, two main operations are carried out in each iteration: (a) evaluate the Jacobian matrix $\mathbf{J}_\mathbf{f}(\mathbf{x}_n)$ and (b) obtain its inverse $\mathbf{J}_\mathbf{f}^{-1}(\mathbf{x}_n)$. To avoid the expensive computation for these operations, we consider using Broyden's method, one of the *quasi-Newton methods*, which approximates the inverse of the Jacobian $\mathbf{J}_{n+1}^{-1} = \mathbf{J}^{-1}(\mathbf{x}_{n+1})$ from the $\mathbf{J}_n^{-1} = \mathbf{J}^{-1}(\mathbf{x}_n)$ in the previous iteration step, so that it can be updated iteratively from the initial $\mathbf{J}_0^{-1} = \mathbf{J}^{-1}(\mathbf{x}_0)$.

"Quasi-Newton method" should be explicitly defined.

We first consider in the single-variable case how to estimate the next derivative $f'_{n+1} = f'(x_{n+1})$ from the current $f'_n = f'(x_n)$ by the secant method:

$$f'_{n+1} \approx \hat{f}'_{n+1} = \frac{f_{n+1} - f_n}{x_{n+1} - x_n} = \frac{\delta f_n}{\delta x_n} = \frac{\hat{f}'_n \delta x_n - \hat{f}'_n \delta x_n + \delta f_n}{\delta x_n}$$
$$= \hat{f}'_n + \frac{\delta f_n - \hat{f}'_n \delta x_n}{\delta x_n} = \hat{f}'_n + \hat{\delta} f'_n \tag{1.113}$$

where

- $\delta f_n = f_{n+1} - f_n$ is the true increment of the function over the interval $\delta x_n = x_{n+1} - x_n$;

- $\hat{f}'_n \delta x_n$ is the estimated increment of the function based on the previous derivate $\hat{f}'_n$;
- $\hat{\delta} f'_n$ is the estimated increment of the derivative:

$$\hat{\delta} f'_n = \frac{\delta f_n - \hat{f}'_n \delta x_n}{\delta x_n} \tag{1.114}$$

The equation above indicates that the derivative f'_{n+1} in the $(n+1)th$ step can be estimated by adding the estimated increment $\hat{\delta} f'_n$ to the derivative $\hat{f}'n$ in the current nth step.

Having obtained $\hat{f}'_{n+1}$, we can use the same iteration in the Newton-Raphson method to find x_{n+1}:

$$x_{n+1} = x_n - \frac{f(x_n)}{\hat{f}'_n} \tag{1.115}$$

This method for single-variable case can be generalized to multiple variable case for solving $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. Following the way we estimate the increment of the derivative of a single-variable function in Eq. (1.114), here we can estimate the increment of the Jacobian of a multi-variable function:

$$\delta \hat{\mathbf{J}}_n = \frac{(\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n)\ \delta \mathbf{x}_n^T}{\delta \mathbf{x}_n^T\ \delta \mathbf{x}_n} = \frac{(\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n) \delta \mathbf{x}_n^T}{||\delta \mathbf{x}_n||^2} \tag{1.116}$$

where $\delta \mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$ and $\delta \mathbf{f}_n = \mathbf{f}_{n+1} - \mathbf{f}_n = \mathbf{f}(\mathbf{x}_{n+1}) - \mathbf{f}(\mathbf{x}_n)$. Now in each iteration, we can update the estimated Jacobian as well as the estimated root:

$$\hat{\mathbf{J}}_{n+1} = \hat{\mathbf{J}}_n + \delta \hat{\mathbf{J}}_n, \qquad \mathbf{x}_{n+1} = \mathbf{x}_n + \delta \mathbf{x}_n = \mathbf{x}_n - \hat{\mathbf{J}}_n^{-1} \mathbf{f}(\mathbf{x}_n) \tag{1.117}$$

The algorithm can be further improved so that the inverse of the Jacobian $\mathbf{J}_n$ is avoided. Specifically, consider the inverse Jacobian:

$$\hat{\mathbf{J}}_{n+1}^{-1} = \left(\hat{\mathbf{J}}_n + \delta \hat{\mathbf{J}}_n\right)^{-1} = \left[\hat{\mathbf{J}}_n + \frac{(\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n) \delta \mathbf{x}_n^T}{||\delta \mathbf{x}_n||^2}\right]^{-1} \tag{1.118}$$

We can apply the Sherman-Morrison formula (Section A.6):

$$(\mathbf{A} + \mathbf{u}\mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}}{1 + \mathbf{v}^T\mathbf{A}^{-1}\mathbf{u}} \tag{1.119}$$

to the right-hand side of the equation above by defining

$$\mathbf{A} = \hat{\mathbf{J}}_n, \qquad \mathbf{u} = \frac{\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n}{||\delta \mathbf{x}_n||^2}, \qquad \mathbf{v} = \delta \mathbf{x}_n \tag{1.120}$$

and rewrite it as:

$$\hat{\mathbf{J}}_{n+1}^{-1} = \hat{\mathbf{J}}_n^{-1} - \frac{\hat{\mathbf{J}}_n^{-1}\,(\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n^T)/||\delta \mathbf{x}_n||^2\, \delta \mathbf{x}_n^T\, \hat{\mathbf{J}}_n^{-1}}{1 + \delta \mathbf{x}_n^T \hat{\mathbf{J}}_n^{-1} (\delta \mathbf{f}_n - \hat{\mathbf{J}}_n \delta \mathbf{x}_n)/||\delta \mathbf{x}_n||^2} = \hat{\mathbf{J}}_n^{-1} - \frac{(\hat{\mathbf{J}}_n^{-1} \delta \mathbf{f}_n - \delta \mathbf{x}_n)\, \delta \mathbf{x}_n^T\, \hat{\mathbf{J}}_n^{-1}}{\delta \mathbf{x}_n^T \hat{\mathbf{J}}_n^{-1} \delta \mathbf{f}_n} \tag{1.121}$$

We see that the next $\hat{\mathbf{J}}_{n+1}^{-1}$ can be iteratively estimated directly from the previous $\hat{\mathbf{J}}_n^{-1}$, thereby avoiding computing the inverse of $\hat{\mathbf{J}}_n$ altogether. The algorithm is listed below:

- Select $\mathbf{x}_0$
- Find $\mathbf{f}_0 = \mathbf{f}(\mathbf{x}_0)$, $\mathbf{J}_0 = \mathbf{J}(\mathbf{x}_0)$, and $\mathbf{J}_0^{-1}$
- $\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}_0^{-1}\mathbf{f}_0$
- $n = 0$
- While $||\mathbf{x}_{n+1} - \mathbf{x}_n|| > tol$ do
  - $\delta\mathbf{x}_n = \mathbf{x}_{n+1} - \mathbf{x}_n$
  - $\mathbf{f}_{n+1} = \mathbf{f}(\mathbf{x}_{n+1})$
  - $\delta\mathbf{f} = \mathbf{f}_{n+1} - \mathbf{f}_n$
  - $\mathbf{J}_n^{-1} = \mathbf{J}_n^{-1} - (\delta\mathbf{x}_n - \mathbf{J}_n^{-1}\delta\mathbf{f}_n)\delta\mathbf{x}_n^T\mathbf{J}_n^{-1}/(\delta\mathbf{x}_n^T\mathbf{J}_n^{-1}\delta\mathbf{f}_n)$
  - $\mathbf{x}_n = \mathbf{x}_{n+1}$, $\mathbf{f}_n = \mathbf{f}_{n+1}$
  - $\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{J}_n^{-1}\mathbf{f}_n$
  - $n = n + 1$