1 Introduction

In this section we consider methods for solving nonlinear equations. Given a nonlinear function $f(x)$, we seek a value of x for which

$$f(x) = 0$$

Such a solution value for x is called a **root** of the equation, and a zero of the function $f(x)$. An example of a nonlinear equation with one variable is

$$f(x) = x^2 - 4\sin(x) = 0$$

As can be seen from fig.1, the function has roots at 0.0 and near 1.9. Graphical examination of a function is often a good way to find the neighborhood of roots.

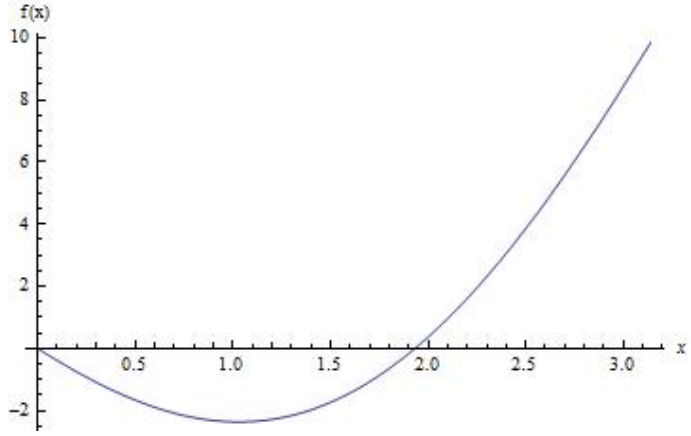

Fig.1

- Graphical techniques are of limited practical value because they are not precise. However, graphical methods can be utilized to obtain rough estimates of roots. These estimates can be employed as starting guesses for numerical methods discussed in this chapter.

2 Solving an equation of one variable $f(x) = 0$

The methods used for solving this type of equation numerically can be divided into two major groups. **bracketing methods**, and **open methods**. In bracketing methods such as the **bisection method and the regula falsi method**, an interval that includes the solution is identified. By definition, the end points of the interval are the upper and lower bound of the solution. Then, by using a numerical scheme, the size of the interval is successively reduced until the distance between the end points is less than the desired accuracy of the solution. In the open method, such as **Newton's method, secant method or fixed point iteration method**, an initial estimate for the solution is assumed. The value of this initial guess should be close to the actual solution. Then, by using a numerical scheme, better (more accurate) values for the solution are calculated. Bracketing methods always converge to the solution. Open methods are usually more efficient but sometimes might not yield the solution.

3 Bracketing Methods

3.1 Bisection method

In general, if $f(x)$ is real and continuous in the interval from a to b and $f(a)$ and $f(b)$ have opposite signs, that is, $f(a) \cdot f(b) < 0$, then there is at least one real root between a and b. The bisection method, which is alternatively called binary chopping, interval halving, or Bolzanos method, is one type of incremental search method in which the interval is always divided in half. If a function changes sign over an interval, the function value at the midpoint is evaluated. The location of the root is then determined as lying at the midpoint of the subinterval within which the sign change occurs. The process is repeated to obtain refined estimates.

3.1.1 Algorithm for the bisection method

A graphical depiction of the method is provided in Fig.2.

1. Choose the first interval by finding points a and b such that a solution exists between them. This means that f(a) and f(b) have different signs such that $f(a) \cdot f(b) < 0$. The points can be determined by examining the plot of versus x.
2. Calculate the first estimate of the numerical solution x_{NS1} by:

$$x_{NS1} = \frac{a+b}{2}$$

3. Determine whether the true solution is between a and x_{NS1}, or between x_{NS1} and b. This is done by checking the sign of the product $f(a) \cdot f(x_{NS1})$:
   - If $f(a) \cdot f(x_{NS1}) < 0$, the true solution is between a and x_{NS1} .
   - If $f(a) \cdot f(x_{NS1}) > 0$, the true solution is between x_{NS1} and b.
4. Select the subinterval that contains the true solution (a to x_{NS1}, or x_{NS1} to b) as the new interval [a, b] , and go back to step 2.
5. Steps 2 through 4 are repeated until a specified tolerance or error bound is attained.

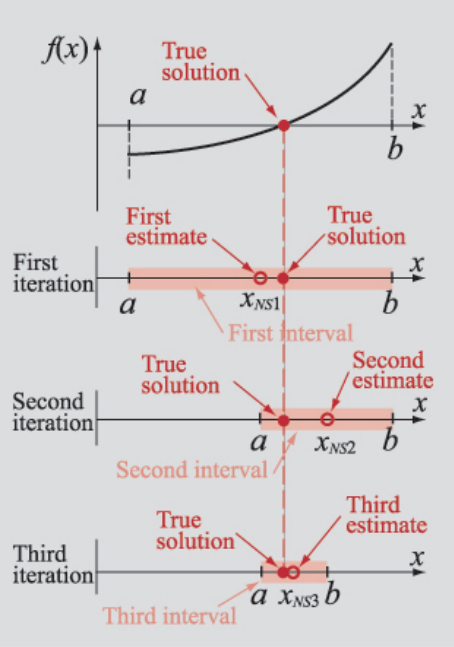

Fig.2

3.1.2 Termination Criteria

Ideally, the bisection process should be stopped when the true solution is obtained. This means that the value of x_{NS} is such that $f(x_{NS}) = 0$. In reality, this true solution generally cannot be found computationally. Two possible options depending on the problem to be solved are

- If it is known that the solution is within the domain [a, b], then the numerical solution can be taken as $(a+b)/2$, and the tolerance (ϵ_a) to be half the distance between a and b. In practice, the iteration is ended if the true solution is found or the tolerance of the iteration is smaller than the desired tolerance.

$$root = \frac{a+b}{2} \pm \epsilon_a$$

$$\epsilon_a = \left| \frac{b-a}{2} \right|$$

$$\epsilon_a \leq \epsilon_s$$

- An alternative to this is to terminate the method when the estimated relative as defined next is less than than a predefined value.

  – when two numerical estimates for the solution are known. This is the case when numerical solutions are calculated iteratively, where in each new iteration a more accurate solution is calculated. If $x_{NS}^{(n)}$ is the estimated numerical solution in the last iteration and $x_{NS}^{(n-1)}$ is the estimated numerical solution in the preceding iteration, then an Estimated Relative Error can be defined by:

$$estimated\ relative\ error = \left| \frac{\mid x_{NS}^{(n)} - \mid x_{NS}^{(n-1)}}{\mid x_{NS}^{(n)}} \right|$$

  When the estimated numerical solutions are close to the true solution, it is anticipated that the difference $\mid x_{NS}^{(n)} - x_{NS}^{(n-1)} \mid$ is small compared to the value of $x_{NS}^{(n)}$, and the Estimated Relative Error is approximately the same as the True Relative Error.

3.1.3 Additional notes

- The method always converges to an answer, provided a root was trapped in the interval to begin with.
- The method may fail when the function is tangent to the axis and does not cross the x-axis at f(x)=0.
- The method converges slowly relative to other methods.

3.2 Regula Falsi method

The convergence process in the bisection method is very slow. It depends only on the choice of end points of the interval $[a, b]$. The function $f(x)$ does not have any role in finding the next approximation (which is just the mid-point of a and c), it is used only to decide the next smaller interval $[a, c]$ or $[c, b]$. For example, if f(a) is much closer to zero than f(b), it is likely that the root is closer to a than to b. A better approximation to next estimate can be obtained by taking the straight line joining the points (a,f(a)) and (b,f(b)) intersecting the x-axis. To obtain the value of next approximation X_{NS}

$$X_{NS} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

The new interval to consider is either $[a, X_{NS}]$ or $[X_{NS}, c]$ depending on whether $f(a) \cdot f(X_{NS}) < 0$ or $f(b) \cdot f(X_{NS}) < 0$. the process is then repeated until the numerical solution is deemed accurate enough.

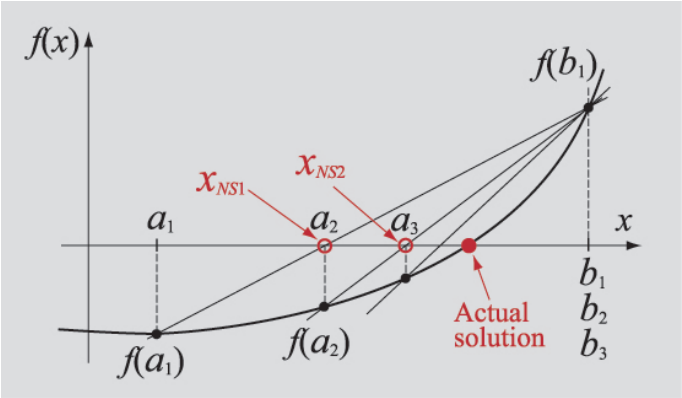

Fig.3

3.2.1 Algorithm for the Regula Falsi method

Identical to the one for the bisection except that

$$X_{NS} = \frac{af(b) - bf(a)}{f(b) - f(a)}$$

3.2.2 Termination Criteria

Same stopping criterion as in the bisection method

3.2.3 Additional notes

- The method always converges to an answer, provided a root was trapped in the interval $[a, b]$.
- Frequently, as in the case shown in Fig.3, the function in the interval is either concave up or concave down. In this case, one of the endpoints of the interval stays the same in all the iterations, while the other endpoint advances toward the root. In other words, the numerical solution advances toward the root only from one side. The convergence toward the solution could be faster if the other endpoint would also move toward the root. Several modifications have been introduced to the regula falsi method that make the subinterval in successive iterations approach the root from both sides.

4 Open Methods

4.1 Newton's Method

Perhaps the most widely used of all root-locating formulas is the Newton-Raphson method (fig.4). If the initial guess at the root is xi, a tangent can be extended from the point [xi, f(xi)]. The point where this tangent crosses the x axis usually represents an improved estimate of the root. The Newton-Raphson method can be derived on the basis of this geometrical interpretation (an alternative method based on the Taylor series is presented later in this paragraph).

The numerical approximation of the derivative of the function f(x) at the current approximation xi is

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}}$$

So the next Newton approximation (iterate) is

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

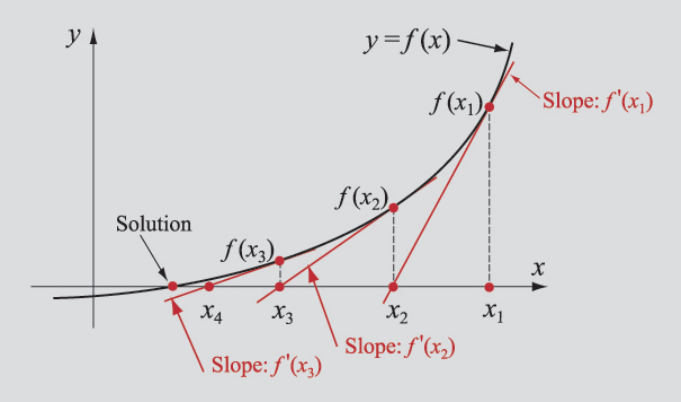

Fig.4

4.1.1 Algorithm for Newton's Method

1. Choose a point x_1 as an initial guess of the solution.
2. For i=1, 2, ...until the error is smaller than a specified value, calculate x_{i+1} by using the previous equation.

4.1.2 Termination Criteria

- In practice, the iterations are stopped when an estimated error is smaller than some predetermined value. A tolerance in the solution, as in the bisection method, cannot be calculated since bounds are not known. Two error estimates that are typically used with Newtons method are:

  1. **Estimated relative error:** The iterations are stopped when the estimated relative error is smaller than a specified value ϵ

  $$\frac{|\, X_{NS}^{(n)} - X_{NS}^{(n-1)} \,|}{|\, X_{NS}^{(n)} \,|} < \epsilon$$

  2. **Tolerance in $f(x)$:** the iterations are stopped when the absolute value of $f(X_{NS})$ is smaller than some number δ

  $$|\, f(X_{NS}) \,| < \delta$$

4.1.3 Additional notes

- Newton's method can be derived using Taylor series

  $$f(x) = f(x_1) + (x - x_1)f'(x_1) + \frac{1}{2!}(x - x_1)^2 f''(x_1) + ...$$

  If x_2 is a solution of the equation $f(x) = 0$ and x_1 is close to x_2 then

  $$f(x_2) = 0 = f(x_1) + (x_2 - x_1)f'(x_1) + \frac{1}{2!}(x_2 - x_1)^2 f''(x_1) + ...$$

  By considering only the first two terms of the expansion we have

$$f(x_1) + (x_2 - x_1)f'(x_1) = 0$$

and from which

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

- Convergence problems typically occur when the value of $f'(x)$ is close to zero in the vicinity of the solution

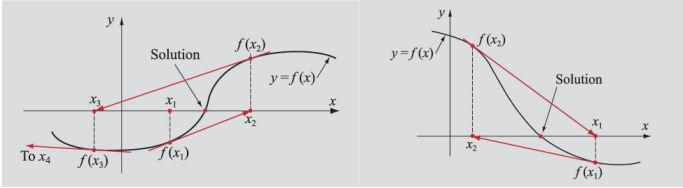

Fig.5

- There is no general convergence criterion for Newton-Raphson. Its convergence depends on the nature of the function and on the accuracy of the initial guess. The only remedy is to have an initial guess that is sufficiently close to the root. And for some functions, no guess will work! Good guesses are usually predicated on knowledge of the physical problem setting or on devices such as graphs that provide insight into the behavior of the solution.
- A function $f'(x)$ has to be substituted in the iterative formula.

4.2 Secant method

The secant method uses two points in the neighborhood of the solution to determine a new estimate for the solution (Fig.6). The two points (marked as x_1 and x_2 in the figure) are used to define a straight line (secant line), and the point where the line intersects the x-axis (marked as x_3 in the figure) is the new estimate for the solution. As shown, the two points can be on one side of the solution or the solution can be between the two points. The slope of the secant line is given by:

$$\frac{f(x_2) - f(x_1)}{x_2 - x_1} = \frac{f(x_2)}{x_2 - x_3}$$

which can be solved for x_3

$$x_3 = x_2 - \frac{f(x_2)}{\dfrac{f(x_2) - f(x_1)}{x_2 - x_1}}$$

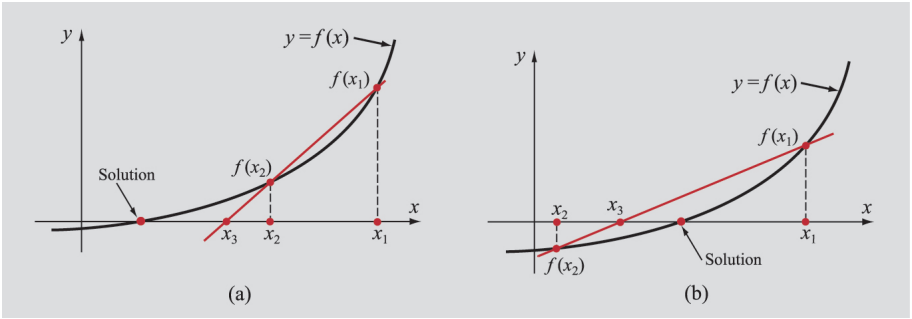

Fig.6

We can now repeat the process. Use x_2 and x_3 to produce another secant line, and then use its root to approximate α. This yields the general iteration formula

$$x_{n+1} = x_n - \frac{f(x_n)}{\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}}$$

4.2.1 Additional notes

- When the two points that define the secant line are close to each other, the method approximates Newton's method.

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

- Unlike Newton's method, it is not necessary to know the analytical form of $f'(x)$

4.3 Fixed-Point iteration method

Open methods employ a formula to predict the root. Such a formula can be developed for simple fixed-point iteration (or, as it is also called, one-point iteration or successive substitution) by rearranging the function f(x)= 0 so that x is on the left-hand side of the equation:

$$x = g(x)$$

This transformation can be accomplished either by algebraic manipulation or by simply adding x to both sides of the original equation. For example,

$$x^2 - 2x + 3 = 0$$

can be simply manipulated to yield

$$x = \frac{x^2 + 3}{2}$$

whereas $sinx = 0$ could be put into the same form by adding x to both sides to yield $x = sinx + x$

The utility of the previous equation is that it provides a formula to predict a new value of x as a function of an old value of x. Thus, given an initial guess at the root x_i,the equation can be used to compute a new estimate x_{i+1} as expressed by the iterative formula

$$x_{i+1} = g(x_i)$$

When the method works, the values of x that are obtained are successive iterations that progressively converge toward the solution. Two such cases are illustrated graphically in Fig.7.

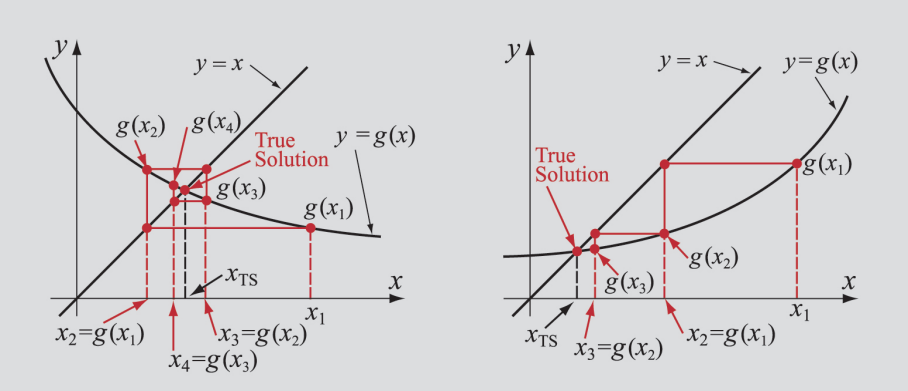

Fig.7

It is possible, however, that the iterations will not converge toward the fixed point, but rather diverge away. This is shown in Fig.8. The figure shows that even though the starting point is close to the solution, the subsequent points are moving farther away from the solution

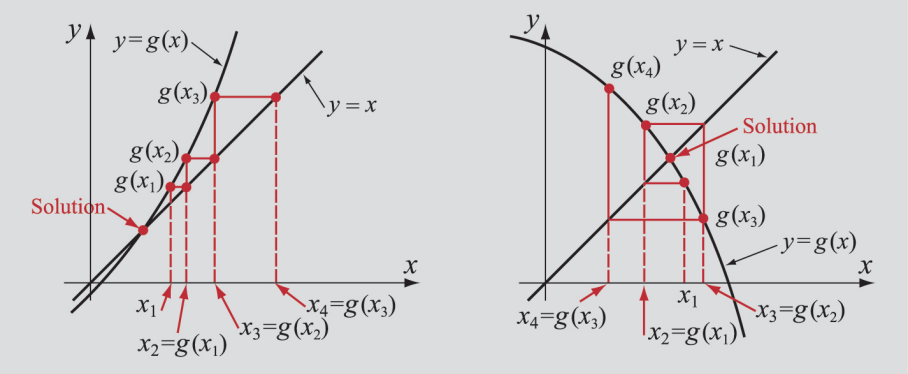

Fig.8

4.3.1 Choosing the appropriate iteration function g(x)

For a given equation $f(x) = 0$, the iteration function is not unique since it is possible to change the equation into the form in different ways. This means that several iteration functions $g(x)$ can be written for the same equation. A $g(x)$ that should be used for the iteration process is one for which the iterations converge toward the solution. There might be more than one form that can be used, or it may be that none of the forms are appropriate so that the fixed-point iteration method cannot be used to solve the equation. In cases where there are multiple solutions, one iteration function may yield one root, while a different function yields other roots. Actually, it is possible to determine ahead of time if the iterations converge or diverge for a specific.

The fixed-point iteration method converges if, in the neighborhood of the fixed point, the derivative of $g(x)$ has an absolute value that is smaller than 1 (also called Lipschitz continuous):

$$| g'(x) | < 1$$

4.3.2 Additional notes

- As with Newton's method, the iterations can be stopped either when the relative error or the tolerance in f(x) is smaller than some predetermined value.

5 MATLAB built in functions

MATLAB has two build in functions for solving equations with one variable. The **fzero** command can be used to find a root of any equation, and **roots** command can be used for finding the roots of a polynomial.

$$x = fzero(function, x_o)$$

$$r = roots(p)$$

where x is the solution, f is the function to be solved, x_o a value of x near to where the function crosses the axis, r is a column vector with the roots of the polynomial, and p is a row vector with the coefficients of the polynomial.

6 Equations with multiple solutions

many non linear equations of the form $f(x) = 0$ have multiple solutions or roots. A general strategy for finding these roots is

- Determine the approximate location of the roots by defining smaller intervals over which the roots exist. This can be done by plotting the function (as shown in Fig. 3-21) or by evaluating the function over a set of successive points and looking for sign changes of the function.
- applying any of the method discussed before over a restricted subinterval.

7 Systems of nonlinear equations

A system of nonlinear equations consists of two or more nonlinear equations that have to be solved simultaneously.

7.1 Newton's method for solving a system of non linear equations

In this section we discuss how a small system of nonlinear equation can be soled using the so called Newton-Ralphson method, for simplicity the method is illustrated for a system of two equations and two variables.

Consider solving a system of two non-linear equations of the form

$$\begin{cases} f_1(x,y) = 0 \\ \\ f_2(x,y) = 0 \end{cases}$$

If (x_2, y_2) are the true solution to the system of equations and if (x_1, y_1) are estimates to the solution assumed to be close to the true solution then using Taylor series expansions about (x_1, y_1) gives us

$$\begin{cases} f_1(x_2,y_2) = 0 = f_1(x_1,y_1) + (x_2 - x_1)\dfrac{\partial f_1}{\partial x} \mid_{x_1,y_1} + (y_2 - y_1)\dfrac{\partial f_1}{\partial y} \mid_{x_1,y_1} + ... \\ \\ f_2(x_2,y_2) = 0 = f_2(x_1,y_1) + (x_2 - x_1)\dfrac{\partial f_2}{\partial x} \mid_{x_1,y_1} + (y_2 - y_1)\dfrac{\partial f_2}{\partial y} \mid_{x_1,y_1} + ... \end{cases}$$

Neglecting the higher order terms we get a system of two linear equations which can be solved using Crammer's rule

$$\begin{cases} (x_2 - x_1)\dfrac{\partial f_1}{\partial x} \mid_{x_1,y_1} + (y_2 - y_1)\dfrac{\partial f_1}{\partial y} \mid_{x_1,y_1} = -f_1(x_1,y_1) \\ \\ (x_2 - x_1)\dfrac{\partial f_2}{\partial x} \mid_{x_1,y_1} + (y_2 - y_1)\dfrac{\partial f_2}{\partial y} \mid_{x_1,y_1} = -f_2(x_1,y_1) \end{cases}$$

$$\begin{cases} (x_2 - x_1) = \dfrac{-f_1(x_1,y_1)\dfrac{\partial f_2}{\partial x} \mid_{x_1,y_1} + f_2(x_1,y_1)\dfrac{\partial f_1}{\partial x} \mid_{x_1,y_1}}{J(f_1(x_1,y_1), f_2(x_1,y_1))} \\ \\ (y_2 - y_1) = \dfrac{-f_2(x_1,y_1)\dfrac{\partial f_1}{\partial x} \mid_{x_1,y_1} + f_1(x_1,y_1)\dfrac{\partial f_2}{\partial x} \mid_{x_1,y_1}}{J(f_1(x_1,y_1), f_2(x_1,y_1))} \end{cases}$$

where

$$J(f_1, f_2) = det \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} \tag{1}$$

Newton's method can be generalized to the case of a system of n nonlinear equations with n unknowns such as

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

Following the same procedure as described before the next approximation of the solution is obtained using a Taylor series expansion about the current approximation which result to solving a system of linear equations of the form

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial_2} & .... & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & .... & \frac{\partial f_2}{\partial x_n} \\ \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & .... & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ ... \\ \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} -f_1 \\ -f_2 \\ ... \\ \\ -f_n \end{bmatrix} \tag{2}$$

The new approximate solution is obtained from

$$\begin{cases} x_{1,i+1} = x_{1,i} + \Delta x_1 \\ x_{2,i+1} = x_{2,i} + \Delta x_2 \\ \\ x_{n,i+1} = x_{n,i} + \Delta x_n \end{cases}$$

7.1.1 Additional notes

As with Newton's method for a single nonlinear equation, convergence is not guaranteed. Newton's iterative procedure for solving a system of nonlinear equations will likely converge provided the following three conditions are met:

- The functions and their derivatives must be continuous and bounded near the solution (root).
- The Jacobian must be nonzero, that is,$J(f_1, f_2, ..., f_n) \neq 0$, near the solution.
- The initial estimate (guess) of the solution must be sufficiently close to the true solution.

7.2 Fixed-point iteration method for solving a system of nonlinear equations

The fixed-point iteration method discussed in Section 3.7 for solving a single nonlinear equation can be extended to the case of a system of nonlinear equations. A system of n nonlinear equations with the unknowns, $x_1, x_2, ..., x_n$, has the form:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) = 0 \\ f_2(x_1, x_2, ..., x_n) = 0 \\ \\ f_n(x_1, x_2, ..., x_n) = 0 \end{cases}$$

The system can be rewritten in the form

$$\begin{cases} x_1 = g_1(x_1, x_2, ..., x_n) \\ x_2 = g_2(x_1, x_2, ..., x_n) \\ \\ x_n = g_n(x_1, x_2, ..., x_n) \end{cases}$$

where the gs are the iteration functions. The solution process starts by guessing a solution, $x_{1,1}, x_{2,1}, ..., x_{n,1}$, which is substituted on the right- hand side of the previous equation. The values that are calculated by this equation are the new (second) estimate of the solution, . The new estimate is substituted back on the right-hand side of the same equation to give a new solution, and so on. When the method works, the new estimates of the solution converge toward the true solution. In this case, the process is continued until the desired accuracy is achieved. For example, the estimated relative error is calculated for each of the variables, and the iterations are stopped when the largest relative error is smaller than a specified value.

Convergence of the method depends on the form of the iteration functions. For a given problem there are many possible forms of iteration functions . In general, several forms might be appropriate for one solution, or in the case where several solutions exist, different iteration functions need to be used to find the multiple solutions. When using the fixed-point iteration method, one can try various forms of iteration functions, or it may be possible in some cases to determine ahead of time if the solution will converge for a specific choice of gs.

The fixed-point iteration method applied to a set of simultaneous nonlinear equations will converge under the following sufficient (but not necessary) conditions:

- $g_1, g_2, ..., g_n, \frac{\partial g_1}{\partial x_1}, ..., \frac{\partial g_1}{\partial x_n}, \frac{\partial g_2}{\partial x_1}, ..., \frac{\partial g_2}{\partial x_n}, \frac{\partial g_n}{\partial x_1}, ..., \frac{\partial g_n}{\partial x_n}$ are continuous in the neighborhood of the solution.
- $$\begin{cases} \mid \frac{\partial g_1}{\partial x_1} \mid + \mid \frac{\partial g_1}{\partial x_2} \mid + ... \mid \frac{\partial g_1}{\partial x_n} \mid \leq 1 \\ \mid \frac{\partial g_2}{\partial x_1} \mid + \mid \frac{\partial g_2}{\partial x_2} \mid + ... \mid \frac{\partial g_2}{\partial x_n} \mid \leq 1 \\ \\ \mid \frac{\partial g_n}{\partial x_1} \mid + \mid \frac{\partial g_n}{\partial x_2} \mid + ... \mid \frac{\partial g_n}{\partial x_n} \mid \leq 1 \end{cases}$$
- The initial guess, $x_{1,1}, x_{2,1}, ..., x_{n,1}$ is sufficiently close to the solution.