

A brief note on finding roots of nonlinear equations

Consider solving a nonlinear equation in one variable x

$$f(x) = 0 \quad (1)$$

We will simply assume $f(x)$ to be a smoothly varying function that can have extrema, minima and/or maxima, over the range we are interested in. A few typical examples of $f(x)$ are

$$f(x) = \cos x - x^3, \quad 3x + \sin x - \exp x, \quad x \exp x - 2, \quad x^3 + 3x - 5 = 0 \quad (2)$$

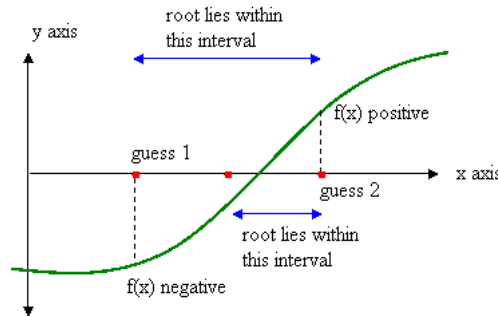
If a value x_0 in the interval (a_0, b_0) satisfies the equation (1) *i.e.* $f(x_0) = 0$, then x_0 is a *root* or *zero* of the function $f(x)$ and is **one** of the solutions in that interval. Since $f(x)$ is a continuous function and there exist two points a_0 and b_0 such that $f(a_0)$ and $f(b_0)$ are of opposite signs, then according to *intermediate value theorem*, the function $f(x)$ has a root in the interval (a_0, b_0) .

Things get complicated when we encounter a system of nonlinear equations, say in two variable, for example

$$x e^y = 1 \quad \text{and} \quad y^2 - x^2 = 1 \quad (3)$$

Many methods of solving nonlinear equations in one variable do not generalize directly to more than one variable problem. The one that does is the Newton's method, which is arguably the most popular method. For that, we need additional knowledge of *Jacobian* and *Hessian* matrices.

Finding root of one variable nonlinear equations numerically always start with guesses $[a_0, b_0]$, either found by trial-and-error or educated guess, at which $f(x)$ have *opposite* signs. Since $f(x)$ is continuous and one root of it is guaranteed to lie between these two values, we say these a_0 and b_0 *bracket* the root. Then we proceed by iterations to produce a sequence of shrinking intervals $(a_0, b_0) \rightarrow (a_i, b_i)$ such that the shrunk intervals always contain one root of $f(x)$.



For convergence, it is necessary to have a *good* initial guess. This might be achieved by plotting $f(x)$ vs. x to get some idea of the root. Here we will deal with four methods for finding of roots of nonlinear equations,

1. Bisection method
2. False position (Regula falsi) method
3. Newton-Raphson method
4. Secant method

Bisection method

It is the simplest but relatively slow method of finding root of nonlinear equations. The method is guaranteed to converge to a root of $f(x)$ if the function is continuous in the interval $[a_0, b_0]$ where $f(a_0)$ and $f(b_0)$ have opposite signs. This is called Intermediate Value Theorem. But bracketing can go wrong if $f(x)$ has double roots or $f(x) = 0$ is an extrema or $f(x)$ has many roots over the interval chosen. The steps involve in bracketing are,

1. Choose a_0 and b_0 , where $a_0 < b_0$, and calculate $f(a_0)$ and $f(b_0)$.
2. If $f(a_0) * f(b_0) < 0$ then bracketing done. Proceed to execute bisection method.
3. If $f(a_0) * f(b_0) > 0$ i.e. same sign, then check whether $|f(a_0)| \leq |f(b_0)|$.
4. If $|f(a_0)| < |f(b_0)|$, shift a_0 further to the left by using, say, $a_0 \leftarrow a_0 - \beta * (b_0 - a_0)$ and then go back to second step. Choose your own β , say 1.5.
5. If $|f(a_0)| > |f(b_0)|$, shift b_0 further to the right by using, say, $b_0 \leftarrow b_0 + \beta * (b_0 - a_0)$ and then go back to second step. Choose your own β , say 1.5.
6. Give up, if you can't satisfy the condition $f(a) * f(b) < 0$ in 10 – 12 iterations. Start with a new pair $[a'_0, b'_0]$ and do the thing all over again.

Once the bracketing is accomplished, the bisection method proceeds as

1. Choose appropriate $[a_0, b_0]$, where $a_0 < b_0$, to bracket the root i.e. $f(a_0) * f(b_0) < 0$.
2. Bisect the interval, the midpoint of the interval is taken as first approximation with $a_1 = a_0$ and $b_1 = b_0$

$$c_1 = \frac{b_1 + a_1}{2} \quad (4)$$

The maximum absolute error of this approximation is

$$|c_1 - \bar{x}| \leq \frac{b_1 - a_1}{2} = \frac{b_0 - a_0}{2} \quad (5)$$

3. If the error in (5) is considered too large, repeat the above step with new interval either $[a_2, b_2] = [a_1, c_1]$ or $[c_1, b_1]$ depending on the sign of $f(c_1)$ i.e. whether $f(c_1) * f(b_1) < 0$ or $f(a_1) * f(c_1) < 0$. The new bisection or midpoint is $c_2 = (b_2 + a_2)/2$ and maximum absolute error is

$$|c_2 - \bar{x}| \leq \frac{b_2 - a_2}{2} = \frac{b_0 - a_0}{4}$$

4. If in the n -th step the corresponding values are a_n, b_n, c_n then

$$c_n = \frac{b_n + a_n}{2} \rightarrow |c_n - \bar{x}| \leq \frac{b_n - a_n}{2} = \frac{b_0 - a_0}{2^n} \quad (6)$$

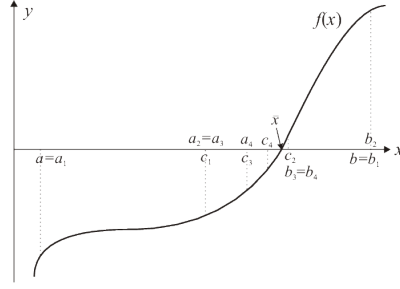
The method converges since in the limit $n \rightarrow \infty$ the factor $2^{-n} \rightarrow 0$, but as we see it converges rather slowly.

5. If our desired maximum error ϵ , say $\epsilon = 10^{-4}$, is reached, we stop

$$|c_n - \bar{x}| \leq \epsilon \Rightarrow \frac{b_0 - a_0}{2^n} \leq \epsilon \quad (7)$$

Along with the above convergence criteria, we can also test if $|f(c)| < \epsilon$ since at root x_0 implies $f(x_0) = 0$.

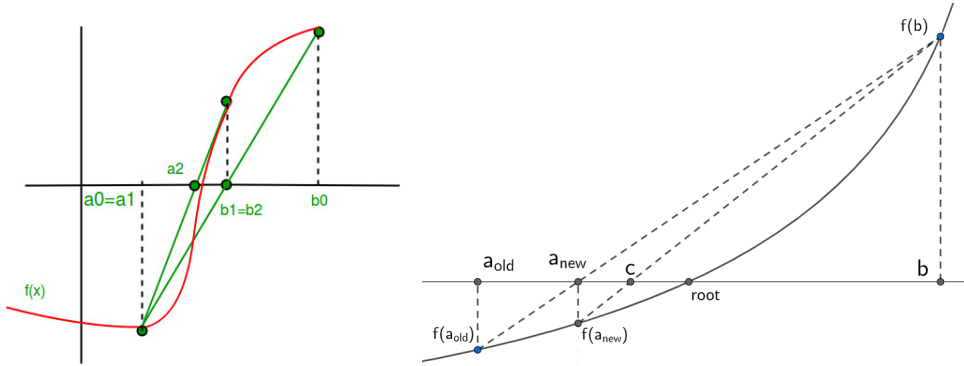
The bisection steps are schematically shown in the figure below.



This method has slowest convergence of all other root finding methods but it is a sure shot to root provided you can bracket properly. But still one cannot get root beyond certain precision because the difference between b and a is limited by floating point precision *i.e.* as the difference $(b_n - a_n)$ decreases. Therefore, the accuracy can never reach machine precision.

Regula falsi method

In Regula Falsi or False Position method, one does some sort of an interpolation to converge on a root faster than Bisection. The method involves determining the slope of the straight line joining $[a_0, b_0]$, which bracketed the root, and finding where this line crosses the abscissa (x -axis). Take that point as either new a_1 or b_1 depending on the relative sign of the functions at those points. One important difference over Bisection method is that the new starting point is directly determined by the function $f(x)$ under consideration apart from checking $f(a_0) * f(b_0) \leq 0$. Two possible scenario is schematically shown in the figure below.



The basic steps involved are

1. Choose appropriate $[a_0, b_0]$, where $a_0 < b_0$ and $f(a_0) * f(b_0) < 0$, to bracket the root. Details about bracketing the root have already been discussed before in Bisection method.
2. Calculate the slope of the straight line joining a_0 and b_0 and obtain c where the line crosses the abscissa *i.e.* $y(c) = 0$,

$$m = \frac{y(b_0) - y(a_0)}{b_0 - a_0} = \frac{y(b_0) - y(c_0)}{b_0 - c_0} \Rightarrow c_0 = b_0 - \frac{(b_0 - a_0) * y(b_0)}{y(b_0) - y(a_0)} \quad (8)$$

where $y(a) = f(a)$ and $y(b) = f(b)$. One can as well use $f(a)$ as reference point instead of $f(b)$, as is shown in the figure above. If the function $f(x)$ is convex or concave, as in the right figure, in the interval $[a, b]$ containing a root, then one of the points a or b is always fixed and the other

point varies with iterations. This sometime makes checking for convergence little tricky. Hence, after n -th step,

$$c_n = b_n - \frac{(b_n - a_n) * f(b_n)}{f(b_n) - f(a_n)} \quad (9)$$

3. If $f(a_n) * f(c_n) < 0$, then root lies to the left of c_n and in such case the new $b_{n+1} = c_n$ and $a_{n+1} = a_n$. If $|c_{n-1} - c_n| < \epsilon$ then c_n is the root implying $f(c_n) \approx 0$. Else iterate step 2 with new b .
4. If $f(a_n) * f(c_n) > 0$, then root lies to the right of c_n , therefore, the new $a_{n+1} = c_n$ and $b_{n+1} = b_n$. If $|c_{n-1} - c_n| < \epsilon$ then c_n is the root implying $f(c_n) \approx 0$. Else iterate step 2 with new a .

The Regula falsi always converges and has improved speed of convergence over Bisection, but the rate of convergence can sometimes drop below Bisection. Note that as a solution is approached, a and b will be very close to each other and subtraction in the denominator of (8) can lose significant digits. This method is slightly different than the *Secant method* where it retains the last two computed points that are obtained from the same formula as above.

Newton-Raphson method

The most famous, but not necessarily most efficient, of all root finding methods is Newton-Raphson. This works also for multivariate functions. Unlike the previous two methods, this one involves both $f(x)$ and its derivative $f'(x)$ but does not require bracketing. And finally, it converges quadratically, meaning near a root the number of significant (*i.e.* correct) digits approximately doubles with each step. The method is based on Taylor series expansion. To solve $f(x) = 0$, Taylor expand $f(x)$ at an initial guess x_0 for a root of $f(x)$,

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{1}{2!}(x - x_0)^2 f''(x_0) + \dots \quad (10)$$

If we are closer to the root, then $(x - x_0)^2 \approx 0$ and we can stop at $f'(x)$ term,

$$f(x) = f(x_0) + (x - x_0)f'(x_0) = 0 \Rightarrow x = x_0 - \frac{f(x_0)}{f'(x_0)} \quad (11)$$

then x is a better approximation of the root than x_0 . Far from a root, the higher derivative terms in the series become important. The approximation to the root can be improved iteratively to move from the x_0 towards the root,

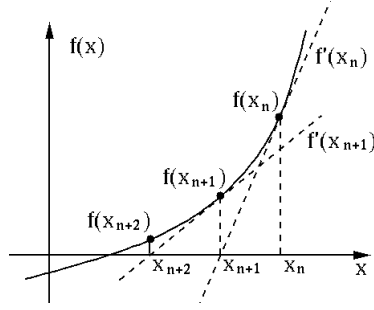
$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 1, 2, \dots \quad (12)$$

There is no bracketing here, just an initial guess x_0 but involved taking a derivative. This can be a problem unless we have an analytical expression for the derivative and cheaper to evaluate. When Newton-Raphson works it converges quadratically but that often is not the case. For a detailed discussion on convergence of Newton-Raphson see Wikipedia or textbooks including Numerical Recipes.

In place of using analytical expression for derivative $f'(x)$ of the function $f(x)$, one can approximate the derivative with finite difference, resulting in a variant of Newton's method called Secant method.

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad (13)$$

The symmetric derivative is preferred over forward ($[f(x+h) - f(x)]/h$) or backward ($[f(x) - f(x-h)]/h$) derivatives simply because it is $\mathcal{O}(h^3)$ improved. Therefore the use of the finite difference formula in (13) requires two initial guesses x_0 and x_1 corresponding to $x \pm h$.



The steps of Newton-Raphson method is fairly straight forward,

1. Make a good guess of x_0
2. Evaluate $f(x)$ and its derivative $f'(x)$ at $x = x_0$.
3. Continue using (12) to improve the estimate of the root until $|x_{n+1} - x_n| < \epsilon$.

When Newton's method works, it is really great. But to make it works crucially depends on $f(x)$ being smooth, $f'(x)$ to exist and the starting guess sufficiently accurate. Let x_* be the root of $f(x)$ and $\epsilon_n = x_* - x_n$ is the error in the n -th iteration. If x_0 is sufficiently close to x_* then

$$\epsilon_{n+1} = \frac{1}{2} \frac{f''(\xi)}{f'(x_n)} \epsilon_n^2 \Rightarrow \mathcal{O}(\epsilon_n^2) \quad (14)$$

for some point ξ between x_n and x_* . This is called quadratic convergence.

One way to get around taking derivative in Newton's method is to use *Secant method*. In this method, instead of one guess x_0 choose also a second one x_1 . Then the equation of the line through $(x_0, f(x_0))$ and $(x_1, f(x_1))$ is,

$$y(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0} (x - x_0) \quad (15)$$

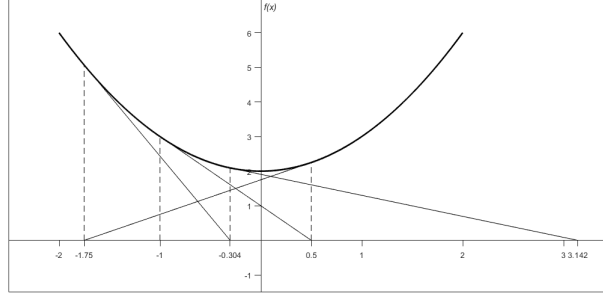
Now, the function $y(x)$ has a zero at x_2 where

$$x_2 = x_0 - \frac{x_1 - x_0}{f(x_1) - f(x_0)} f(x_0) \quad (16)$$

The x_2 is the next approximation to replace x_0 and the subsequent approximation x_3 is obtained by joining $(x_1, f(x_1))$ and $(x_2, f(x_2))$ and so on

$$\begin{aligned} x_3 &= x_1 - \frac{x_2 - x_1}{f(x_2) - f(x_1)} f(x_1) \\ &\vdots \\ x_{n+1} &= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \end{aligned} \quad (17)$$

This continues till $|x_{n+1} - x_n| < \epsilon$. This method avoids getting $f'(x_n) = 0$, *i.e.* divergence at inflection point in Newton's method as given in eqn. (12). It also avoids oscillations near local extrema as shown below in the figure.



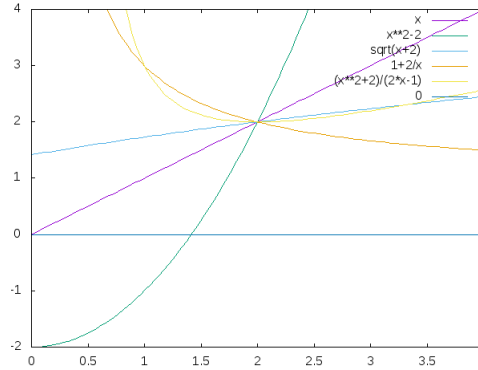
Fixed point method

Find root(s) of a nonlinear function $f(x)$ amounts to finding x_0 , where $f(x_0) = 0$, which is what we were trying above to do numerically. A *fixed point* of a function $g(x)$ is such that $x = g(x)$. In the fixed point iteration method the expression $f(x) = 0$ is replaced with the expression $x = g(x)$. If x_n is a fixed point of $g(x_n)$, then x_n is the root of $f(x)$. The iteration scheme is,

$$x_{n+1} = g(x_n) \quad (18)$$

An initial guess for the root x_1 is assumed and input as an argument for the function $g(x)$. The output $g(x_1)$ is then the estimate x_2 . The process is then iterated until the output $x_{n+1} \approx g(x_n)$. The iteration terminates when $|x_{n+1} - x_n| < \epsilon$. For a given $f(x) = 0$, there may be many equivalent fixed point problems with different choices for $g(x)$ (but not all choice converge equally),

$$f(x) = x^2 - x - 2 \rightarrow f(x) = 0 \Rightarrow x = g(x) = x^2 - 2, \quad \sqrt{x+2}, \quad 1 + \frac{2}{x}, \quad \frac{x^2 + 2}{2x - 1} \quad (19)$$



We extend the fixed point method to solve system of nonlinear equations

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2, \dots, x_n) \\ f_2(x_1, x_2, \dots, x_n) \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) \end{pmatrix} = 0 \quad \text{where} \quad \mathbf{x}^T = (x_1, x_2, \dots, x_n) \quad (20)$$

Since there is no concept of bracketing in higher dimension, a foolproof method like bisection that are guaranteed to converge does not exist. Hence, fixed point method is a fix to this problem. The fixed

point function $g(x)$ can be written as,

$$\left. \begin{array}{lcl} x_1 & = & g_1(x_1, x_2, \dots, x_n) \\ x_2 & = & g_2(x_1, x_2, \dots, x_n) \\ \vdots & & \\ x_n & = & g_n(x_1, x_2, \dots, x_n) \end{array} \right\} \Rightarrow \mathbf{x} = \mathbf{g}(\mathbf{x}) \Rightarrow \mathbf{x}_{k+1} = \mathbf{g}(\mathbf{x}_k) \quad (21)$$

The stopping criteria would be,

$$\frac{\|\mathbf{x}_{k+1} - \mathbf{x}_k\|}{\|\mathbf{x}_{k+1}\|} < \epsilon \quad \text{where,} \quad \|\mathbf{x}_k\|^2 = (x_1)_k^2 + (x_2)_k^2 + \dots + (x_n)_k^2 \quad (22)$$

For example, try solving the following

$$x_1^2 + x_1 x_2 = 10 \quad \text{and} \quad x_2 + 3x_1 x_2^2 = 57 \quad (23)$$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \mathbf{g}(\mathbf{x}) = \begin{pmatrix} g_1(x_1, x_2) \\ g_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} \frac{10-x_1^2}{x_2} \\ 57 - 3x_1 x_2^2 \end{pmatrix} \quad (24)$$

Using an initial guess of $x_1 = 1.5$, $x_2 = 3.5$ yields divergence!

$$\begin{aligned} x_1 &= \frac{10 - (1.5)^2}{3.5} = 2.21429, \quad x_2 = 57 - 3(2.21429)(3.5)^2 = -24.37516 \\ x_1 &= \frac{10 - (2.21429)^2}{-24.37516} = -0.20910, \quad x_2 = 57 - 3(-0.20910)(-24.37516)^2 = 429.709 \end{aligned} \quad (25)$$

$$\vdots \quad (26)$$

Continuing the iteration shows it is diverging in x_2 . A different choice of $g(x)$ works rather well,

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \sqrt{(10 - x_1 x_2)} \\ \sqrt{(57 - x_2)/3x_1} \end{pmatrix} \quad (27)$$

The same initial guess *i.e.* $x_1 = 1.5$, $x_2 = 3.5$ yields,

x_1	x_2	x'_1	x'_2	ϵ
1.5000	3.5000	2.1794	2.8605	0.259459
2.1794	2.8605	1.9405	3.0496	0.084286
1.9405	3.0496	2.0205	2.9834	0.028792
\vdots				
1.9999	3.0001	2.0000	3.0000	0.000046

Hence, the take home message is that the fixed point iteration method is not guaranteed to give a possible solution. The initial guess and the form chosen affect whether a solution can be obtained or not.

Newton-Raphson for multivariables

The Newton-Raphson for multivariable nonlinear equations enjoys similar level of popularity as with single variable. Assume the same set of nonlinear equations as in eqn. (20). Suppose after i -th iteration $(x_1, x_2, \dots, x_n) \rightarrow (x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)})$, then Taylor expansion of the first equation around

these is,

$$\begin{aligned}
f_1(x_1^{(i+1)}, x_2^{(i+1)}, \dots, x_n^{(i+1)}) &\approx f_1(x_1^{(i)}, x_2^{(i)}, \dots, x_n^{(i)}) + \\
&\frac{\partial f_1}{\partial x_1} \Big|_{x^{(i)}} (x_1^{(i+1)} - x_1^{(i)}) + \frac{\partial f_1}{\partial x_2} \Big|_{x^{(i)}} (x_2^{(i+1)} - x_2^{(i)}) + \\
&\dots + \frac{\partial f_1}{\partial x_n} \Big|_{x^{(i)}} (x_n^{(i+1)} - x_n^{(i)})
\end{aligned} \tag{28}$$

Similar Taylor expansion for f_2, f_3, \dots, f_n yields,

$$\mathbf{f}(\mathbf{x}^{(i+1)}) = \mathbf{f}(\mathbf{x}^{(i)}) + \mathbf{J}(\mathbf{x}^{(i)}) (\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}) = 0 \Rightarrow \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \mathbf{J}^{-1}(\mathbf{x}^{(i)}) \mathbf{f}(\mathbf{x}^{(i)}) \tag{29}$$

where $\mathbf{J}(\mathbf{x}^{(i)})$ is the *Jacobian matrix*

$$[\mathbf{J}(\mathbf{x}^{(i)})]_{pq} = \frac{\partial f_p(\mathbf{x})}{\partial x_q} \Big|_{\mathbf{x}^{(i)}} \tag{30}$$

Compare the above multivariate Newton's equation (29) with single variable Newton equation (12),

$$x^{(i+1)} = x^{(i)} - (f'(x^{(i)}))^{-1} f(x^{(i)}) \quad \text{and} \quad \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} - \mathbf{J}^{-1}(\mathbf{x}^{(i)}) \mathbf{f}(\mathbf{x}^{(i)}) \tag{31}$$

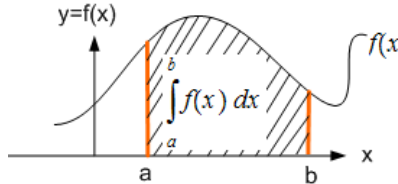
where $f'(x)$ is replaced with Jacobian $\mathbf{J}(\mathbf{x})$. If the Jacobian is invertible then the iteration converges to a solution. In practice, the Jacobian is usually not inverted. It is a two step method. A linear system of equations is solved first then iterate $\mathbf{x}^{(i)}$ in the second step,

$$\text{step I. } \mathbf{J}(\mathbf{x}^{(i)}) \mathbf{s}^{(i)} = -\mathbf{f}(\mathbf{x}^{(i)}) \quad \text{step II. } \mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \mathbf{s}^{(i)} \tag{32}$$

If the Jacobian matrix is not singular then the convergence of Newton's method for nonlinear multivariate systems is quadratic too. But numerical cost per iteration of Newton's method for dense problem in N dimension is substantial. Computing Jacobian costs N^2 function evaluations and solving linear system costs $\mathcal{O}(N^3)$ operations. The next best thing is *Broyden's method* in which an approximate Jacobian matrix is built through successive iterations instead of explicit evaluation of derivatives. Its numerical cost is less than Newton but quadratic convergence is not guaranteed.

A brief note on numerical integration

In numerical integration we try an approximate solution to a definite integral up to desired precision



$$\mathcal{I} = \int_a^b f(x) dx \tag{33}$$

where $f(x)$ is a (piecewise) continuous, well-behaved (smooth) function over the interval $[a, b]$ of our interest. The definite integration is obviously the area under the curve as shown. There can be various reasons for doing integration numerically

1. $f(x)$ may be known only at certain points,
2. $f(x)$ is known but analytical integration may be too difficult or at times impossible to carry out, and
3. the integration can be carried out analytically but numerically it may be far more easier to a given accuracy.

The numerical integration is often called as *numerical quadrature* or simply *quadrature* especially when it is applied to one dimensional integration *i.e.* integration over only one variable. At the most basic, the numerical integration amounts to evaluating the integrand at finite set of points (may or may not be equally spaced) bounded by the interval $[a, b]$ and doing a *weighted* sum of these values to approximate the integral.

$$\mathcal{I} = \int_a^b f(x) dx \approx \sum_{n=1}^N w(x_n) f(x_n) \equiv \mathcal{I}_N \quad (34)$$

where $w(x_n)$ is the weight function and N is the number of integration points when the integration limit is sliced up. Choice of N depends on the maximum error $|\mathcal{I} - \mathcal{I}_N| < \epsilon$ we desire. We will discuss more about the errors later.

The expression in (34) follows from approximating a function $p(x)$ with Lagrange interpolating polynomial, often written in the form

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n = f(x_0) \cdot L_0(x) + f(x_1) \cdot L_1(x) + \cdots + f(x_N) \cdot L_N(x) \quad (35)$$

where $l_n(x)$ are n -degree polynomials. Integrating the polynomial, we get,

$$\begin{aligned} \mathcal{I}_N &= \int_a^b p(x) dx = \int_a^b \sum_{n=0}^N f(x_n) \cdot L_n(x) dx \\ &= \sum_{n=0}^N \left(f(x_n) \cdot \int_a^b L_n(x) dx \right) \equiv \sum_{n=0}^N f(x_n) \cdot w(x_n) \end{aligned} \quad (36)$$

Here the following techniques of numerical integration will be explored –

- Midpoint rule
- Trapezoidal rule
- Simpson's rule

The underlying strategy in all these methods is to approximate the integrand $f(x)$ with a polynomial: $p(x) = a_0 + a_1x + a_2x^2 + \cdots$. For Midpoint we only use the constant term a_0 , for Trapezoidal upto the linear term a_1x and upto the quadratic term a_2x^2 for Simpson.

Midpoint method

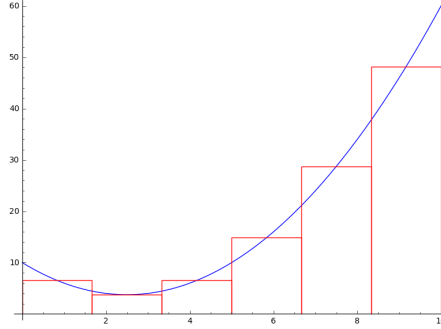
1. Let us divide the integration range $[a, b]$ in N equal parts of width h

$$h = (b - a)/N \quad (37)$$

2. Determine the midpoint of each intervals which are the integration points

$$x_1 = \frac{(a) + (a + h)}{2}, x_2 = \frac{(a + h) + (a + 2h)}{2}, x_3 = \frac{(a + 2h) + (a + 3h)}{2}, \dots \quad (38)$$

Hence, $f(x)$ is evaluated only at x_n and kept *constant* over the sub-interval and the assumption being the area of each rectangle evaluated at each integration point, $hf(x_n)$, approximates the area under the curve over that sub-interval.



3. The sum of all such rectangular area is

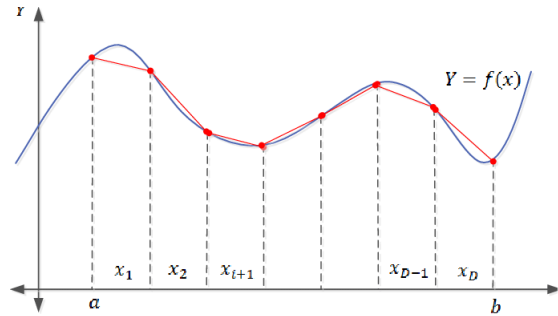
$$\mathcal{M}_N = \sum_{n=1}^N hf(x_n) \Rightarrow \lim_{N \rightarrow \infty} \mathcal{M}_N = \int_a^b f(x) dx \quad (39)$$

where the weight function $w(x_n) = w = 1$ or h for all x_n i.e. constant.

We will discuss later how to choose N so that the integration in (39) is accurate enough.

Trapezoidal method

The trapezoidal rule for estimating definite integrals uses straight lines connecting consecutive $f(x_n)$ to generate trapezoids whose areas are expected to better approximate the area under the curve over each interval.



1. As before in (37), we have N intervals each of width h . This h forms the width of each trapezoid.
2. Let the endpoints of each interval be at $x_0, x_1, x_2, \dots, x_N$ where

$$x_0 = a, x_1 = x_0 + h, x_2 = x_0 + 2h, \dots, x_n = x_0 + Nh = b \quad (40)$$

3. Evaluate the $f(x_n)$ and calculate the area of each trapezoid,

$$\mathcal{T}_n = \frac{h}{2} \left(f(x_{n-1}) + f(x_n) \right) \quad (41)$$

4. Sum over all the \mathcal{T}_n 's to approximate the integral

$$\begin{aligned} \mathcal{T}_N &= \sum_{n=1}^N \mathcal{T}_n = \frac{h}{2} \left([f(x_0) + f(x_1)] + [f(x_1) + f(x_2)] + \cdots + [f(x_{N-1}) + f(x_N)] \right) \\ &= \frac{h}{2} \left(f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{N-1}) + f(x_N) \right) \\ &= \sum_{n=1}^N w(x_n) f(x_n) \Rightarrow \lim_{N \rightarrow \infty} \mathcal{T}_N = \int_a^b f(x) dx \end{aligned} \quad (42)$$

where $w(x_0) = w(x_N) = 1$ or $h/2$ and $w(x_1) = w(x_2) = \cdots = w(x_{N-1}) = 2$ or h . Once again, we will come back to the accuracy question later. But contrary to the expectation, trapezoidal rule tends to be less accurate than the midpoint rule, particularly when the curve is strictly concave or convex over the integration limits.

Simpson's method

In Midpoint method the area under the curve is estimated by rectangles *i.e.* piecewise constant functions. In Trapezoidal rule the area is estimated by trapeziums *i.e.* piecewise linear functions. In the Simpson's rule we will approximate the curves in an interval with a quadratic functions $\propto x^2$. We, therefore, need three points to describe a quadratic curve – the obvious two are the (sub) interval boundaries and the other is taken to be the average of these two *i.e.* the midpoint.

$$\text{For } \int_{x_0}^{x_2} f(x) dx \text{ we need } (x_0, f(x_0)), (x_1, f(x_1)) \text{ and } (x_2, f(x_2)) \quad (43)$$

where $x_1 = (x_0 + x_2)/2$. The calculation of the integral above in (43) proceeds as,

$$\begin{aligned} \int_{x_0}^{x_2} f(x) dx &\approx \int_{x_0}^{x_2} \left(a_2 x^2 + a_1 x + a_0 \right) dx \\ &= \left(\frac{a_2}{3} x^3 + \frac{a_1}{2} x^2 + a_0 x \right) \Big|_{x_0}^{x_2} \\ &= \frac{a_2}{3} (x_2^3 - x_0^3) + \frac{a_1}{2} (x_2^2 - x_0^2) + a_0 (x_2 - x_0) \\ &= \frac{x_2 - x_0}{6} \left(2a_2 (x_2^2 + x_2 x_0 + x_0^2) + 3a_1 (x_2 + x_0) + 6a_0 \right) \end{aligned} \quad (44)$$

Now, let us take $h = (x_2 - x_0)/2$ and rearranging the terms using $x_1 = (x_2 + x_0)/2$ in (44) we get

$$\begin{aligned} \int_{x_0}^{x_2} f(x) dx &\approx \frac{h}{3} \left((a_2 x_2^2 + a_1 x_2 + a_0) + (a_2 x_0^2 + a_1 x_0 + a_0) + a_2 (x_2^2 + 2x_2 x_0 + x_0^2) \right. \\ &\quad \left. + 2a_1 (x_2 + x_0) + 4a_0 \right) \\ &= \frac{h}{3} \left(f(x_2) + f(x_0) + a_2 (2x_1)^2 + 2a_1 (2x_1) + 4a_0 \right) \\ &= \frac{h}{3} \left(f(x_2) + f(x_0) + 4(a_2 x_1^2 + a_1 x_1 + a_0) \right) \\ &= \frac{h}{3} \left(f(x_0) + 4f(x_1) + f(x_2) \right) \end{aligned} \quad (45)$$

Similarly for $\int_{x_2}^{x_4} f(x)dx = h(f(x_2) + 4f(x_3) + f(x_4))$ and so on. Therefore, in the final step we get the answer from Simpson's method as,

$$\begin{aligned}\mathcal{S}_N &= \frac{h}{3} \left(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 2f(x_{N-2}) + 4f(x_{N-1}) + f(x_N) \right) \\ &= \sum_{n=1}^N w(x_n) f(x_n) \Rightarrow \lim_{N \rightarrow \infty} \mathcal{S}_N = \int_a^b f(x) dx\end{aligned}\tag{46}$$

where, the weight functions are $w(x_0) = w(x_N) = 1$, for odd i the $w(x_n) = 4$ and for the even n the $w(x_i) = 2$. It is interesting to note that

$$\mathcal{S}_{2N} = \frac{2}{3} \mathcal{M}_N + \frac{1}{3} \mathcal{T}_N\tag{47}$$

Error estimate : Suppose we already know the actual answer of the integration \mathcal{I} in (33), then the absolute and relative errors with respect to that obtained numerically obviously are

$$\text{Absolute error: } |\mathcal{I}_N - I| \quad \text{and Relative error: } \left| \frac{\mathcal{I}_N - I}{I} \right| \times 100\%\tag{48}$$

where $\mathcal{I}_N \in \mathcal{M}_N, \mathcal{T}_N, \mathcal{S}_N$. But often, the very reason for doing numerical integration is that we do not have or cannot calculate an \mathcal{I} , and in that case all we can calculate is the *upper bound* of the error that each method will yield.

Let us estimate the possible error in Midpoint method by looking at a single interval (and using Mean Value Theorem),

$$\begin{aligned}\mathcal{I} - \mathcal{I}_N &= \int_{x_i}^{x_{i+1}} \left[f(x) dx - h f(c_i) \right] \quad \text{where } c_i = (x_{i+1} + x_i)/2 \\ &\approx \int_{x_i}^{x_{i+1}} \left[f(x) - f(c_i) \right] dx \\ &= \int_{x_i}^{x_{i+1}} \left[f(c_i) + (x - c_i) f'(c_i) + \frac{1}{2} f''(\xi_i) (x - c_i)^2 - f(c_i) \right] dx \\ &= \frac{1}{2} f''(\xi_i) \int_{x_i}^{x_{i+1}} (x - c_i)^2 dx = \frac{h^3}{24} f''(\xi_i) = \frac{h^3}{24} f''(\xi_i)\end{aligned}\tag{49}$$

where $\xi_i \in [x_i, x_{i+1}]$ for which f'' is maximum in the interval (since we are trying to get estimate of maximum error). Taking sum over all the N intervals,

$$\sum_{i=1}^N \frac{h^3}{24} f''(\xi_i) = \frac{h^2(b-a)}{24} \left(\frac{1}{N} \sum_{i=1}^N f''(\xi_i) \right) = \frac{(b-a)^3}{24N^2} f''(\xi)\tag{50}$$

where $f''(\xi)$ is an average of f'' . Similar argument can be used for trapezoidal and Simpson method to determine the upper bound of the error. Thus, if $f(x)$ is a continuous function over $[a, b]$, having a second derivative $f''(x)$ for Midpoint and Trapezoidal and fourth derivative $f''''(x)$ for Simpson over this interval, then the estimated upper bounds for the error in using numerical integration schemes to

estimate $\int_a^b f(x) dx$ are

$$\text{Midpoint : Error in } M_N \leq \frac{(b-a)^3}{24N^2} |f''(x)|_{\max} \quad (51)$$

$$\text{Trapezoidal : Error in } T_N \leq \frac{(b-a)^3}{12N^2} |f''(x)|_{\max} \quad (52)$$

$$\text{Simpson : Error in } S_N \leq \frac{(b-a)^5}{180N^4} |f'''(x)|_{\max} \quad (53)$$

The requirement of upper bound for the error determines N . Two things to keep in mind – (i) N chosen ought to be the smallest integer value greater than or equal to N (basically $\text{ceil}(N)$), (ii) the actual estimate may, in fact, be much better than is indicated by the error upper bound (usually they are). It may sound strange, but it is often true, that taking N larger than what one obtains from the error bounds can actually deteriorate the accuracy, contrary to what the limit $N \rightarrow \infty$ suggests.

What if $f''(x) = 0$? It, however, does not mean the error is zero but simply indicates that estimate of error upper bound can not be made.

Consider an example – try to estimate numerically the following integral whose exact analytical answer is available,

$$\int_0^1 x^2 dx = \frac{1}{3} = 0.333 \quad (54)$$

Without worrying about error upper bound, let us divide up the integration limit $[0, 1]$ in $N = 4$ intervals – $[0, 1/4]$, $[1/4, 2/4]$, $[2/4, 3/4]$ and $[3/4, 1]$, the interval length being $h = 1/4$. For Midpoint method, we need the midpoints of these subintervals: $1/8, 3/8, 5/8, 7/8$. Hence, from (39),

$$\begin{aligned} M_4 &= \frac{1}{4} \left[f\left(\frac{1}{8}\right) + f\left(\frac{3}{8}\right) + f\left(\frac{5}{8}\right) + f\left(\frac{7}{8}\right) \right] \\ &= \frac{1}{4} \left[\left(\frac{1}{8}\right)^2 + \left(\frac{3}{8}\right)^2 + \left(\frac{5}{8}\right)^2 + \left(\frac{7}{8}\right)^2 \right] = \frac{21}{64} = 0.328 \end{aligned} \quad (55)$$

The absolute error is $|0.333 - 0.328| \approx 0.0052$ and relative error is 1.56%. Let us now make use of (51) to determine N , first the $|f''(x)|_{\max}$,

$$f(x) = x^2 \Rightarrow f'(x) = 2x \Rightarrow f''(x) = 2 \Rightarrow |f''(x)|_{\max} = 2 \quad (56)$$

So, N required for maximum error bound of 0.001 is

$$0.001 = \frac{(1-0)^3}{24N^2} 2 \Rightarrow N = 9. \quad (57)$$

Funnily, Trapezoidal method for $N = 4$ yields 0.3437 and thus the absolute and relative errors are 0.104 and 3.12% respectively. And for maximum error bound of 0.001, Trapezoidal rule requires $N = 13$.

Gaussian quadrature

The errors for Midpoint and Trapezoidal methods are $\mathcal{O}(h^3)$ whereas for Simpson it is $\mathcal{O}(h^5)$. The above methods are collectively called *Newton-Cotes* method whose general form is integrating the

Lagrange polynomial that interpolates the integrand $f(x)$ at N equally spaced points,

$$\mathcal{I} = \int_a^b f(x) dx \approx \sum_{i=1}^N w(x_i) f(x_i)$$

$$\text{where } w(x_n) = \int_a^b L_n(x) dx \text{ with } k\text{-point } L_n(x) = \prod_{k=1, k \neq n}^N \frac{x - x_k}{x_n - x_k} \quad (58)$$

But suppose the integration points are not necessarily equally spaced. In that case we have $2n$ free parameters, the weights $w(x_n)$ and the interval boundaries x_n . *Gaussian quadrature* rule says that the optimal abscissas x_n of the n -point Gaussian quadrature formulas are precisely the roots of the orthogonal polynomial for the same interval and weight functions. It is optimal since it fits all polynomials up to degree $2n - 1$ exactly defined by the $2n$ parameters.

Gaussian quadrature is usually much faster and very accurate than the above three *closed* Newton-Cotes methods, namely Midpoint, Trapezoidal and Simpson. It allows to carry out integration,

$$\int_{-1}^1 f(x) dx \text{ or } \int_a^b F(t) dt \text{ where, } x = \frac{2t - a - b}{b - a} \quad (59)$$

Without proving, we state that on the integration interval $[-1, 1]$ to achieve maximum degree of *exactness* $2n - 1$ with n points and n weights one chooses the points x_0, x_2, \dots, x_n to be the roots of the degree- n **Legendre polynomial** $P_n(x)$ and the weights then are given by the equation (58). From the expression in eqn. (35),

$$p(x) = \sum_{k=1}^N f(x_k) w(x_k) \text{ where } x_k = \text{zeros of } P_k(x) \quad (60)$$

where x_k are zeros of Legendre polynomial of order n .

Hence, the working principle is,

1. Find the Legendre polynomial $P_n(x)$. The first few are

$$P_{n+1}(x) = \frac{2n+1}{n+1} x P_n(x) - \frac{n}{n+1} P_{n-1}(x) \text{ where, } P_0(x) = 1, P_1(x) = x \quad (61)$$

$$P_2(x) = \frac{3}{2} \left(x^2 - \frac{1}{3} \right), P_3(x) = \frac{5}{2} \left(x^3 - \frac{3}{5} x \right), P_4(x) = \frac{1}{8} (35x^4 - 30x^2 + 3) \quad (62)$$

2. Find the roots of $P_n(x)$ in $[-1, 1]$ and those values are the integration points x_n

$$P_2 : x_2 = \pm \frac{1}{\sqrt{3}}, P_3 : x_3 = 0, \pm \sqrt{\frac{3}{5}}, P_4 : x_4 = \pm \sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}, \pm \sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}} \quad (63)$$

3. Find the Lagrange polynomial that interpolates the integrand $f(x)$ at $x_1, x_2, x_3, \dots, x_N$.

4. Integrate the Lagrange polynomial to determine the weights w_1, w_2, \dots, w_N ,

$$n = 2 : w_2 = 1, 1 \quad n = 3 : w_3 = \frac{8}{9}, \frac{5}{9}, \frac{5}{9} \quad n = 4 : w_4 = \frac{18 + \sqrt{30}}{36}, \frac{18 - \sqrt{30}}{36} \quad (64)$$

where the weights for w_4 are repeated twice.

5. Find the integral by summing over as in (60).

As an example, let us try the following integration whose analytical result is available for comparison,

$$\int_{-1}^1 x e^x dx = \frac{2}{e} = 0.735\ 758\ 88 \quad (65)$$

We compare the convergence of Gaussian quadrature with Simpson against the steps taken to achieve the same level of accuracy,

Simpson		Gaussian Q	
N	\mathcal{I}_S	N	\mathcal{I}_G
20	0.735 764 50	4	0.735 756 50
40	0.735 759 23	5	0.735 758 87
60	0.735 758 95	6	0.735 758 88
80	0.735 758 90		
100	0.735 758 88		
120	0.735 758 88		

The above table resoundingly establish the advantage of Gaussian quadrature over Newton-Cotes equal spaced methods. The Gauss-Legendre quadrature is not the only quadrature for \int_{-1}^{+1} integrals, Jacobi and Chebyshev polynomials can be used as well. For \int_0^∞ integrals Laguerre and $\int_{-\infty}^\infty$ integrals Hermite are used.

Monte Carlo integration

We briefly discuss here Monte Carlo integration technique. Monte Carlo method is a class of computational algorithm that involves repeated random sampling to estimate an integration numerically. It is a widely used method to estimate particularly higher dimensional integrals. The Monte Carlo integration starts with choosing random numbers X_i with a *probability distribution function* (PDF) $p(x)$. Suppose the *domain* of X is discrete, as in tossing of coin $X \in \{h, t\}$ or rolling of a dice $X \in \{1, 2, 3, 4, 5, 6\}$. Then the $p(x)$ gives the probability or relative frequency with which a particular X occurs, $p(x) = \text{Prob}(X = x)$. For a continuous domain, however, we consider $p(x) dx$ to be the probability for X to assume any value within an interval dx around x ($[x \pm dx]$). Two important properties of PDF $p(x)$ are

$$0 \leq p(x) \leq 1 \quad \text{and} \quad \sum_{x_i \in D} p(x_i) = 1 \quad \text{or} \quad \int_D p(x) dx = 1 \quad (66)$$

where D is the domain of x . As an example, consider $p(x) = \text{constant}$ *i.e.* uniformly distributed over a domain $D = [a, b]$

$$\begin{aligned} \int_a^b p(x) dx &= \int_a^b \mathcal{C} dx = 1 \\ \Rightarrow p(x) &= \mathcal{C} = \frac{1}{b-a} \end{aligned} \quad (67)$$

Another widely used PDF is Gaussian distribution $\mathcal{N}(\mu, \sigma)$,

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (68)$$

Suppose $f(x)$ is a function on the, say discrete, domain of X whose PDF is $p(x)$ and evaluated over M random X , then

$$\langle f \rangle = \frac{1}{M} \sum_{i=1}^M f(x_i) p(x_i) \quad (69)$$

$$\sigma_f^2 = \frac{1}{M} \sum_{i=1}^M \left(f(x_i) - \langle f \rangle \right)^2 p(x_i) = \langle f^2 \rangle - \langle f \rangle^2 \quad (70)$$

Suppose we assemble N such independent $\langle f \rangle$ and their corresponding σ_f , then the *global* average and variance will be

$$\langle \langle f \rangle \rangle = \frac{1}{N} \sum_{i=1}^N \langle f \rangle_i \quad \text{and} \quad \sigma_N^2 = \frac{\sigma_f^2}{N} \Rightarrow \sigma_N \sim \frac{1}{\sqrt{N}} \quad (71)$$

The error on the measurement of f thus decreases as $1/\sqrt{N}$. So if we want Monte Carlo estimate of $\int_a^b f(x)dx$, then the method is certainly at disadvantage when compared to, say, Trapezoidal or Simpson where errors fall as $1/N^2$ and $1/N^4$ respectively. But it is true for one or fewer dimensions, as we move to higher dimensions (*i.e.* large number of variables) Monte Carlo becomes significantly efficient.

In Monte Carlo integration, we wish to estimate the integral $\int_a^b f(x)dx$ (remember, in MC we cannot calculate but estimate) and for this we define an *estimator*. Given a random variable X drawn from a PDF $p(x)$, then the estimator is defined as

$$\mathcal{F}_N \equiv \frac{1}{N} \sum_{i=1}^N \frac{f(X_i)}{p(X_i)} \quad (72)$$

Then the average of the estimator \mathcal{F}_N is

$$\langle \mathcal{F}_N \rangle = \int_a^b \mathcal{F}_N p(x) dx = \int_a^b f(x) dx \quad (73)$$

with the variance as in (70). For our particular case of uniform PDF in $[a, b]$,

$$\mathcal{F}_N = \frac{b-a}{N} \sum_{i=1}^N f(X_i) \quad (74)$$

The above expression (74) looks very similar to Midpoint expression (39). Therefore, the steps involve in Monte Carlo integration methods are,

1. Choose a N , say 10 or 20 or 50 or whatever.
2. Draw N number of random variables X_i from its domain $[a, b]$. Usually the in-built random numbers in any language return uniform random numbers in the range $[0, 1]$. To convert it to $[a, b]$ one may use

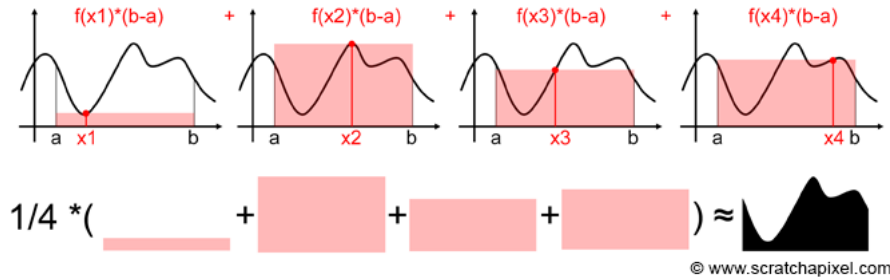
$$X = a + (b - a)\xi \quad \text{where, } \xi \in [0, 1]$$

3. For each X_i calculate $f(X_i)$ and determine \mathcal{F}_N using eqn. (74) and σ_f using

$$\sigma_f^2 = \frac{1}{N} \sum_{i=1}^N f(X_i)^2 - \left(\frac{1}{N} \sum_{i=1}^N f(X_i) \right)^2 \quad (75)$$

4. Either tabulate or plot \mathcal{F}_N versus N . Also keep track of σ_f for each N .
5. Go to step (1), increase N by 10 or whatever times and repeat the above cycle.

Schematically the monte carlo integration looks very much like the following figure

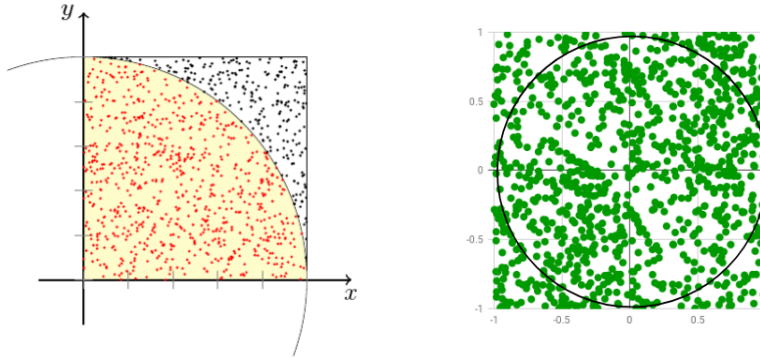


where we chose $N = 4$ random numbers $\{x_1, x_2, x_3, x_4\}$ and calculated the area $(b - a) * f(x_i)$ to estimate $\int_a^b f(x) dx$. As N becomes larger and larger, you will find \mathcal{F}_N converging to a value but decrease in error or σ_f will be rather slow ($\sim 1/\sqrt{N}$).

A popular way to demonstrate the application of monte carlo is to evaluate the integral

$$\int_0^1 \sqrt{1-x^2} dx = \frac{\pi}{4} \quad (76)$$

which return an estimate for π . The method is straight forward – generate uniformly distributed random number $\in [0, 1]$ *i.e.* within a unit square and count how many of them falls within quarter circle. The situation is shown in the left-hand figure below. From the four times the ratio of those inside and outside gives the estimate of π .



One can also use a full unit circle in a unit square, then the radius of the circle is $1/2$ and area is $\pi/4$. But in this case we need random numbers for x and y coordinates $\in [-0.5, 0.5]$ and check for $x^2 + y^2 \leq 1$. This situation is depicted in the right-hand figure above.