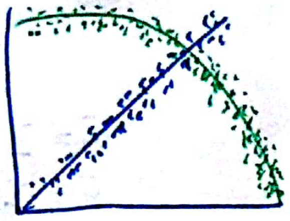
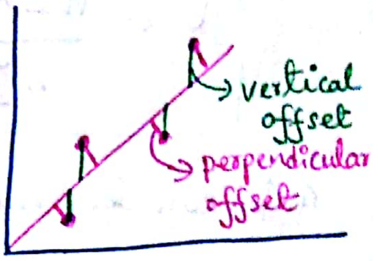
Least square fitting



To find best fitting curve to a set of point is by minimizing the sum of the squares of the offsets (residuals) of the points from the curve. As it is squared, function is continuous differentiable.



Vertical offsets from a line, surface is computed and not perpendicular offset, as former provides a fitting function for the independent variable x to estimate $y = f(x)$, easy to implement along x or y axis, & also provides simpler analytic form for fit parameters.

linear least square fit / linear regression, formulated by Gauss & Legendre solves for a straight line best fit. This also works good for simple non-linear function like log, exp, power law as one can transform to linear, e.g. $T = 2\pi\sqrt{\frac{l}{g}}$ for simple pendulum, fit T vs. $\sqrt{l}$ which is a straight line.

Vertical least square fit of n data point $R^2 = \sum_{i=1}^{n}[y_i - f(x_i, a_1, a_2, \ldots, a_n)]^2$

R^2 to minimum, $\frac{\partial R^2}{\partial a_i} = 0 \in i = 1, \ldots, n.$

for linear fit $f(\overset{x_i}{a}, b) = a + bx_i$, So $R^2(a,b) = \sum_{i=1}^{n}[y_i - (a+bx_i)]^2$

$$\frac{\partial R^2}{\partial a} = -2\sum_{i=1}^{n}[y_i - (a+bx_i)] = 0 \text{ or, } na + b\sum_{i=1}^{n}x_i = \sum_{i=1}^{n}y_i$$

$$\frac{\partial R^2}{\partial b} = -2\sum_{i=1}^{n}[y_i - (a+bx_i)]x_i = 0 \text{ or, } a\sum_{i=1}^{n}x_i + b\sum_{i=1}^{n}x_i^2 = \sum_{i=1}^{n}x_iy_i$$

In matrix form, $\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix}\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}$

$$\text{so } \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}^{-1}\begin{pmatrix} \sum y_i \\ \sum x_i y_i \end{pmatrix}.$$

$$= \frac{1}{n\sum x_i^2 - (\sum x_i)^2} \begin{pmatrix} \sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i \\ n\sum x_i y_i - \sum x_i \sum y_i \end{pmatrix}.$$

So, $$\boxed{\begin{aligned} a &= \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n\sum x_i^2 - (\sum x_i)^2} \\ b &= \frac{n\sum x_i y_i - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \end{aligned}}$$ "Regression coefficients"

$\bar{x} = \sum x_i$
$\bar{y} = \sum y_i$

This can be rewritten in simpler form by defining the sum of squares

$$S_{xx} = \sum(x_i - \bar{x})^2 = \sum x_i^2 - 2n\bar{x}^2 + n\bar{x}^2 = \sum x_i^2 - n\bar{x}^2 = n\sigma_x^2$$

$$S_{yy} = \sum(y_i - \bar{y})^2 = \sum y_i^2 - n\bar{y}^2 = n\sigma_y^2$$

$$S_{xy} = \sum(x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n\bar{x}\,\bar{y}. = n\,\mathrm{cov}(x,y)$$

σ_x^2, σ_y^2 = variance, $\mathrm{cov}(x,y)$ = covariance. So,

$$b = \frac{\mathrm{cov}(x,y)}{\sigma_x^2} = \frac{S_{xy}}{S_{xx}}, \quad a = \bar{y} - b\bar{x} = \bar{y} - \frac{S_{xy}}{S_{xx}}\bar{x}$$

The quality of the fit is parametrized in terms of correlation coefficient $r^2 = \frac{S_{xy}^2}{S_{xx}\,S_{yy}}$. If $\hat{y}_i$ is the vertical coordinate of the best fit line with coordinate x_i, $\hat{y}_i = a + bx_i$, then error between actual vertical point y_i & fitted point is $e_i = y_i - \hat{y}_i$, so that variance of e_i is defined as

$$s^2 = \sum_{i=1}^{n} \frac{e_i^2}{n-2}, \quad s = \sqrt{\frac{S_{yy} - bS_{xy}}{n-2}} = \sqrt{\frac{S_{yy} - S_{xy}^2/S_{xx}}{n-2}}$$

So that standard error for a and b is

$$a_{SE} = s\sqrt{\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}}}, \quad b_{SE} = \frac{s}{\sqrt{S_{xx}}}.$$

Goodness of fit is calculated from coefficient of determination $R^2 = 1 - \frac{S_{residue}}{S_{total}}$

$$S_{residue} = \sum_{i=1}^{n} e_i^2, \quad S_{total} = (n-1)\sigma_y^2$$

Root finding using Bisection & Newton-Raphson Method

For a linear equation in one-dimension $f(x)=0$, we can find a desired root, or for a set of linear equation, implicit function theorem we can solve N equations with N unknowns simultaneously

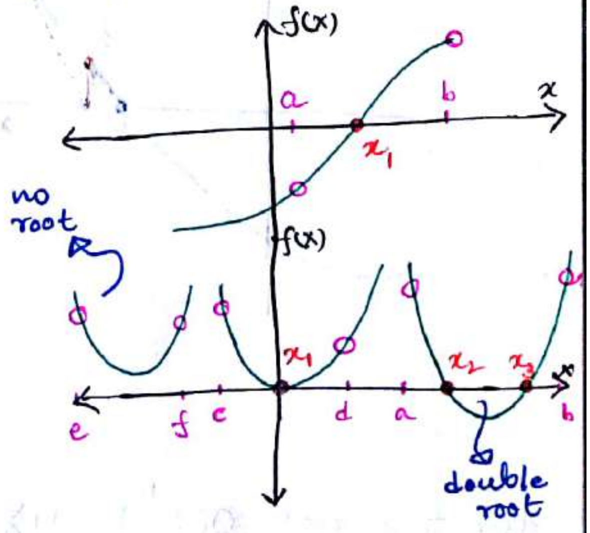
$$f(\vec{x}) = \vec{0}.$$

In one-dimension a root canbe found by trapping ("bracketing") within an interval. In multidimension, its impossible to guarantee a root by this and if equations are non-linear, then equations may or maynot have a root. In all dimensions, except in linear problems, root finding proceeds via iteration from some approximate trial solution towards convergence. Sign change of function is a precursor to hit a root, hence to choose the bracket, but multiple roots is a problem. for instance, if minima found is exactly zero, then you've found a "double root."

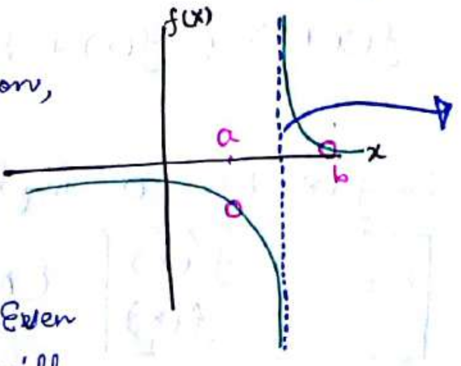
In 1D, Brent's method is best choice, when $f'(x)$ is hard to find. Ridders method is also good. If $f'(x)$ can beeasily computed then Newton-Raphson is best choice, even when in multidimension.

Bisection Method

Intermediate value theorem says if the function is continuous, then at least one root must be there within the bracketed interval (a,b) with $f(a)$ & $f(b)$ have opposite signs.



For a bounded distincontinuous function, say $f(x) = \frac{1}{x-c}$, a step discontinuity that crosses zero cannot be treated as a root. Even if Bisection method will converge to $x=c$, but $|f(x)| \to \infty$



To choice a bracket is a hard call, for example, for $f(x) = 3x^2 + \frac{\ln[(\pi - x)^2]}{\pi^4} + 1$ is well behaved except at $x = \pi$ and dips below zero in the interval $x = \pi \pm 10^{-667}$.

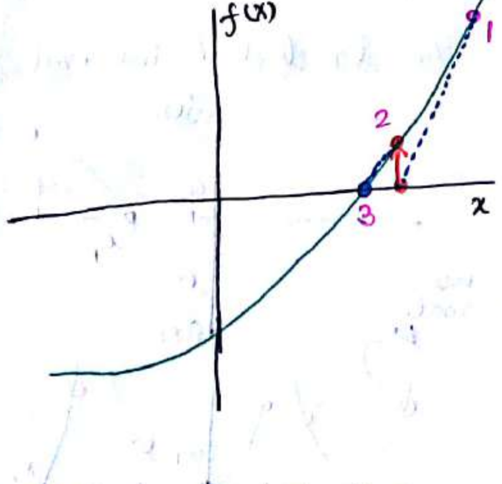
In Bisection : (i) Evaluate $f(x)$ at interval's midpoint and examine its sign, (ii) Use the midpoint to replace whichever limit has the same sign. After each iteration, bracket containing the root decrease by a factor of two. If after n iterations, bracket size is ϵ_n, then in next iteration bracket size will be $\epsilon_{n+1} = \epsilon_n/2$.

So # of iterations required to achieve given tolerance

$$n = \log_2 \frac{\epsilon_0}{\epsilon}, \quad \epsilon_0 = \text{initial size of bracket}, \ \epsilon = \text{tolerance.}$$

If more than 1 root, then Bisection will find only one. For a smooth function. Secant (or false position) method is faster than Bisection method.

Newton - Raphson method Find out $f(x)$ and $f'(x)$ at arbitrary x. Geometrically, a tangent drawn at x cuts x-axis (abscissa) then setting next guess x_{i+1} to the abscissa. So NR-method extrapolates the local derivative to find the next estimate of the root. Algebra-ically, using Taylor series,



$$f(x+\delta) \approx f(x) + \delta f'(x) + \frac{\delta^2}{2!} f''(x) + \cdots$$

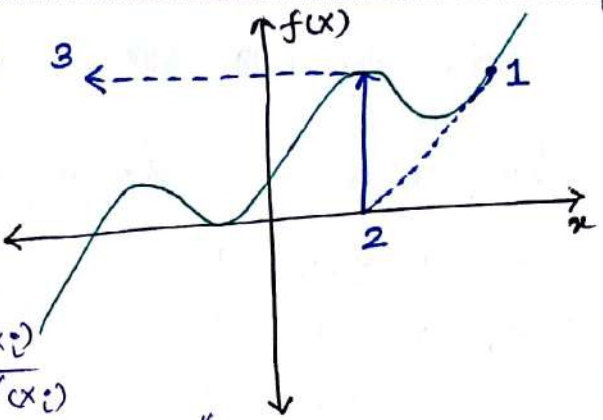
(nonlinear are unimportant)

So $f(x+\delta) = 0$ implies

$$\boxed{\delta = -\frac{f'(x)}{f(x)}} \quad \text{(Near the root)}$$

far from root $O(\delta^n)$, $n \geq 2$ are important, so NR method is inaccurate. If there is a local extrema, then NR method fails because $f'(x) = 0$.

Newton-Raphson converges quadratically as,



$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \frac{\epsilon^2}{2!} f''(x) + \cdots$$

$$f'(x+\epsilon) = f'(x) + \epsilon f''(x) + \cdots$$

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

we can write, $\epsilon_{i+1} = \epsilon_i - \frac{f(x_i)}{f'(x_i)} = -\epsilon_i^2 \frac{f''(x)}{2f'(x)}$ by recurrence relation.

So near a root, number of significant digits doubles with each step.

Computation of $f'(x)$ numerically using finite-difference has its pros and cons.

Lagrange Interpolation

In polynomial interpolation, Lagrange polynomial is the polynomial of lowest degree, so that for a set of data points $(x_1, y_1 = f(x_1))$ $(x_2, y_2 = f(x_2)), \cdots, (x_n, y_n = f(x_n))$ with no two x_i are identical,

$$p(x) = \sum_{j=0}^{n} y_j l_j(x) \quad \text{(linear combination)}$$

where Lagrange basis polynomials $l_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{n} \frac{x - x_k}{x_j - x_k}$

~~$\ell_j(x)$~~ = So interpolating polynomial $P(x)$ is

$$P(x) = \sum_{j=0}^{n} \frac{x-x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_n}{x_j - x_n} y_j$$

$x_j \neq x_k$ for well-defined function.

Now $$l_{j\neq i}(x_i) = \prod_{k\neq j} \frac{x_i - x_k}{x_j - x_k} = \frac{x_i - x_0}{x_j - x_0} \cdots \frac{\overset{=0}{\cancel{x_i} - \cancel{x_i}}}{x_j - x_i} \cdots \frac{x_i - x_n}{x_n - x_k} = 0$$

$l_i(x_i) = \prod_{k\neq i} \frac{x_i - x_k}{x_i - x_k} = 1$. So all basis polynomials are zero except $x = x_i$; $l_i(x) = 1$ because it doesn't have $(x - x_i)$ term.

So $l_j(x_i) = \delta_{ij}$, so that $P(x_i) = \sum_{j=0}^{n} y_j \delta_{ij} = y_i$

Suppose we want to interpolate $f(x) = x^3$ within bracket $1 \le x \le 3$.

$x_0 = 1,\ f(x_0) = 1$
$x_1 = 2,\ f(x_1) = 8$, So $P(x) = 1 \frac{x-2}{1-2}\frac{x-3}{1-3} + 8\frac{x-1}{2-1}\frac{x-3}{2-3} +$
$x_2 = 3,\ f(x_2) = 27$ $\quad 27\frac{x-1}{3-1}\frac{x-2}{3-2}$

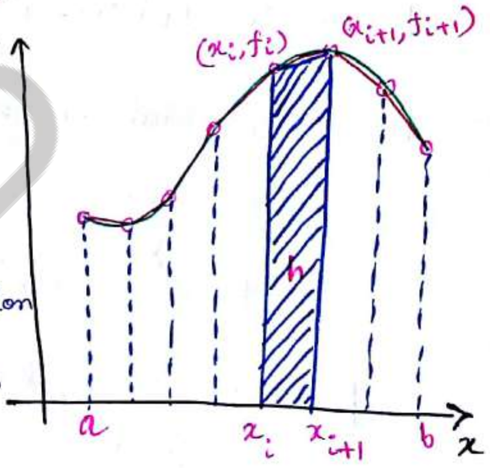
$= 6x^2 + 6 - 11x.$

Polynomial interpolation (Lagrange) is susceptible to Runge Phenomena of large oscillation. Commonly cubic spline or trigonometric interpolation is used.

Integration of functions - Quadrature

Newton - Leibnitz formula

$$I = \int_a^b f(x)\,dx = F(b) - F(a),$$



$F(x)$ = primitive antiderivative. is often hard to find analytically. Numerical integration of bounded integral in 1D is called quadrature and in higher dimension, called cubature.

Geometrically $\int_a^b f(x)\,dx$ is the area within $y = f(x)$ and lines $x=a$, $x=b$ and x-axis. For uniform sampling within $[a,b]$, equidistant points are $x_i = a + (i-1)h$, $i = 1, 2, \dots, n$. with $h = \frac{b-a}{n-1}$.

Trapezoidal Rule Replace the graph of the integrand by the polygonal line defined by a finite number of integrand values & then sum the formed trapezoidal area.

$$\int_a^b f(x)\,dx \approx \frac{h}{2}(f_1 + f_2) + \dots + \frac{h}{2}(f_i + f_{i+1}) + \dots + \frac{h}{2}(f_{n-1} + f_n)$$

$$\simeq h\left[\frac{f_1 + f_n}{2} + \sum_{i=2}^{n-1} f_i\right]$$

$\lim_{h\to 0} \int_a^b f(x)\,dx$ is exact (Riemann integral)

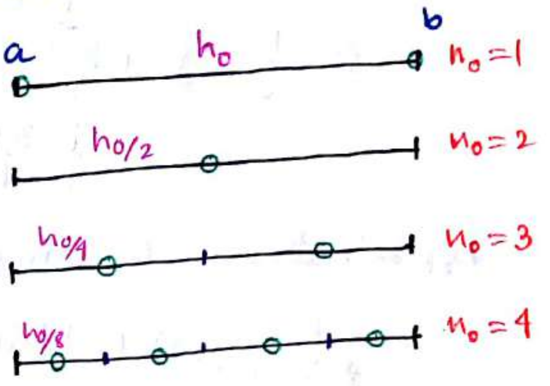
Trapezoidal rule cumulates error $O(h^2)$. We'll use step-halving technique for adaptive control of the integration mesh, which is (i) calculate integral for given h (n - node), (ii) compare result for $h/2$ (2n-1) node. until relative difference drops under a tolerance.

First 3 approximations by trapezoidal rule

$$S_0 = \frac{h_0}{2}[f(a)+f(b)]; \quad h_0 = b-a$$

$$S_1 = \frac{h_0}{4}\left[f(a)+f(b)+2f\left(a+\frac{h_0}{2}\right)\right]$$

$$S_2 = \frac{h_0}{8}\left[f(a)+f(b)+2f\left(a+\frac{h_0}{4}\right)+ 2f\left(a+\frac{h_0}{2}\right)+2f\left(a+\frac{3h_0}{4}\right)\right]$$



Simpson's 1/3rd rule

It can be shown that Trapezoidal & Simpson's rule can be obtained from a reductionist approach to Newton-Cotes quadrature formula that says $\int^b f(x)\,dx \approx (b-a)\sum_{i=1}^{n} H_i f_i$ where Cotes coefficient

$$H_i = \frac{\int_0^{n-1} \prod_{j\neq i}^{n} [q-(j-1)]\,dq}{(-1)^{n-i}(i-1)!\,(n-i)!\,(n-1)}, \quad i = 1, 2, \dots, n. \text{ with } \sum_{i=1}^{n} H_i = 1$$

and $H_i = H_{n-i+1}$.

For odd-number of mesh point $n=3$, $H_1 = \frac{1}{6}$, $H_2 = \frac{2}{3}$, $H_3 = \frac{1}{6}$.

and $b-a = x_3 - x_1 = 2h$, Simpson's formula :

$$\int_{x_1}^{x_3} f(x)\,dx \approx \frac{h}{3}(f_1 + 4f_2 + f_3).$$

Geometrically, we replace $y = f(x)$ by the parabola $y = P_2(x)$ with the Lagrange polynomial $P_2(x)$ defined by the 3 points (x_1, f_1), (x_2, f_2) and (x_3, f_3).

Similar to Trapezoidal rule, using additive property of integrals for subintervals, divide interval $[a,b]$ by odd number $n = 2m+1$

of equally spaced point $x_i = a+(i-1)h$, $i=1,2,\dots,n$ with $h = \frac{b-a}{n-1} = \frac{b-a}{2m}$. So now Simpson's formula can be applied as

$$\int_a^b f(x)dx \approx \frac{h}{3}(f_1+4f_2+f_3) + \frac{h}{3}(f_3+4f_4+f_5) + \dots + \frac{h}{3}(f_{n-4}+4f_{n-3}+f_{n-2}) + \frac{h}{3}(f_{n-2}+4f_{n-1}+f_n)$$

$$\approx \frac{h}{3}\Big(f_1 + 2\sum_{\substack{i=3,5,\dots \\ (\text{odd})}}^{n-2} f_i + 4\sum_{\substack{i=2,4,\dots \\ (\text{even})}}^{n-1} f_i + f_n\Big).$$

Approximations in Simpson's rule,

$$S_1 = \frac{h_1}{3}[f(a) + 4f(a+h_1) + f(b)], \quad h_1 = \frac{h_0}{2} = \frac{b-a}{2}$$

$$S_2 = \frac{h_2}{3}[f(a) + 4f(a+h_2) + 2f(a+2h_2) + 4f(a+3h_2) + f(b)], \quad \left\{h_2 = \frac{h_1}{2} = \frac{h_0}{4}\right\}$$

It's easy to derive, $S_1^{\text{simpson}} = \left.\frac{4S_1 - S_0}{3}\right|_{\text{trapezoidal}}$, $S_2^{\text{simpson}} = \left.\frac{4S_2 - S_1}{3}\right|_{\text{trapezoidal}}$

so that $S_k^{\text{simpson}} = \left.\frac{4S_k - S_{k-1}}{3}\right|_{\text{Trapezoidal}}$.

Simpson's method converges faster & it is $O(h^4)$ accurate.

Systems of linear Equations

Numerical methods for solving linear systems $\Rightarrow$ (i) Direct methods (ii) Iterative methods. Direct method can solve a set of linear equations or order n at $\approx n^3$ floating point operations and very good solvers for small systems, but round off error is a serious problem as size of n increases, especially e.g. Cramer's rule. Examples are Gaussian and Gauss-Jordan Elimination, LU-factorization, Cholesky decomposition for SPD system.

Iterative methods, e.g. Jacobi and Gauss-Seidel iteration circumvents n^3-dependence & for a well-conditioned matrix converges to exact solution. Truncation error affects this class & can be controlled by tolerance.

Linear system $\boxed{\overline{\overline{A}}\cdot\vec{X} = \vec{B}}$ where $\overline{\overline{A}} = [a_{ij}]_{nn}$, $\vec{X} = [x_j]_n$, $\vec{B} = [b_i]_n$

written explicitly,

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1\\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2\\
\vdots \qquad \vdots \qquad\qquad \vdots \quad &\quad \vdots\\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n,
\end{aligned}$$

$[a_{ii} \neq 0.]$ else arrange so that this condition is met.

Inverting,

$$\begin{aligned}
x_1 &= t_1 + s_{12}x_2 + \cdots + s_{1n}x_n\\
x_2 &= s_{21}x_1 + t_2 + \cdots + s_{2n}x_n\\
&\vdots\\
x_n &= s_{n1}x_1 + s_{n2}x_2 + \cdots + t_n, \text{ where}
\end{aligned}$$

$$\left\{\begin{aligned}
s_{ii} &= 0, \quad i = 1, 2, \ldots, n\\
s_{ij} &= -a_{ij}/a_{ii}, \quad i \neq j, \; j = 1, 2, \ldots, n.\\
t_i &= b_i/a_{ii}
\end{aligned}\right\}$$

or, $\boxed{\vec{x} = \overline{\overline{S}}\cdot\vec{x} + \vec{t}}$ where, $\overline{\overline{S}} = [s_{ij}]_{nn}$, $\vec{t} = [t_i]_n$.

We solve this reduced system by the <u>method of successive approximations</u> from initial approximation $\vec{x}^{(0)} = \vec{t}$, & using recurrence relation, the K^{th} order approximation based on the $(K-1)$th approximation

$$\vec{x}^{(K)} = \overline{\overline{S}}\cdot\vec{x}^{(K-1)} + \vec{t}, \quad K = 1, 2, \cdots \quad \text{with } \vec{x}_{exact} = \lim_{K\to\infty} \vec{x}^{(K)}.$$

Explicitly the iterative procedure is

$$x_i^{(K)} = \sum_{\substack{j\neq i}}^{n} s_{ij}\, x_j^{(K-1)} + t_i, \quad i = 1, 2, \ldots, n.$$

If absolute error is $\Delta_i^{(K)} = x_i^{(K)} - x_i^{(K-1)}$, $i = 1, 2, \ldots, n$, then Jacobi iteration takes the form,

Jacobi

$$\begin{cases} \Delta_i^{(K)} = \sum_{j=1}^{n} s_{ij}\, x_j^{(K-1)} + t_i \\ x_i^{(K)} = x_i^{(K-1)} + \Delta_i^{(K)}, \quad i=1,2,\dots,n. \end{cases} \qquad \begin{cases} s_{ij} = -a_{ij}/a_{ii}, \quad i,j=1,2,\dots n \\ t_i = b_i/a_{ii} \\ s_{ii} = -1 \end{cases}$$

In Gauss-Seidel method, Jacobi method is improved by using the most recently updated $x_i^{(K)}$ instead from the previous iteration $x_i^{(K-1)}$ before even completion of the iteration.

Gauss-Seidel

$$\begin{cases} \Delta_i^{(K)} = \sum_{j=1}^{i-1} \underbrace{s_{ij}\, x_j^{(K)}}_{\text{updated}} + \sum_{j=i}^{n} s_{ij}\, x_j^{(K-1)} + t_i \\ x_i^{(K)} = x_i^{(K-1)} + \Delta_i^{(K)}, \quad i=1,2,\dots,n. \end{cases}$$

For SPD matrices, Gauss-Seidel always converges irrespective of the initial approximation. Also these algorithms are convergent for following conditions, $\sum_{j=1}^{n} |s_{ij}| < 1, \quad i=1,2,\dots,n$

$$\sum_{i=1}^{n} |s_{ij}| < 1, \quad j=1,2,\dots,n$$

and $|a_{ii}| > \sum_{j\neq i} |a_{ij}|, \quad i=1,2,\dots,n.$

"strictly diagonally dominant."

In practice, in most system $|a_{ii}| > \max_{j\neq i} |a_{ij}|, \quad i=1,2,\dots,n$