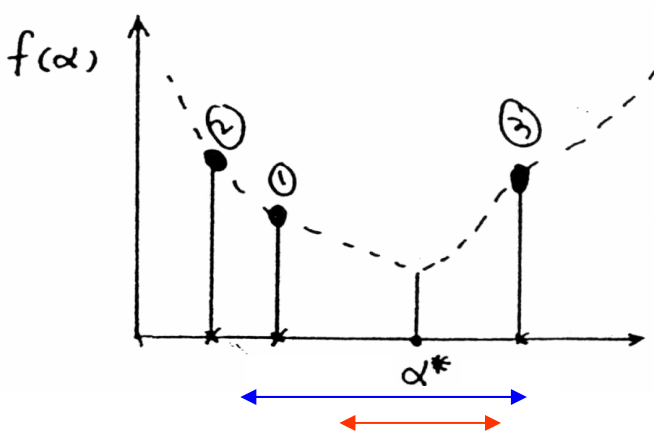
Introduction to Design Optimization: Search Methods

1-D Optimization



- The Search
  - We don't know the curve. Given α, we can calculate $f(\alpha)$.
  - By inspecting some points, we try to find the approximated shape of the curve, and to find the minimum, α^* numerically, using as few function evaluation as possible.
- The procedure can be divided into two parts:

  a) Finding the "range" or region " known " to contain α^* .

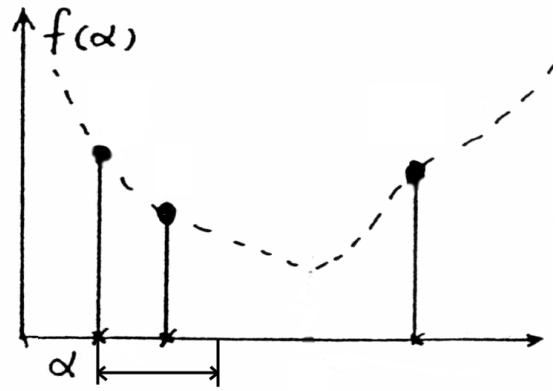
  b) Calculating the value of α^* as accurately as designed or as possible within the range – narrowing down the range.

Search Methods

- Typical approaches include:
  - Quadratic Interpolation (Interpolation Based)
  - Cubic Interpolation
  - Newton-Raphson Scheme (Derivative Based)
  - Fibonacei Search (Pattern Search Based)
  - Golden Section Search
- Iterative Optimization Process:
  - Start point $\alpha_o \rightarrow$ OPTIMIZATION $\rightarrow$ Estimated point α_k $\rightarrow$ New start point α_{k+1}
  - Repeat this process until the stopping rules are satisfied, then $\alpha^* = \alpha_n$.

Iterative Process for Locating the Range

- Picking up a start point, $\alpha_{o,}$ and a range;
- Shrinking the range;
- Doubling the range;
- Periodically changing the sign.

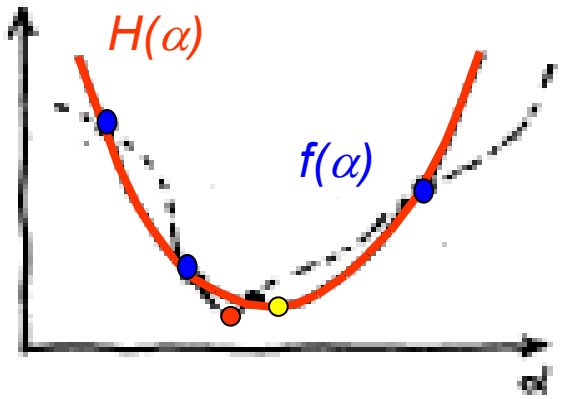


Typical Stopping Rules:

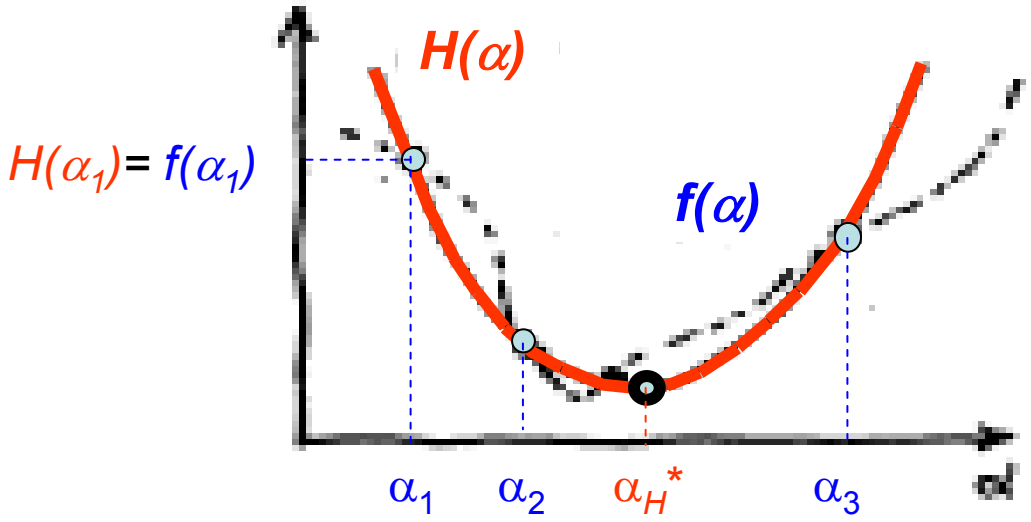
$$\frac{\left|f(new\ \alpha^*) - f(last\ \alpha^*)\right|}{\left|f(new\ \alpha^*)\right|} < \varepsilon \quad or \quad \frac{\left|new\ \alpha^* - last\ \alpha^*\right|}{new\ \alpha^*} < \varepsilon$$

Quadratic Interpolation Method

$$f(\alpha) \Leftarrow H(\alpha) = a + b\alpha + c\alpha^2$$



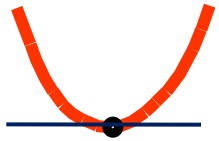
- Quadratic Interpolation uses a quadratic function, $H(\alpha)$, to approximate the "unknown" objective function, $f(\alpha)$.
- Parameters of the quadratic function are determined by several points of the objective function, $f(\alpha)$.
- The known optimum of the interpolation quadratic function is used to provide an estimated optimum of the objective function through an iterative process.
- The estimated optimum approaches the true optimum.
- The method requires proper range being found before started.
- It is relatively efficient, but sensitive to the shape of the objective



Assume $f(\alpha) \approx H(\alpha) = a + b\alpha + c\alpha^2$

$$\frac{dH(\alpha)}{d\alpha} = b + 2c\alpha = 0$$

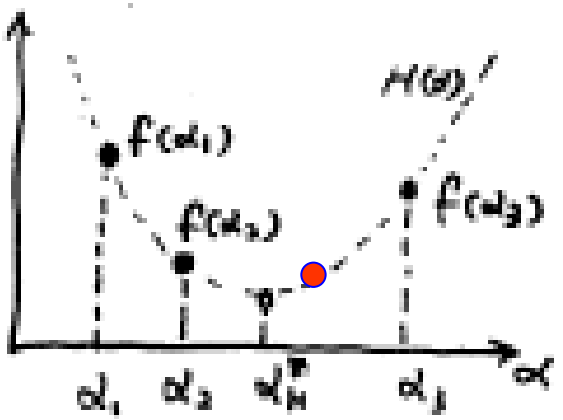
$$\alpha_H^* = -\frac{b}{2c}$$



α_H^* is the optimum for $H(\alpha)$, and is used to estimate the optimum for $f(\alpha)$.

To find parameters a, b, c, three function evaluations are required at $\alpha_1, \alpha_2, \alpha_3$:

$$a + b\alpha_1 + c\alpha_1^2 = f(\alpha_1) = f_1$$
$$a + b\alpha_2 + c\alpha_2^2 = f(\alpha_2) = f_2$$
$$a + b\alpha_3 + c\alpha_3^2 = f(\alpha_3) = f_3$$



$$b = \frac{(f_1 - f_2) - C(\alpha_1^2 - \alpha_2^2)}{\alpha_1 - \alpha_2}$$

$$c = \frac{(f_1 - f_3)(\alpha_1 - \alpha_2) - (f_1 - f_2)(\alpha_1 - \alpha_3)}{(\alpha_1^2 - \alpha_3^2)(\alpha_1 - \alpha_2) - (\alpha_1^2 - \alpha_2^2)(\alpha_1 - \alpha_3)}$$

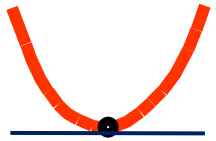
numerator – 2

denominator - 1

$$\boxed{\alpha_H^* = -\frac{b}{2C}}$$ — necessary condition for the optimum of $H(\alpha)$.

Now, chose α_1, α_2, α_3 to satisfy the sufficient condition for α_H^* to be the optimum of $H(\alpha)$

$$\frac{d^2H(\alpha)}{d\alpha^2} = 2C > 0 \quad \text{or} \quad C > 0$$



Consider $0 \le \alpha_1 < \alpha_2 < \alpha_3$

— α value chosing law

- The denominator of C:

$$(\alpha_1 - \alpha_3)(\alpha_1 - \alpha_2)[(\alpha_1 + \alpha_3) - (\alpha_1 + \alpha_2)]$$

$$= (\alpha_3 - \alpha_1)(\alpha_2 - \alpha_1)(\alpha_3 - \alpha_2) > 0$$

- The numerator of C:

$$N = (\alpha_3 - \alpha_1)(f_1 - f_2) - (\alpha_2 - \alpha_1)(f_1 - f_3)$$

$\because (\alpha_3 - \alpha_1) > (\alpha_2 - \alpha_1)$

if $(f_1 - f_2) > (f_1 - f_3)$ and $(f_1 - f_2) > 0$, $N > 0$

or $C > 0$

Condition: $f_1 > f_2$ and $f_2 < f_3$

Range required to find the optimum using quadratic interpolation:

$$0 \le \alpha_1 < \alpha_2 < \alpha_3$$

$$f_1 > f_2 < f_3$$

Point update schemes based on the relations between the center point, α_2, $f(\alpha_2)$, and the present optimum: α^*, $f(\alpha^*)$.

Algorithm:

After sorting up the 3 points from Range Finding, carry out an interpolation to calculate α_M^* & $f(\alpha_M^*)$

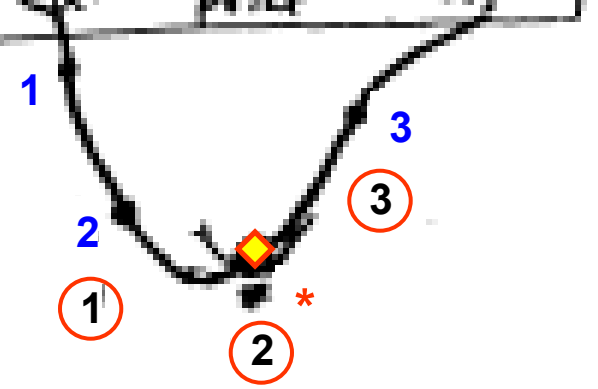
- relative mag. of $f(\alpha^*)$ > center < $f(\alpha_2)$
- relative pos. of α^* > center point < α_2

1) If $f(\alpha_M^*) < f(\alpha_2)$

and $\alpha_2 < \alpha_M^* < \alpha_3$

Let $\alpha_1 = \alpha_2$, $\alpha_2 = \alpha_M^*$

$\alpha_3 = \alpha_3$ and recompute α_M^*

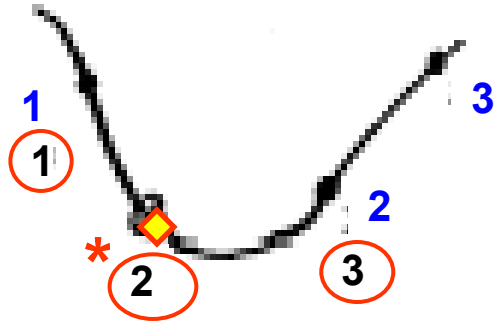


2) If $f(\alpha_M^*) < f(\alpha_2)$

and $\alpha_1 < \alpha_M^* < \alpha_2$

Let $\alpha_1 = \alpha_1$, $\alpha_2 = \alpha_M^*$

$\alpha_3 = \alpha_3$ and recompute α_M^*



3) If $f(\alpha_H^*) > f(\alpha_2)$

and $\alpha_2 < \alpha_H^* < \alpha_3$

Let $\alpha_1 = \alpha_2$, $\alpha_2 = \alpha_H^*$

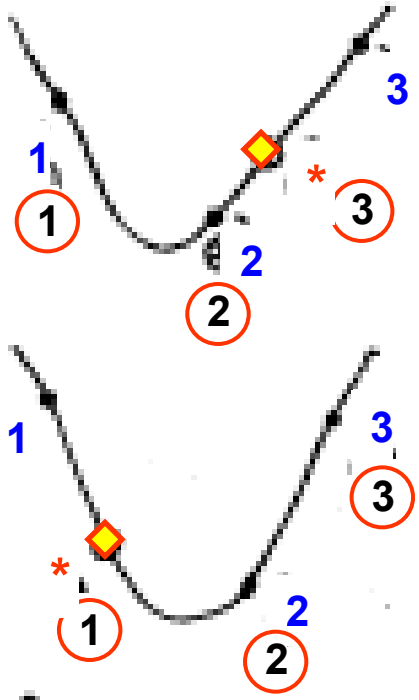
$\alpha_3 = \alpha_H^*$ and recompute α_H^*

4) If $f(\alpha_H^*) > f(\alpha_2)$

and $\alpha_1 < \alpha_H^* < \alpha_2$

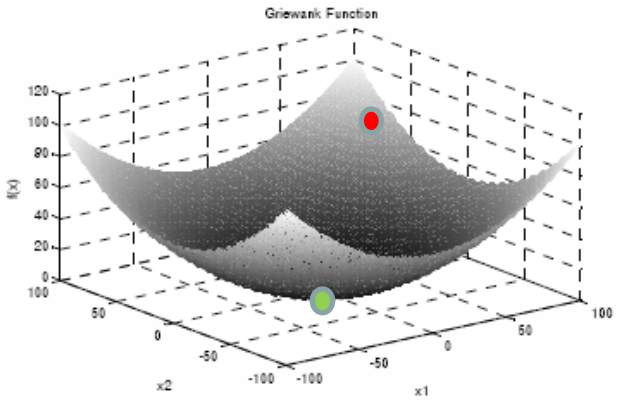
Let $\alpha_1 < \alpha_H^*$, $\alpha_2 = \alpha_3$

$\alpha_3 = \alpha_2$ and recompute α_H^*



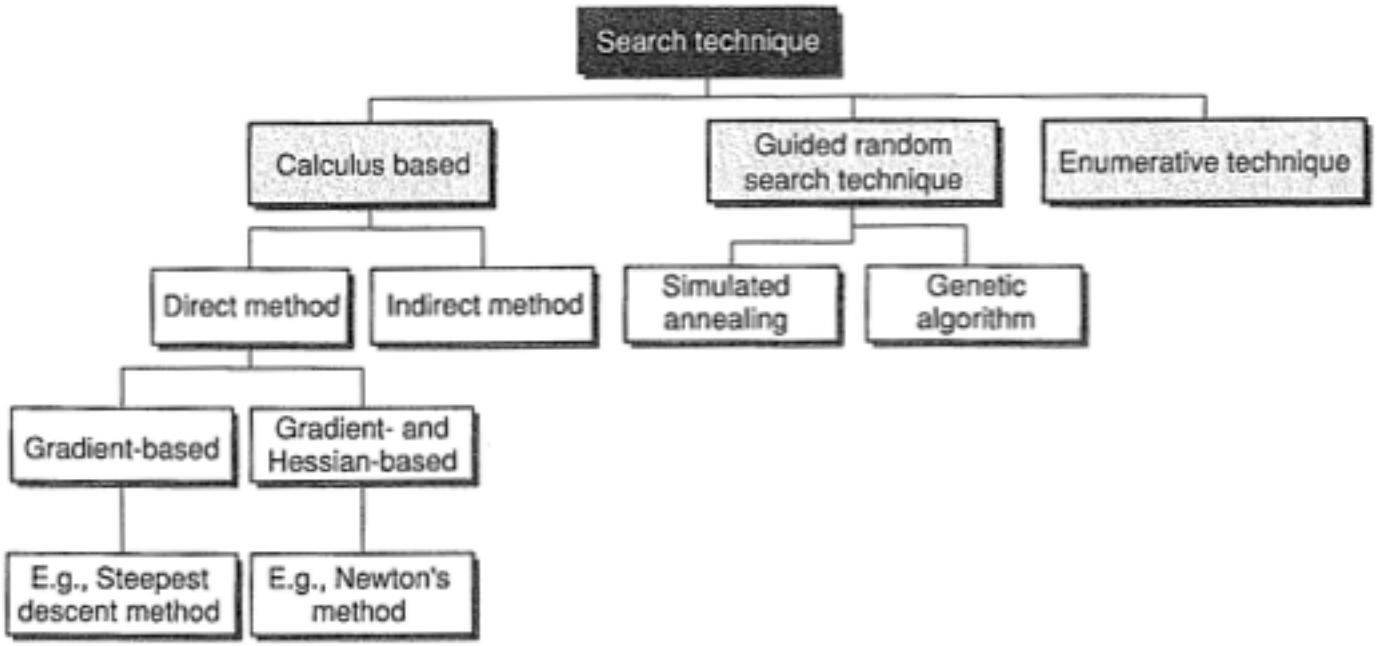
Repeat, until a "stopping rule" is satisfied.

N-Dimensional Search



- The Problem now has N Design Variables.
- Solving the Multiple Design Variable Optimization (Minimization) Problem Using the 1-D Search Methods Discussed Previously
- This is carried out by:
  - To choice a direction of search
    - To deal one variable each time, in sequential order - easy, but take a long time (*e.g.* $x_1, x_2, \ldots, x_N$)
    - To introduce a new variable/direction that changes all variables simultaneously, more complex, but quicker (*e.g.* $\boldsymbol{S}$)
  - Then to decide how far to go in the search direction (small step $\varepsilon = \Delta x$, or large step determining α by 1D search)

N-D Search Methods

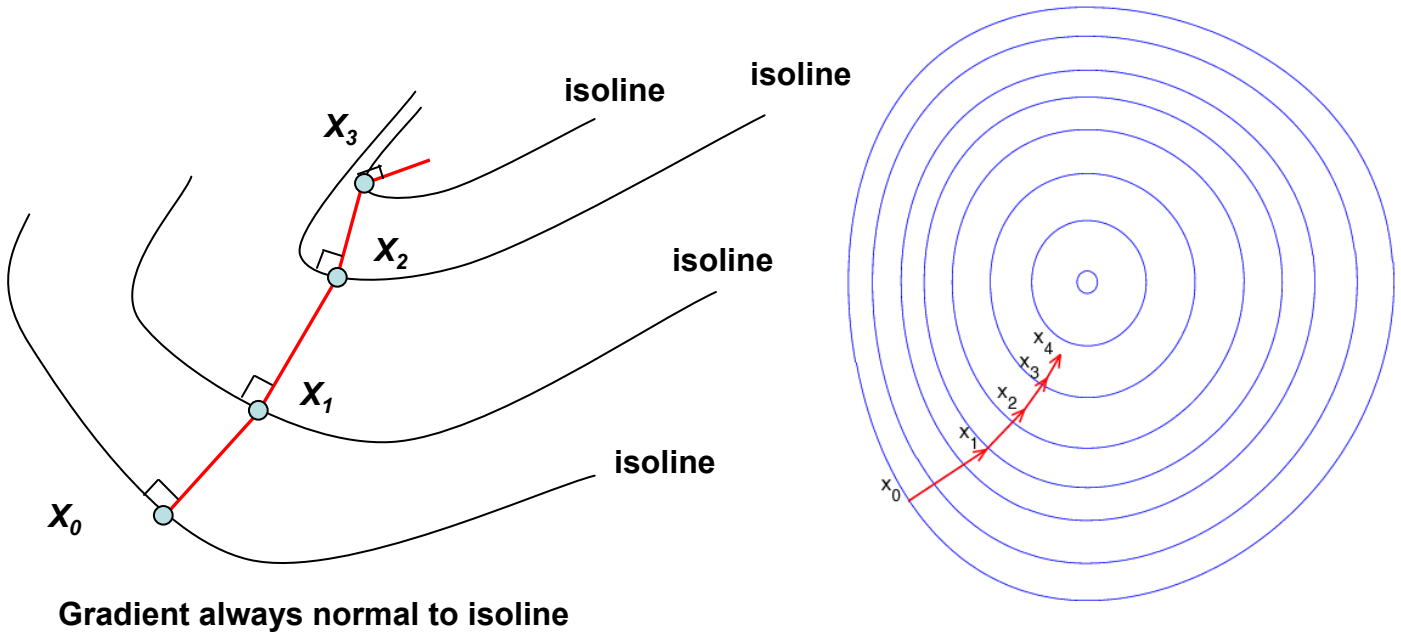


N-D Search Methods

- Calculus Based
  - Indirect method: knowing the objective function set the gradient to Zero. If we need to treat the function as a “black box” we cannot do this. We only know F(X) at the point we evaluate the function.
  - Indirect Method
    - Steepest Descent method
    - Different flavors of Newton methods
- Guided random search-combinatory techniques
  - Genetic method
- Enumerative method: scan the whole domain. This is simple but time consuming

Steepest Descent or Gradient Descent

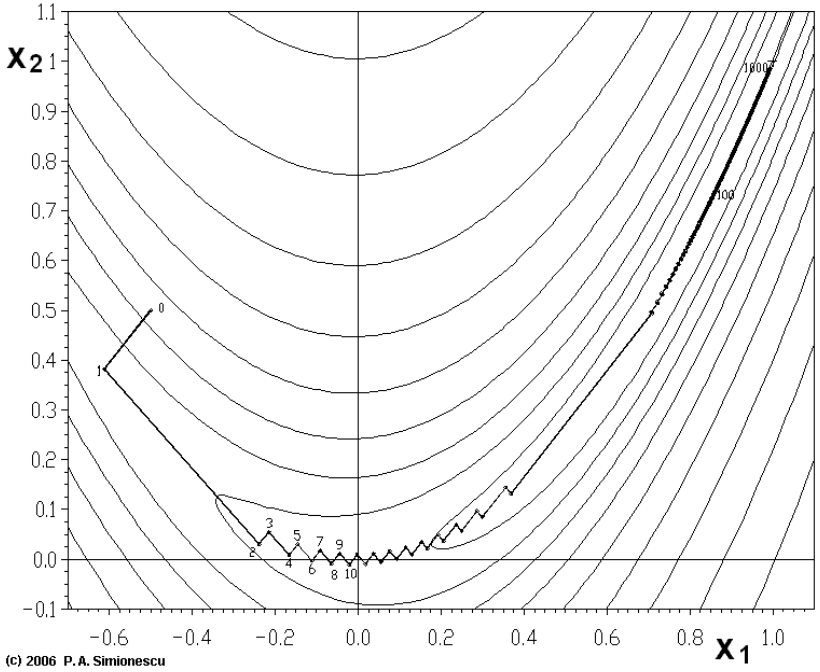
- the gradient of a scalar field is a vector field which points in the direction of the greatest rate of increase of the scalar field, and whose magnitude is the greatest rate of change. This means that if we move in its negative direction we should go downhill and find a minimum. This is the same path a river would follow. Given a point in the domain the next point is chosen as it follows:



Gradient always normal to isoline

Steepest descent and ill-conditioned functions

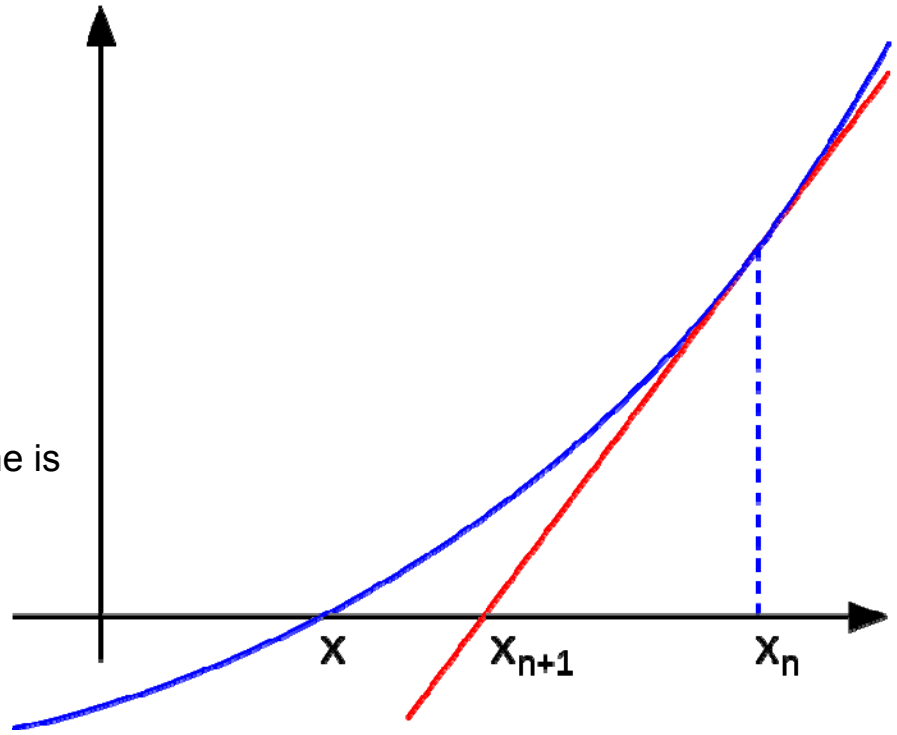
- Gradient descent has problems with ill-conditioned functions such as the Rosenbrock function shown here. The function has a narrow curved valley which contains the minimum. The bottom of the valley is very flat. Because of the curved flat valley the optimization is zig-zagging slowly with small stepsizes towards the minimum.



Newton-Raphson Method

- The Newton-Raphson method is defined by the recurrence relation:

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

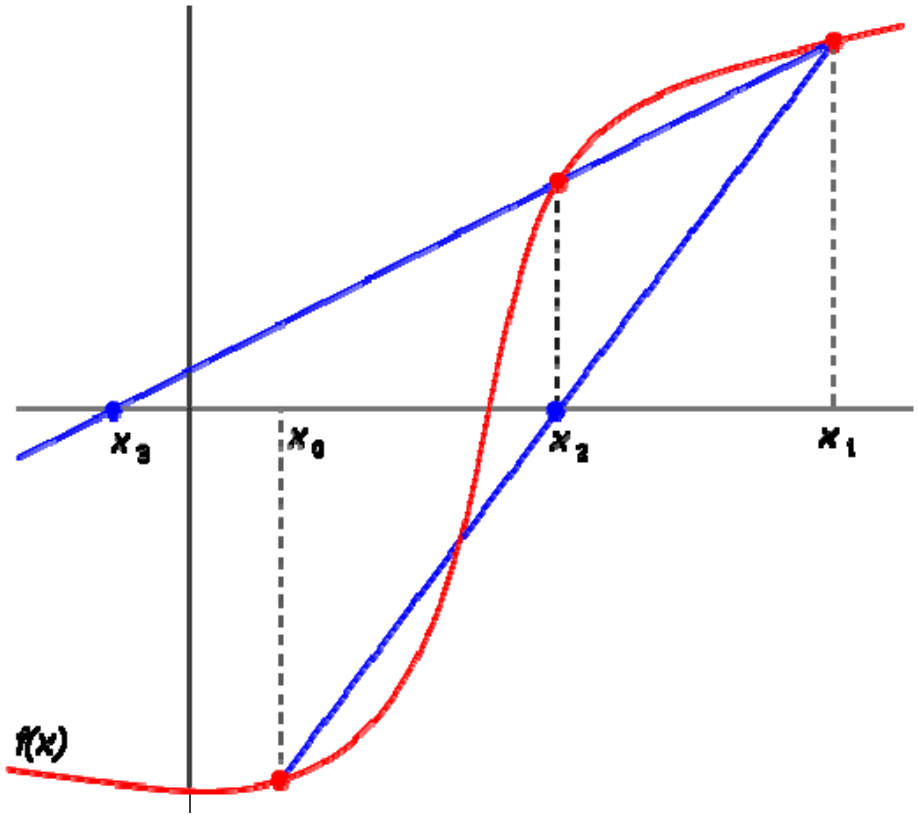


An illustration of one iteration of Newton's method (the function f is shown in blue and the tangent line is in red). We see that x_{n+1} is a better approximation than x_n for the root x of the function f.

Secant Method

- The Secant method is defined by the recurrence relation:

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



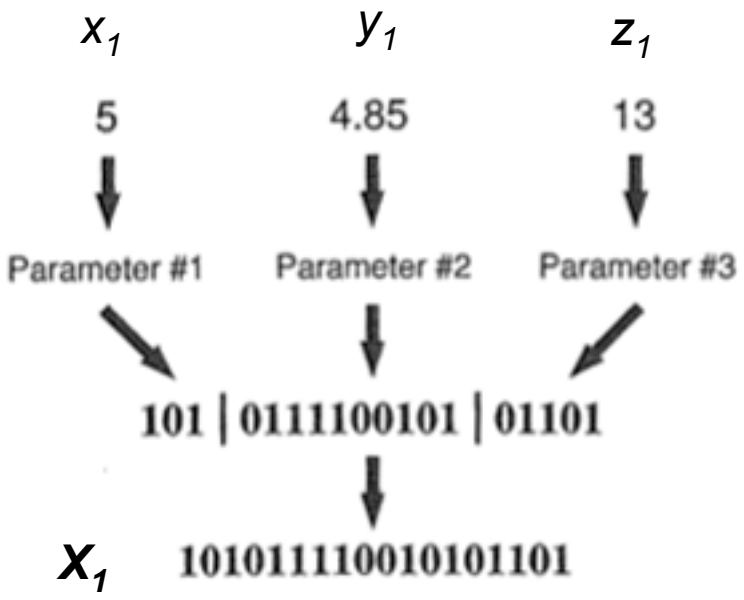
The first two iterations of the secant method. The red curve shows the function f and the blue lines are the secants.

Combinatory Search: Genetic Algorithm

- Valid for discrete variables
- One of the best “all purposes” search method.
- Emulates the genetic evolution due to the “survival of the fittest”
- Each ***variable value*** >a ***GENE***, a binary string value in the variable range
- Vector variables ***X***> a ***CHROMOSOME***, a concatenation of a random combinations of ***Genes*** (strings) one per type (one value per variable). A ***Chromosome*** ($\boldsymbol{X_i}$) is a point in the ***X*** domain and is also defined as *genotype*.
- Objective Function ***F(X)***>*phenotype*. ***F($\boldsymbol{X_i}$)*** is a point in the Objective Function domain corresponding to $\boldsymbol{X_i}$.

Genetic Algorithm

- Construction of a chromosome $\boldsymbol{X}_i(x_i, y_i, z_i)$



Genetic Algorithm

1) Construct a population of genotypes (chromosomes) and evaluate the phenotypes (objective function).

2) Associate a fitness value between 0 and 1 to each phenotype with a fitness function. This function normalizes the phenotype (objective function) and assigns to its genotype an estimate (between 0 and 1) of its ability to survive.

3) Reproduction. The ability of a genotype to reproduce is a probabilistic law biased by the value given by the fitness function. Reproduction is done as it follows:

Given 2 candidate for reproduction, we have:

a) Cloning. The offspring is the same as the parents

b) Crossover. Chromosomes are split in two (head and tail) at a random point between genes and rejoined swapping the tails. When crossover is performed Mutation takes place. Each Gene is slightly changed to explore more possible outcomes.

4) Convergence. The algorithm stops when all genes in all individuals are at 95% percentile

<Genetic Algorithm>

```
BEGIN /*Genetic algorithm*/
    Generate initial population:
    Compute fitness of each individual

    WHILE NOT finished DO
    BEGIN /*Produce new generation*/

        FOR population_size/2 DO
        BEGIN /*Reproduction cycle*/
            Select two individuals from old generation for mating;
            /*Biased in favor of the fitter ones*/
            Recombine the two individual to give offspring;
            Compute fitness of the two offspring;
            Insert offspring in new generation;
        END

        IF population has converged THEN
            finished:=TRUE;

    END

END
```

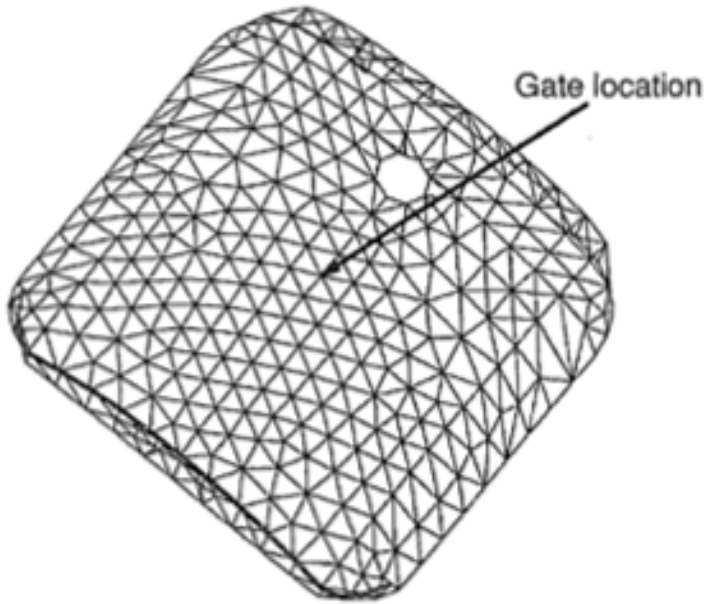
Genetic Algorithm. Example

9.5.2 Application

The genetic algorithm can be used to find the optimal values of the molding condition in the injection molding process of a plastic part. Let's consider an upper cover of a washing machine, as illustrated in Figure 9.19. We want to find the optimal mold temperature, melt temperature, and filling time for a maximum performance

Figure 9.19

Test model



Genetic Algorithm. Example

Figure 9.20

Range of molding conditions

	Minimum	Maximum	Values
Melt temperature	220	260	32
Mold temperature	50	70	32
Filling time	1	4	16

index that is related to the quality of the part to be fabricated. For simplicity, we assume that the performance index is calculated by simulation of the molding conditions. The lower and upper bounds of the optimization variables are given in Figure 9.20.

Now let's consider how to encode these variables. As shown in Figure 9.20, the melt temperature is assumed to have 32 discrete values and is thus represented by 5 digits. Similarly the mold temperature and the filling time are represented by 5 and 4 digits, respectively. Thus the chromosome in this example will be a binary string of length 14.

Genetic Algorithm. Example Results

Outcome:

	Melt temperature	Mold temperature	Filling time	Object function
Result	220.0	70.0	2.6	45.00