VALUE ENGINEERING AND OPTIMIZATION SUMMARY

1. Simplex Method

Basic Rules
- Input all functions and fill out columns by adding slack variables.
- Keep going until all #s on the top row are positive
- Valid for standard constraints (≤)

How to select a pivot point (on the top column.)
- Choose: COLUMN with most negative #
  ROW for which $\frac{\text{value last colum}}{\text{value current column}}$ is the smallest yet larger than 0.

2.1. The Big-M Method (case "=")

- To be used if one of the constraints features "=" instead of the standard "≤".

  e.g. ⇒ standard : $3x_1 + 2x_2 \leq 18$
  case "=" : $3x_1 + 2x_2 = 18$

- Set up the same as normal, but for the "=" constraint, introduce an artificial variable instead of a regular slack.

Difference between a slack variable and an artificial variable
- Slack variable : Introduces $0 \cdot x_5$ in z-equation (so, nothing)
- Artificial variable : Introduces $M \cdot x_5$ in z= equation

2.2. Big-M Method (case "≥")

- Constraint has form "≥" : eg. $3x_1+2x_2 \geq 18$

- Solution: Add the following to the constraint
  - Surplus variable (same as slack but negative) — Deals with ">"
  - Artificial variable — Deals with "="

→ Together they deal with "≥"

- Proceed, once these two have been added, with the regular solution:
  - → Get rid of M from x_5 column.
  - → Proceed with Simplex method.

Note: Always maximize with the Big-M method. If you need to minimize then just maximize $(-z)$

3. The 2-phase Method

- Used as an alternative to the Big-M method
- Still based on it.
- When we introduce an artificial variable, we need to consider it in the new objective equation, which is obtained by dividing the augmented Z equation with M (which is really big, so only M terms are left!

∴ $z_{new} = x_3 + x_4$

$z_{original} = 0.4x_1 + 0.5x_2$

Phase 1: Minimize until $x_3, x_4 = 0$

Phase 2: Minimize once $x_3, x_4 = 0$

Phase 1: Used to derive an initial basic feasible solution to the problem.

Phase 2: Used to find the optimal solution.

For minimizing →
For maximizing →

Basic Solution: x (of $Ax=b$, where A is the constraint matrix and b the constraints) is a basic solution if it features m basic variables and n-m non-basic variables.

Basic Variable: A variable in the basic solution (not $=0$)

Non-basic Variable: A variable not in the basic solution ($=0$)

4. The Simplex Method in Matrix Form.

- Maximize: $Z = \vec{c}\,\vec{x} = [c_1 \dots c_n] \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = c_1x_1 + \dots + c_nx_n$

- Constraints: $A\vec{x} \leq \vec{b}$ $\quad \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$

- Non-negativity Constraints $\quad \vec{x} \geq 0 \quad \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \geq \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$

- To obtain the augmented form of a problem we can introduce slack variables in a column vector.

This is basically an identity matrix.

The constraints then become:

$$[A, I] \begin{bmatrix} \vec{x} \\ \vec{x}_s \end{bmatrix} = \vec{b} \quad , \quad \begin{bmatrix} \vec{x} \\ \vec{x}_s \end{bmatrix} \geq 0$$

Constraints → ; ← Non negativity constraints.

where $\vec{x}_s = \begin{bmatrix} x_{n+1} \\ \vdots \\ x_{n+m} \end{bmatrix}$

We can then put everything together in one big system:

$$\begin{bmatrix} 1 & -\vec{c} & \vec{0}^T \\ \vec{0} & A & I \end{bmatrix} \begin{bmatrix} Z \\ \vec{x} \\ \vec{x}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{b} \end{bmatrix}$$

Note that this matrix breakdown is valid for the standard form only, i.e ("$\leq$")

Basis Matrix B

- A basic variable is part of the final solution.
- The Basis matrix is a matrix containing the factors by which to multiply the basic variables with in order to obtain the solution that fits the constraints

$$\therefore \quad B\vec{x}_B = \vec{b}$$

B: $m \times m$ Matrix

$\vec{x}_B$: Vector of basic variables i.e. the solution variables.

Then: $x_b = B^{-1}b$

And because x_B contains the solution, then

$$\boxed{Z = \vec{C}_B B^{-1} b} = \vec{C}_B \vec{x}_B$$

$$\begin{bmatrix} Z \\ \vec{x}_b \end{bmatrix} = \begin{bmatrix} 1 & \vec{C}_B B^{-1} \\ \vec{0} & B^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ \vec{b} \end{bmatrix}$$

Remember that we perform multiplication on both sides, so:

$$\begin{bmatrix} 1 & C_B B^{-1} \\ \vec{0} & B^{-1} \end{bmatrix} \begin{bmatrix} 1 & -\vec{c} & 0 \\ 0 & A & I \end{bmatrix} = \begin{bmatrix} 1 & \vec{C}_B B^{-1} A - C & \vec{C}_B B^{-1} \\ 0 & B^{-1}A & B^{-1} \end{bmatrix}$$

Giving:

$$\begin{bmatrix} 1 & \vec{C}_B B^{-1} A - \vec{c} & \vec{C}_B B^{-1} \\ \vec{0} & B^{-1}A & B^{-1} \end{bmatrix} \begin{bmatrix} Z \\ \vec{x} \\ \vec{x}_s \end{bmatrix} = \begin{bmatrix} \vec{C}_B B^{-1} \vec{b} \\ B^{-1}\vec{b} \end{bmatrix}$$

With the constraint that $B^{-1}b \geq 0$

we can use that to determine the range of b.

The Simplex Method in Matrix Form: Example

$$\left[\begin{array}{c|cc|ccc} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \hline 1 & -3 & -5 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 & 0 \\ 0 & 3 & 2 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c} Z \\ \hline x_1 \\ x_2 \\ \hline x_3 \\ x_4 \\ x_5 \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline 4 \\ 12 \\ 18 \end{array}\right]$$

- **Iteration 0:** $x_B = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} \rightarrow B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$\therefore x_B = B^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$\vec{b}$

These are the values of the objective function coefficients at the columns chosen

$$C_B = [0 \ 0 \ 0] \rightarrow Z = C_B B^{-1} b = 0$$

- **Iteration 1:** $x_B = \begin{bmatrix} x_3 \\ x_2 \\ x_5 \end{bmatrix} \rightarrow B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix}$

$$\therefore x_B = B^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix}$$

Note that this is the negative of $\vec{c}$!

$$C_B = [0 \ 5 \ 0] \rightarrow Z = C_B B^{-1} b = [0 \ 5 \ 0] \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix} = 30$$

Iteration 2: $x_B = \begin{bmatrix} x_3 \\ x_2 \\ x_1 \end{bmatrix} \rightarrow B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{bmatrix}$

$$x_B = B^{-1} \vec{b} = \begin{bmatrix} 1 & 1/3 & -1/3 \\ 0 & 1/2 & 0 \\ 0 & -1/3 & 1/3 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

$$C_B = [0 \ 5 \ 3] \qquad Z = \vec{C}_B B^{-1} \vec{b} = \underline{\underline{36}}$$

The convenience comes in that if the right side of the constraints changes (i.e. $\vec{b}$), $\vec{C_B}$ and B will not change!

So the new solution is just a matter of plugging in a different b vector in the matrix multiplication.

C_B and B are only dependent on the iteration in use, but once the iteration is optimal, they stay constant.

The Simplex Method in Matrix Form: A fundamental insight

→ Entire rows of the simplex table can be obtained by matrix multiplication.

$S^* = B^{-1}$: Coefficients of the slack variables

$A^* = B^{-1}A$: Coefficients of original variables

$y^* = C_B B^{-1}$: Coefficients of slack variables in row 0

$\vec{z}^* = C_B B^{-1} A \rightarrow \vec{z}^* - c \Rightarrow$: Coefficients of original variables in row 0

$Z = C_B B^{-1} b$: Optimal solution of objective function

$b^* = B^{-1} b$: Optimal right hand side for rows 1→m

$$\therefore \quad \begin{bmatrix} 1 & \vec{z}^* - \vec{c} & y^* \\ 0 & A^* & S^* \end{bmatrix} \begin{bmatrix} z \\ x \\ x_s \end{bmatrix} = \begin{bmatrix} Z^* \\ b^* \end{bmatrix}$$

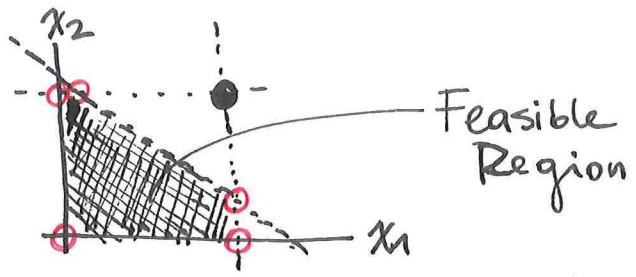
S^* is really important! See the following:

$$E = [-\vec{c} \mid 0 \mid 0]$$
$$T = [A \mid I \mid \vec{b}]$$

Basically the formulation is achieved by multiplying the whole bottom part with B^{-1} (S^*)

$$E^* = [z^* - c \mid y^* \mid Z^*]$$
$$T^* = [A^* \mid S^* \mid b^*] = [S^*A \mid S^* \mid S^*\vec{b}] = S^*T$$

5. Corner Point Feasible (CPF) solution



● : Corner Point Feasible Solution
● : Corner Point Infeasible Solution

The constraint boundary equation is obtained by replacing the boundary $\leq, =, \geq$ with $(=)$.

[This is so that only the limits are observed]

Properties of the CPF solution

1. If there is an optimal solution, it must be a CPF. If there are multiple, at least 2 must be adjacent.
2. There can only be a finite # of CPF solutions.
3. If a CPF solution has no adjacent CPF solutions that are better, then there are no better CPF solutions anywhere.
   This is related to the second part of property 1.

Augmented forms of the problem and their extensions

Augmented refers to the fact that we have introduced slack/surplus/augmented variables.

Basic Solution ⇒ Augmented CP(not feasible) solution

Basic Feasible Solution ⇒ Augmented CPF solution

6. The essence of sensitivity analysis

Using the matrix representation of the simplex method, sensitivity analysis becomes less computationally intensive.

Assume that we go on a rampage and decide to change 3 parameters at once, such that:

$A \rightarrow A_{new}$
$c \rightarrow c_{new}$
$b \rightarrow b_{new}$

} so we pretty much change everything

Using what we learned from the fundamental insight, we can take a shortcut to determine the new solution.

Then $\left[\begin{array}{cccc|c} 1 & y^*A - c & & y^* & y^*b \\ 0 & S^*A & & S^* & b^* \end{array}\right]$

Note:
- This may not necessarily give the optimum, since we changed a lot, but it will definitely bring us close to the solution.
- More computationally efficient!
- In case of only changing $\vec{b}$, then we immediately get the new optimum.

7. Shadow Prices, Reduced Cost & Allowable range

SHADOW PRICES

⇒ The shadow price of a constraint is the change in the optimal value of the objective function per unit increase in the right handed value of the constraints.

→ This refers to how much the optimal solution increases if we increase one of the constraints.

- It is a property of the constraint
- It tells us the impact of changing the constraints

REDUCED COST

- This applies to the non-negativity constraints
- It is the same concept as shadow price but applied to those constraints, so...

what if $x_1 \geq 0$ becomes $x_1 \geq 1$?

- This has an impact because it forces a minimum value.

Both shadow price and reduced cost could be thought of as "opportunity cost".

ALLOWABLE RANGE

- Sensitive parameters are those that, if changed, affect the final solution

Definition
- If a parameter is not sensitive, then it is useful to know the range of values of the coefficient over which the optimal solution remains unchanged.

- This could also extend to what point the right hand values of the constraints are changed such that the solution becomes unfeasible.

Range of ~~b~~!

Maximizing or minimizing

With IOR we always want to Maximize.
Luckily, we can just reverse the problem:

Minimizing

$$Z = \sum_{j=1}^{m} c_j x_j \quad \equiv$$

Maximizing

$$(-Z) = \sum_{j=1}^{m} (-c_j)(x_j)$$

Then, once you have found the solution to the basic variables, you can plug these in the original function for Z.

8. The Dual Problem

Primal Problem

Max. $Z = \sum_{j=1}^{n} c_j x_j$

$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$

$x_j \geq 0$

Dual Problem

Min $W = \sum_{i=1}^{m} b_i y_i$

$\sum a_{ij} y_i \geq c_j$

$y_i \geq 0$

Because you want to minimize the sum of the constraints

Swap these two

Changes sign.

While keeping a minimum max value!

The Dual Problem is particularly useful when there is a large amount of constraints.

It allows for less computations but eventually provides the same answer.

Striving for optimality in the primal problem is equivalent to striving for feasibility in the dual problem.

Example of Dual Simplex Method

Minimize

$Z = +4y_1 + 12y_2 + 18y_3$

$y_1 + 3y_2 \geq 3$

$2y_1 + 2y_3 \geq 5$

$y_1, y_2, y_3 \geq 0$

You need to get this in standard form!

∴ Maximize

$-Z = -4y_1 - 12y_2 - 18y_3$

$-y_1 - 3y_2 \leq -3$

$-2y_1 - 2y_3 \leq$

$y_1, y_2, y_3 \geq 0$

Z	y_1	y_2	y_3	y_4	y_5	RHS
1	4	12	18	0	0	0
0	-1	-3	-3	1	0	-3
0	-2	0	-2	0	1	-5

The transportation Problem

This is a streamlined simplex method.

P&T Company Example

- 3 cannaries of peas
- 4 warehouses
- objective: Minimize transportation from cannaries to warehouses.

Shipping costs per truckload.

Cannery \ Warehouses	1	2	3	4	output
1	464	513	654	867	75
2	352	416	690	791	125
3	995	682	388	685	100
Allocation	80	65	70	85	300

Truckloads sent out from each cannery

Max Truckloads accepted at each warehouse.

Total exchange (the sum must add up on both sides, otherwise introduce a dummy variable)

∴ Minimize Z, the cost of transportation.

↳ Minimize $Z = |AX|$ ← Determinant.

$$A = \begin{bmatrix} 464 & 513 & 654 & 867 \\ 352 & 416 & 690 & 791 \\ 995 & 682 & 388 & 685 \end{bmatrix}$$

$$X = \begin{bmatrix} x_{11} & x_{21} & x_{31} \\ x_{12} & x_{22} & x_{32} \\ x_{13} & x_{23} & x_{33} \\ x_{14} & x_{24} & x_{34} \end{bmatrix}$$

Packs from each cannery to be sent to each warehouse.

Cannaries → columns; Warehouses → rows

Truckloads accepted by each warehouse

$$X = [\vec{x}_1 \quad \vec{x}_2 \quad \vec{x}_3]$$

This gives rise to certain constraints:

$$\text{sum}(\vec{x}_1) = 75$$
$$\text{sum}(\vec{x}_2) = 125$$
$$\text{sum}(\vec{x}_3) = 100$$

Constraints of cannaries

$$x_{11} + x_{21} + x_{31} = 80$$
$$x_{12} + x_{22} + x_{32} = 65$$
$$x_{13} + x_{23} + x_{33} = 70$$
$$x_{14} + x_{24} + x_{34} = 85$$

Constraints of warehouses.

Another way of setting up the constraints would be:

$$x_{11} + x_{12} + x_{13} + x_{14} = 75$$

$$x_{21} + x_{22} + x_{23} + x_{24} = 125$$

$$x_{31} + x_{32} + x_{33} + x_{34} = 100$$

$$x_{11} + x_{21} + x_{31} = 80$$

$$x_{12} + x_{22} + x_{32} = 65$$

$$x_{13} + x_{23} + x_{33} = 70$$

$$x_{14} + x_{24} + x_{34} = 85$$

$\vec{x}_1$ $\vec{x}_2$ $\vec{x}_3$

In Matrix Form

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \end{bmatrix} = \begin{bmatrix} 75 \\ 125 \\ 100 \\ 80 \\ 65 \\ 70 \\ 85 \end{bmatrix}$$

This is the starting point towards a streamlined solution towards solving the problem.

Intermezzo: General Form of the transportation problem

Goal: Distributing something from sources to destinations

(sources have a supply; destinations have a demand)

Requirements assumption: Supply and demand are fixed. All sources produce a certain amount, all destinations expect a certain amount.

Feasible Solutions property: A transportation solution is feasible iff...

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{m} d_j$$

(sum of units produced = sum of units expected.)

- Dummy sources can be used to circumvent the feasible solutions property.
  ↑ or destinations!

- The cost assumption: Cost is proportional to units moved (1 unit → 3$, 2 units → 6$)
- The Model: Once the problem
  - → Satisfies the requirements assumption
  - → Satisfies the feasible solutions property (with or without dummies)
  - → Satisfies the cost assumption

Then we can proceed to solve it by setting up a parameter table.

Set up of transportation method.

Supply \ Destinations	1	2	...	n	Supply
1	c_{11} x_{11}	c_{12} x_{12}		c_{1n} x_{1n}	s_1
2	c_{21} x_{21}	c_{22} x_{22}		c_{2n} x_{2n}	s_2
⋮			...		⋮
m	c_{m1} x_{m1}	c_{m2} x_{m2}		c_{mn} x_{mn}	s_m
Demand	d_1	d_2	...	d_n	Z

Initial solution: Simple objective of getting all values of demand and supply down to zero.

This is known as Vogel's approximation method

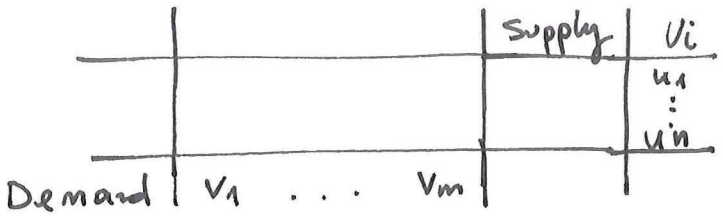
Start from the top left.
Always pick the lowest # out of demand or supply to subtract.

	1	2	3	4	5	
1	16 | 30 →	16 | 20				50
2		14 | 0 ↓ →	13 | 60			60
3			29 | ~~10~~ →	~~30~~ →	10 ↓	50
4					50	50
	30	20	10	30	60	

This will provide an initial solution, which can then be optimised.

$Z = 16(30) + 16(20) + 14(0) + \dots$

The next step is to fill out the shared cost values, v_i and v_i



$u_1 + v_1 = c_{11}$
$u_1 + v_2 = c_{12}$
$u_2 + v_2 = c_{22}$
$u_2 + v_3$
⋮

This is done in accordance to all basic variable locations.

Set the value of u_i with the most allocations to zero and solve.

~~Basically solve that $c_{ij} = u_i - v_j = 0$~~

Plug in u_i and v_i, which are then used to determine the non-basic variables in the left-over cells.

These non-basic variables simbolize a sort of "opportunity cost"

The solve using the whole loop method

→ Entering basic variable → Nonbasic variable that is most negative

How to make the loop → Doesn't really matter as long as the # of ⊕ is equal to the # of ⊖.

Select leaving basic variable → This is most likely going to be the smallest basic variable with a ⊖ sign.

To see if it's correct, check if the balance is kept.

Big-M method → Put an M as a cost of something is infeasible. The Big M signifies infinite cost, so it's impossible

Dummy sources/destination → Put zero cost on the whole row/column and assume they can produce/house the leftover amount to ensure supply = demand.

Non-Linear Programming

The problem set up stays the same: Maximize $f(\vec{x})$

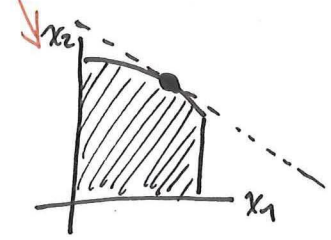
$g_i(\vec{x}) \leq b_i \quad i = 1, \dots, m$

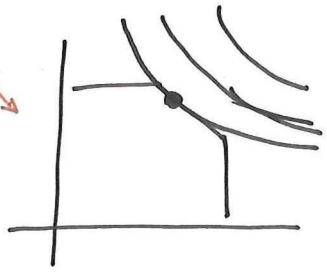
$\vec{x} \geq 0$

But we can observe non-linearity on different things.

→ Nonlinearity in constraints: e.g. $9x_1^2 + 5x_2^2 \leq 216$

→ Nonlinearity in objective equation: e.g. $Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$





Note: • Optimum is not (necessarily!) on the boundary of the feasible region.

⇕

• Optimum is not necessarily a CPF solution anymore.

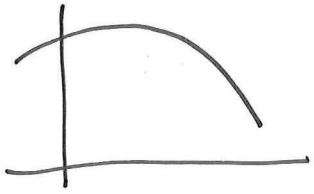
• A local maxima is not necessarily a global maxima, as we had for CPF and linear problem solutions!

Exception: For unconstrained problems

A local maximum/minimum is also global if the function is concave.

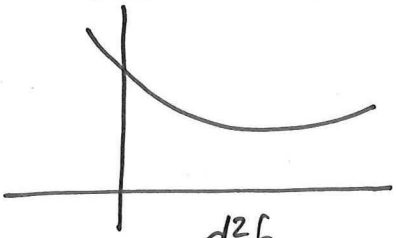
For constrained problems: Feasible region is a convex set.

Concave function



$\frac{d^2f}{dx^2} \leq 0$

Convex function



$\frac{d^2f}{dx^2} \geq 0$

If $<$ or $>$ instead of $\leq$ or $\geq$, then the function is strictly concave / convex

For a non-linear problem that is also constrained, there is a condition for which a local optimum becomes a global optimum.

→ Feasible region must be a convex set of equations. And the objective function must be concave!

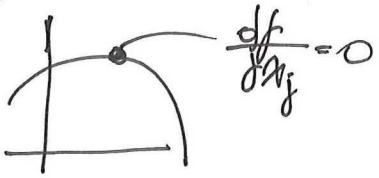
To be a convex set, all constraint functions $g_i(x) \leq b_i$ must be convex.
Note that a linear function can be both.

∴ For local max = global max — concave objective function, convex constraints

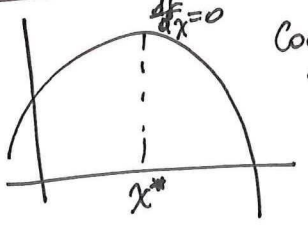
Unconstrained Optimization

Maximize $f(x)$

When $f(x)$ is concave, it is sufficient to determine the point at which $\frac{\partial f}{\partial x_j} = 0$.



One-Variable unconstrained Optimization



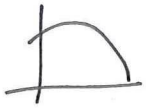
Concave function, No constraints, find $\frac{df}{dx} = 0$ at $x = x^*$

How can we solve such a problem?

There are several numerical methods:

- Bisection Method
- Newton's Method
- Golden Section Search

The Bisection Method

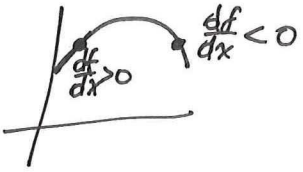


- Can be applied if $f(x)$ is concave to find a maximum
- Uses $\frac{df(x)}{dx}$ to determine which way to go.
- This method makes use of the midpoint rule to converge to x^*.

Steps :

1. Select initial $\underline{x}$ (lower bound) and $\overline{x}$ (upper bound) by inspection.
2. First guess $x' = \frac{\underline{x}+\overline{x}}{2}$
3. Evaluate $\frac{df(x)}{dx}$ at x'
4. If $\frac{df(x)}{dx} > 0$, $\underline{x} = x'$ (Shift lower bound up!)

   If $\frac{df(x)}{dx} < 0$, $\overline{x} = x'$ (Shift upper bound down!)



5. Repeat until $\overline{x} - \underline{x} \leq 2\epsilon$

You are free to select ϵ depending on the required accuracy.

Newton's Method

- The Bisection method works, but is quite slow to converge.
- Newton's method is more efficient as it also makes use of the second derivative, $\frac{\partial^2 f}{\partial x^2}$.

This is based on the Taylor expansion of a function.

$$f(x_{i+1}) \approx f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2}(x_{i+1} - x_i)^2 + \dots$$

which we can truncate at the second term to get

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

This "quadratic" function is so good an approximation that $f'(x_i) = f''(x_i)$ when $x_{i+1} = x_i$. We are striving for this point.

The maximum of the above expansion can be found by setting the derivative to zero.

$$\therefore \quad f'(x_{i+1}) \approx \underbrace{f'(x_i)}_{=0} + f''(x_i)(x_{i+1} - x_i)$$

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

Steps:

1. Select initial trial point by inspection, x_i
2. Determine $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$
3. Stop when $|x_{i+1} - x_i| \leq \epsilon$ → once again, you get to select this.

This method is simple, and a lot faster than the bisection in terms of iterations.

Multivariable Unconstrained Optimization.

Problem: Maximize concave $f(\vec{x})$ of multiple variables.
No constraints of feasibility.

Up to now we were dealing with only $f(x)$, now we will start dealing with $x_1, x_2 \ldots$ and so forth.

[redacted]

GRADIENT SEARCH PROCEDURE

Goal: Reach a point where all partial derivatives are zero.

Gradient function: $\nabla f(\vec{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]^T$

$\vec{x} = [x_1, x_2, \ldots x_n]^T$

We aim for: $\nabla f(\vec{x}^*) = \vec{0}$

$\vec{x}^*$ is the solution!

Steps
1. Select initial position $\vec{x}'$ — Vector!
2. Determine the gradients $\nabla f(\vec{x}')$
3. Express $f(\vec{x}' + t\nabla f(\vec{x}'))$ as a function of t by: [redacted] set $x'_{j,new} = x_j' + t\frac{df}{dx_j}$ — A linear function of the gradient.

You do this for the [redacted] highest gradient!

4. Find $t = t^*$ that maximizes f along this line, and put it in the equation to obtain a new $\vec{x}'$.
5. Repeat until $\left|\frac{\partial f}{\partial x_j}\right| < \epsilon$

t^* tells us how far to move along that gradient!

$\frac{\partial f}{\partial x_j}$ is the highest derivative [redacted] out of $\nabla f(\vec{x}')$ if we are aiming to maximize. If we are aiming to minimize then it's the smallest.

Newton's Multivariable Method

. Newton's method was actually designed for multivariable.

. So instead of $x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$, use

$$\vec{x}' = \vec{x} - [\nabla^2 f(\vec{x})]^{-1} \nabla f(\vec{x})$$

This is the second derivative matrix, also known as a Hessian matrix.

There are several ways to estimate! the inverse of the Hessian, but they are beyond the scope.

Convex Programming (constrained)

. When the objective function to be maximized is concave but the constraints are convex.

. A local maximum is then a global maximum.

FRANK-WOLFE ALGORITHM

. This algorithm for convex programming is quite similar to the gradient search.

Steps

1. Start at a feasible initial solution (0,0) is usually ok if constraints are in standard form.

2. Determine $\frac{\partial f}{\partial x_1}$ and $\frac{\partial f}{\partial x_2}$, i.e. $\nabla f(\vec{x})$ at the point.

3. Generate a function $g(x) = \frac{\partial f}{\partial x_1} x_1 + \frac{\partial f}{\partial x_2} x_2 \dots$

$g(x)$ is now a linear problem for which an optimal solution can be found.

$$(x_1, x_2) = x_{old} + t(x_{LP} - x_{old})$$

Where x_{LP} is a point on the boundary
Then maximize the function and get a new starting point, then repeat!

Sequential Unconstrained Minimization Technique (SUMT) 21

- SUMT replaces the original problem by a sequence of unconstrained optimization problems.
- This approach is attractive because then problems are much easier to solve.
- The trick is to maximize a function which consists of the original function minus a barrier function, $B(\vec{x})$.

$$\therefore \text{ Maximize } P(\vec{x}; r) = f(\vec{x}) - rB(\vec{x}), \quad r>0$$

$B(\vec{x})$ must have the following properties:

1. Small if x is far from boundaries
2. Large if x is close to boundaries

$\therefore B(x) \to \infty$ as distance to boundary $\to 0$

Common choice:

$$B(x) = \sum_{i=1}^{m} \frac{1}{b_i - g_i(\vec{x})} + \sum_{j=1}^{n} \frac{1}{x_j}$$

$B \to \infty$ as $g_i(\vec{x}) \to b_i$

So this prevents the function from being maximized beyond that point, and thus stay away from the boundaries.

Note: A convex programming problem requires:
- → Concave objective function
- → Convex boundaries

Metaheuristics

- Nonlinear problems may have multiple local optima. Previous methods have no way of determining whether they found a local or global optimum, and of escaping the local in case they got it wrong.

- This is necessary for nonconvex problems

- Metaheuristics looks close by (derivatives and what not) like previous methods, but also goes beyond to try and look further for better possibilities.

SIMULATED ANNEALING

3 main parameters:

Z_c = Current trial solution
Z_n = Current candidate for next trial solution
T = Tendency of acceptance of Z_n even if it is not an improvement.

Move Selection Rule:

If $Z_n \geq Z_c$ always accept
If $Z_n < Z_c$ accept according to the following probability:

$$\text{Prob}\{\text{acceptance}\} = e^x, \quad x = \frac{Z_n - Z_c}{T}$$

This is for a maximization problem. It is the opposite for a minimization problem.

T will decrease with more iterations, such that the probability of acceptance decreases.

Temperature Schedule:

To implement this, the easiest way is to generate a random # between 0 and 1

If rand # < e^x, accept

When the desired # of iterations have been performed at the current value of T, decrease T to the next value in the temperature schedule.

- Temperature schedule:
  - Specify the initial (relatively large) value of T.
  - Specify the subsequent values of T to be used
  - Specify how many moves (iterations) are to be made with each value of T.

- Stopping Rule: When to stop?
  - When the Temperature schedule is completed

  OR
  - When no immediate neighbour solution is accepted, albeit this requires performing the check until the temperature schedule runs out anyway.

Accept the best trial solution found at any iteration as the best solution.

Genetic Algorithm

- Based on evolution: Survival of the fittest.
- The fitness of a member is measured by the value of the objective function.
- So rather than testing one trial solution as a time, as with other methods, we test the whole population and see "who survives".

So 2 things can happen: Reproduction (2 parents make a child, or 2 trial solutions make a new trial solution that is a mix of the two)

Mutation → Change certain trial solutions such that they have "new" features that were not present in their parents.

Typical Outline:

- Start with an initial population of trial solutions (random, for example). Evaluate the fitness (value of cost function) for each
- Use a random process, biased towards the fittest members, and select an even amount to become parents.
- Each pair of parents will reproduce 2 children that will take over as new trial solutions. The features/genes of the children are a random mixture of the features of the parents.

(• A percentage of children will also be subject to mutations.)

- Repeat (note that population size has stayed changed)

If the newborn child is outside of the feasibility region, then we can look at it as a miscarriage. The parents "keep trying" until a child is born that is within the feasible solution space.

Note: You must select a ratio of normal reproduction to mutation (usually can't have too many mutations or you risk deviating too much from the optimal)

Stopping Rule : There are several options

Examples
- Max # of iterations
- Max run time of code
- Fixed # of consecutive ikrations without improvement

Use the best trial solution found on any iteration as the final solution.