Linear Programming

Discrete Mathematics and Optimization
Bioinformatics

The diet problem

- Three foods available: broccoli, milk, and oranges
- Institute for Health recommends per day at least
  3.7 liters water, 1.000mg calcium, 90mg vitamin C
- Table listing for each food the nutrient content and cost

Food 100g	Vitamin C	Calcium (mg)	Water (g)	Cost
Broccoli	107	47	91	0.381
Whole Milk	0	276	87	0.100
Oranges	53.2	40	87	0.272

Problem: compute a minimum cost diet satisfying the requirements

The diet problem

- Three foods available: broccoli, milk, and oranges
- Institute for Health recommends per day at least
  3.7 liters water, 1.000mg calcium, 90mg vitamin C
- Table listing for each food the nutrient content and cost

Food 100g	Vitamin C	Calcium (mg)	Water (g)	Cost
Broccoli	107	47	91	0.381
Whole Milk	0	276	87	0.100
Oranges	53.2	40	87	0.272

Problem: compute a minimum cost diet satisfying the requirements

- x_1, x_2, x_3 amount of units of each food.
- Minimize $0.381x_1 + 0.100x_2 + 0.272x_3$ subject to:

$$107x_1 + 53.2x_2 \geq 90$$
$$47x_1 + 276x_2 + 40x_3 \geq 1000$$
$$91x_1 + 87x_2 + 87x_3 \geq 3700$$

The Diet Problem

- n type of food items $F_1, \ldots, F_n$
- Each unit of F_j has a cost of c_j
- m nutrients (vitamin A, proteins, etc.) $N_1, \ldots, N_m$
  a_{ij} = amount of nutrient N_i in a unit of food F_j.

A healthy diet should contain at least b_i quantity of nutrient N_i

Goal:
Design a diet of minimum cost satisfying health requirements

Formalisation of a linear programing problem

In order to model it as a linear program one has to identify

1. **Decision variables**: x_i = number of units of F_i in the diet
2. **Explicit constraints**:
$$a_{i1}x_1 + \cdots + a_{in}x_n \geq b_i$$
where the left-hand side is the amount of N_i in the diet
3. **Implicit constraints**: $x_i \geq 0$

LP problem

Minimize	$c^T x$	(cost of the diet)
Subject to	$Ax \geq b$	(health requirements)
	$x \geq 0$	(non-negative amount of amount of food)

What is Linear Programming?

Optimizing a linear function subject to linear constraints.

$n =$ number of variables

$m =$ number of constraints

$x = (x_1, \ldots, x_n)$ is the vector of variables

A linear program in standard form is of the form

$$\begin{array}{ll} \text{Maximize} & c_1x_1 + \cdots + c_nx_n \\ \text{Subject to} & a_{11}x_1 + \cdots + a_{1n}x_n \leq b_1 \\ & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \leq b_m \\ & x_1, \ldots, x_n \geq 0 \end{array}$$

Alternatively, it can be of the form

$$\begin{array}{ll} \text{Minimize} & c_1x_1 + \cdots + c_nx_n \\ \text{Subject to} & a_{11}x_1 + \cdots + a_{1n}x_n \geq b_1 \\ & \vdots \\ & a_{m1}x_1 + \cdots + a_{mn}x_n \geq b_m \\ & x_1, \ldots, x_n \geq 0 \end{array}$$

Examples. A simple example

Maximize $\quad 2x + 2y$

Subject to $\quad 2x + 3y \leq 12$

$3x + y \leq 10$

$x + y \leq 4.5$

$x, y \geq 0$

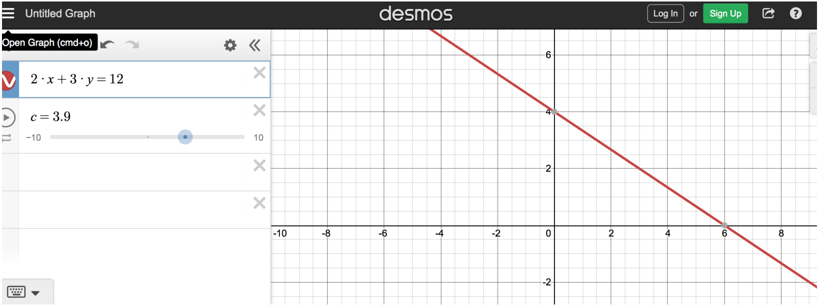
To visualize the regions: www.desmos.com/calculator

Examples. A simple example

Maximize $2x + 2y$

Subject to
$$2x + 3y \leq 12$$
$$3x + y \leq 10$$
$$x + y \leq 4.5$$
$$x, y \geq 0$$

To visualize the regions: www.desmos.com/calculator

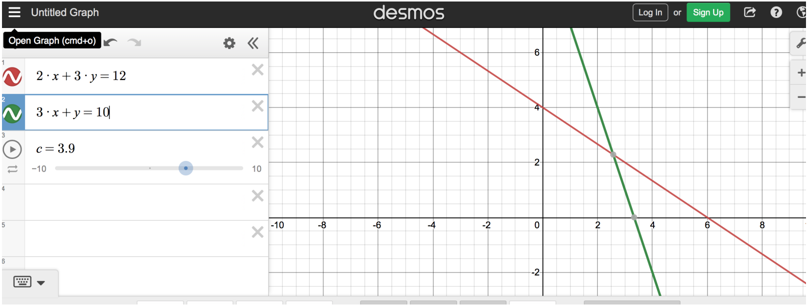


Examples. A simple example

Maximize $\quad 2x + 2y$

Subject to
$$2x + 3y \leq 12$$
$$3x + y \leq 10$$
$$x + y \leq 4.5$$
$$x, y \geq 0$$

To visualize the regions: www.desmos.com/calculator



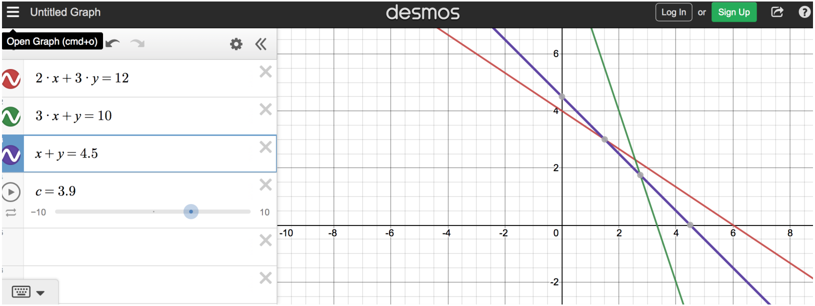
Examples. A simple example

Maximize $2x + 2y$

Subject to

$$2x + 3y \leq 12$$
$$3x + y \leq 10$$
$$x + y \leq 4.5$$
$$x, y \geq 0$$

To visualize the regions: www.desmos.com/calculator

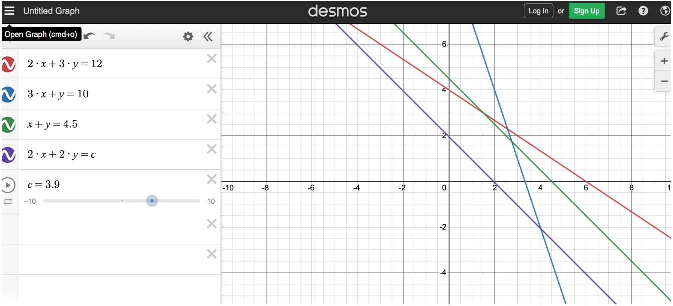


Examples. A simple example

Maximize $2x + 2y$

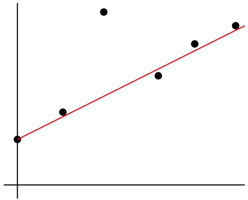
Subject to
$$2x + 3y \leq 12$$
$$3x + y \leq 10$$
$$x + y \leq 4.5$$
$$x, y \geq 0$$

To visualize the regions: www.desmos.com/calculator



Examples. Data fitting: least absolute deviations

n experimental data $(s_1, t_1), \ldots, (s_n, t_n)$
want to find the line $ax + b$ that fits best the data.



In this case the criterion is

$$\text{Minimize} \quad D(a, b) = \sum_{i=1}^{n} |as_i + b - t_i|.$$

This alternative to least squares is more resistant to outliers, but the function to be minimized is not differentiable.

Examples. Data fitting: least absolute deviations

In this case the criterion is

$$\text{Minimize} \quad D(a,b) = \sum_{i=1}^{n} |as_i + b - t_i|.$$

D is not linear (nor differentiable,) but...

$e_i = |as_i + b - t_i|$ = error for estimating the i-th point:

$$-e_i \leq as_i + b - t_i \leq e_i$$

$$\begin{array}{lll} \text{Minimize} & e_1 + \cdots + e_n & \text{(sum of absolute errors)} \\ \text{Subject to} & e_i \geq as_i + b - t_i & i = 1, \ldots, n \\ & e_i \geq -(as_i + b - t_i) & i = 1, \ldots, n \end{array}$$

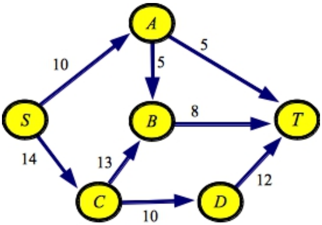
Not in standard form

Variables e_i are auxiliary, we only need a and b

Remark. Objective functions or constraints involving absolute values can often be handled by LP methods by introducing extra variables and/or extra constraints.

Examples. Flow in a network

Send flow through a network from a source to a target with constraints on the edges.



;

$$
\begin{array}{ll}
\text{Maximize} & x_{SA} + x_{SC} \\
\text{Subject to} & 0 \leq x_{SA} \leq 10, 0 \leq x_{SC} \leq 14, 0 \leq x_{AB} \leq 5, 0 \leq x_{AT} \leq 5 \\
& 0 \leq x_{CB} \leq 13, 0 \leq x_{CD} \leq 10, x_{BT} \leq 8, 0 \leq x_{DT} \leq 12 \\
& x_{SA} = x_{AB} + x_{AT}, x_{BT} = x_{AB} + x_{CB}, \\
& x_{SC} = x_{CB} + x_{CD}, x_{CD} = x_{DT}
\end{array}
$$

(flow from S

Equational form and basic feasible solutions

Linear programs in equational form

Maximize	$c^T x$
Subject to	$Ax = b$
	$x \geq 0$

- A is $m \times n$ of rank $m \leq n$,
- x is an n-column vector, and
- b is an m-column vector.

From a standard LP to an LP in equational form

Introduce slack variables. For each inequality

$$a_{i1}x_1 + \cdots + a_{in}x_n \leq b_i$$

introduce a new variable $y_i \geq 0$ such that

$$a_{i1}x_1 + \cdots + a_{in}x_n + y_i = b_i.$$

Basic feasible solutions

Maximize $c^T x$
Subject to $Ax = b$
$x \geq 0$

Notation: A_B = matrix of columns of A indexed by the subset $B \subset [n]$

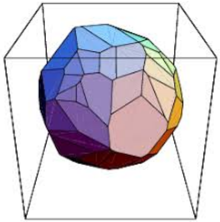
A basic feasible solution of the linear program is x such that, for some m–element subset $B \subset [n]$,

- x is a feasible solution,
- $x_i = 0$ for $i \notin B$, and
- A_B is nonsingular

Variables x_j with $j \in B$ are basic. The remaining variables are nonbasic.
A basic feasible solution is uniquely determined by B.

A geometric view

The feasible set of a linear program is a polyhedron P in $\mathbb{R}^n$, bounded or not.



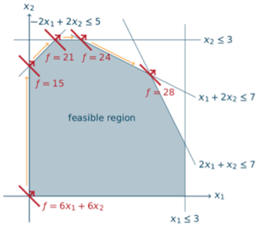


Figure 2 Linear programming problem - Simplex Solution

A vertex (a tip, a corner) of P is a point $v \in P$, such that there is a hyperplane H such that v is precisely the intersection of P and H.

Theorem

Let P be the set of feasible solutions of a linear program in equational form. Then v is a vertex of the polyhedron P if and only if v is a basic feasible solution of the linear program.

A geometric view

The feasible set of a linear program is a polyhedron P in $\mathbb{R}^n$, bounded or not.

Theorem

Let P be the set of feasible solutions of a linear program in equational form. Then v is a vertex of the polyhedron P if and only if v is a basic feasible solution of the linear program.

Theorem

An optimal solution of a linear program in equational form is achieved by a basic feasible solution, that is, in a vertex of the polyhedron P of feasible solutions.

This provides a (nonefficient) algorithm to find a solution to an LP problem: check the values of the objective function on vertices of the polyhedron of feasible solutions

The simplex method

Input: a linear program in equational form

$$\begin{array}{ll} \text{Maximize} & c^T x \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array}$$

Output: a solution x where the objective function is maximum or a certificate of nonexistence (unboundedness).

Description of the algorithm:

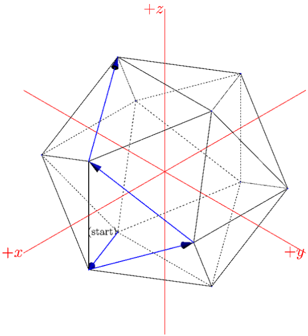
1. Find a basic feasible solution
2. Move to a neighboring basic feasible solution while increasing the value of the objective function
3. When no further increase is possible, the optimum is reached.

The simplex method

Input: a linear program in equational form

$$\begin{array}{ll} \text{Maximize} & c^T x \\ \text{Subject to} & Ax = b \\ & x \geq 0 \end{array}$$

Output: a solution x where the objective function is maximum or a certificate of nonexistence (unboundedness).



An example

$$
\begin{array}{ll}
\text{Maximize} & x_1 + x_2 \\
\text{Subject to} & -x_1 + x_2 \leq 1 \\
& x_1 \leq 3 \\
& x_2 \leq 2 \\
& x_1, x_2 \geq 0
\end{array}
$$

- Step 0: Put the problem in equational form.

  Introduce Slack variables x_3, x_4, x_5

$$
\begin{array}{lcccccl}
\text{Maximize} & x_1 & +x_2 & & & & \\
\text{Subject to} & -x_1 & +x_2 & +x_3 & & & = 1 \\
& x_1 & & & +x_4 & & = 3 \\
& & x_2 & & & +x_5 & = 2 \\
& x_1, & x_2, & x_3, & x_4, & x_5 & \geq 0
\end{array}
$$

An example (II)

$$\begin{array}{lrrrrrr}
\text{Maximize} & x_1 & +x_2 & & & & \\
\text{Subject to} & -x_1 & +x_2 & +x_3 & & & = 1 \\
 & x_1 & & & +x_4 & & = 3 \\
 & & x_2 & & & +x_5 & = 2 \\
 & x_1, & x_2, & x_3, & x_4, & x_5 & \geq 0
\end{array}$$

- Step 1a: Identify a basic feasible condition:
  - $(0, 0, 1, 3, 2)$ is a basic feasible solution.
  - x_3, x_4, x_5 basic variables, x_1, x_2 nonbasic
- Step 1b: Organize the input in a tableau (or dictionary) with these data:

$$\begin{array}{llrr|l}
x_3 & = 1 & +x_1 & -x_2 & \text{eq.}1 \\
x_4 & = 3 & -x_1 & & \text{eq.}2 \\
x_5 & = 2 & & -x_2 & \text{eq.}3 \\
\hline
z & = x_1 & +x_2 & & z = 0
\end{array}$$

$(0, 0, 1, 3, 2)$ basic feasible solution.

An example (III)

$$
\begin{array}{llll|l}
x_3 & = 1 & +x_1 & -x_2 & \text{eq.1} \\
x_4 & = 3 & -x_1 & & \text{eq.2} \\
x_5 & = 2 & & -x_2 & \text{eq.3} \\
\hline
z & = x_1 & +x_2 & & z = 0
\end{array}
$$

$(0, 0, 1, 3, 2)$ basic feasible solution.

- Step 2: Identify entering and leaving variables: pivot step
  - entering: one with positive coefficient in z (increasing its value increases the objective function).
    Two choices, x_1, x_2. Choose x_2 entering, for instance.
  - leaving: the one which restricts most the increasing of the entering variable.

$$
\begin{aligned}
x_3 \geq 0 &\rightarrow x_2 \leq 1 \\
x_4 \geq 0 &\rightarrow x_2 < \infty \\
x_5 \geq 0 &\rightarrow x_2 \leq 2
\end{aligned}
$$

x_3 leaving

An example (IV)

- Step 3: Increase the entering variable as much as possible ($x_2 = 1$), update the tableau

$$
\begin{array}{llll|l}
x_3 & = 1 & +x_1 & -x_2 & \\
x_4 & = 3 & -x_1 & & \\
x_5 & = 2 & & -x_2 & \\
\hline
z & = 0 & +x_1 & +x_2 & z = 0
\end{array}
\quad \to \quad
\begin{array}{llll|l}
x_2 & = 1 & +x_1 & -x_3 & \\
x_4 & = 3 & -x_1 & & \\
x_5 & = 1 & -x_1 & +x_3 & \\
\hline
z & = 1 & +2x_1 & -x_3 & z = 1
\end{array}
$$

$(0, 1, 0, 3, 1)$ basic feasible solution.

An example (V)

- Step 4: Repeat Step 2 and Step 3 while possible.
  - x_1 entering, x_5 leaving. Maximum increase $x_1 = 1$

$$\begin{array}{lllll|l} x_2 & = 1 & +x_1 & -x_3 & & \\ x_4 & = 3 & -x_1 & & & \\ x_5 & = 1 & -x_1 & +x_3 & & \\ \hline z & = 1 & +2x_1 & -x_3 & & z = 1 \end{array} \rightarrow \begin{array}{llll|l} x_1 & = 1 & +x_3 & -x_5 & \\ x_2 & = 2 & & -x_5 & \\ x_4 & = 2 & -x_3 & +x_5 & \\ \hline z & = 3 & +x_3 & -2x_5 & z = 3 \end{array}$$

$(1, 2, 0, 2, 0)$ basic feasible solution.

  - x_3 entering, x_4 leaving. Maximum increase $x_1 = 1$

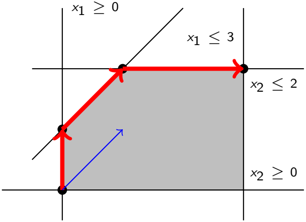
$$\begin{array}{llll|l} x_1 & = 1 & +x_3 & -x_5 & \\ x_2 & = 2 & & -x_5 & \\ x_4 & = 2 & -x_3 & +x_5 & \\ \hline z & = 3 & +x_3 & -2x_5 & z = 3 \end{array} \rightarrow \begin{array}{llll|l} x_1 & = 3 & & -x_4 & \\ x_2 & = 2 & & -x_5 & \\ x_3 & = 2 & -x_4 & +x_5 & \\ \hline z & = 5 & -x_4 & -x_5 & z = 5 \end{array}$$

$(3, 2, 2, 0, 0)$ basic feasible solution.

- Return $x_1 = 3, x_2 = 2$ and $z = 5$.

An example (VI): a geometric illustration

$$
\begin{array}{lrrrrrl}
\text{Maximize} & x_1 & +x_2 & & & & \\
\text{Subject to} & -x_1 & +x_2 & +x_3 & & & = 1 \\
& x_1 & & & +x_4 & & = 3 \\
& & x_2 & & & +x_5 & = 2 \\
& x_1, & x_2, & x_3, & x_4, & x_5 & \geq 0
\end{array}
$$



Outline of the simplex algorithm

0 Put the problem in equational form.
1 Find a basic feasible solution. Initialize the Tableau.
2 Pivot step: Select leaving and entering variables.
3 Increase entering variable. Update Tableau and current basic feasible solution.
4 Repeat steps 2 and 3 while possible.
5 Output solution.

The example again

Initialize

$$\begin{array}{llll|l} x_3 & = 1 & +x_1 & -x_2 & \\ x_4 & = 3 & -x_1 & & \\ x_5 & = 2 & & -x_2 & \\ \hline z & = 0 & +x_1 & +x_2 & z = 0 \end{array}$$

$(0, 0, 1, 3, 2)$

x_2 in; x_3 out

$$\begin{array}{llll|l} x_2 & = 1 & +x_1 & -x_3 & \\ x_4 & = 3 & -x_1 & & \\ x_5 & = 1 & -x_1 & +x_3 & \\ \hline z & = 1 & +2x_1 & -x_3 & z = 1 \end{array}$$

$(0, 1, 0, 3, 1)$

x_1 in; x_5 out

$$\begin{array}{llll|l} x_1 & = 1 & +x_3 & -x_5 & \\ x_2 & = 2 & & -x_5 & \\ x_4 & = 2 & -x_3 & +x_5 & \\ \hline z & = 3 & +x_3 & -2x_5 & z = 3 \end{array}$$

$(1, 2, 0, 2, 0)$

x_3 in ; x_4 out

$$\begin{array}{llll|l} x_1 & = 3 & & -x_4 & \\ x_2 & = 2 & & -x_5 & \\ x_3 & = 2 & -x_4 & +x_5 & \\ \hline z & = 5 & -x_4 & -x_5 & z = 5 \end{array}$$

$(3, 2, 2, 0, 0)$

Outline of the simplex algorithm

0 Put the problem in equational form.
1 Find a basic feasible solution. Initialize the Tableau.
2 Pivot step: Select leaving and entering variables.
3 Increase entering variable. Update Tableau and current basic feasible solution.
4 Repeat steps 2 and 3 while possible.
5 Output solution.

Outline of the simplex algorithm

Handling troubles

0 Put the problem in equational form.

1 Find a basic feasible solution. Initialize the Tableau.
Infeasibility: no solutions?

2 Pivot step: Select leaving and entering variables.
Rules of choice

3 Increase entering variable. Update Tableau and current basic feasible solution.
Degeneracy: what if no choice increases objective function?

4 Repeat steps 2 and 3 while possible.
Unboundeedness: what if no maximum exists?

5 Output solution.

Exception handling (I)

Unboundedness

$$
\begin{array}{lllll}
x_1 & = 1 & +x_2 & -x_3 \\
x_4 & = 3 & & -x_3 \\
\hline
z & = 1 & +x_2 & -x_3
\end{array}
$$

- Entering variable x_2: no variable restricts its increase.
- For any $t \geq 0$, $(1+t, t, 0, 3)$ is feasible solution with value $z = 1 + t$.

Output: No maximum for the objective function (or maximum is infinity).

Exception handling (I)

Unfeasibility

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax \leq b, x \geq 0 \end{aligned}$$

If $b \geq 0$, introduction of m slack variables always allows for an initial basic feasible solution.

But:

$$\begin{aligned} \text{Maximize} \quad & c^T x \\ \text{Subject to} \quad & Ax = b, x \geq 0 \end{aligned}$$

Exception handling (II)

Infeasibility

$$\text{Maximize } \; c^T x$$
$$\text{Subject to } \; Ax = b, x \geq 0$$

- Assume all $b_i \geq 0$
- Introduce an auxiliary variable y_i for each equation.
  (if we set all variables to 0, by how much they fail to satisfy the equation.)
- Write a Linear Programming problem for maximization of

$$-y_1 - y_2 \cdots - y_m$$

  - If maximum value is zero, then the values of the variables $x_1, \ldots, x_n$ provide a basic feasible solution to start the original problem.
  - Otherwise the program is infeasible.

Exception handling (II)

Unfeasibility: Example

$$\begin{array}{ll} \text{Maximize} & x_1 + 2x_2 \\ \text{Subject to} & x_1 + 3x_2 + x_3 = 4 \\ & 2x_2 + x_3 = 2 \\ & x_1, x_2 \geq 0 \end{array}$$

- Introduce auxiliary variables x_4 and x_5 and write new Linear Program problem:

$$\begin{array}{ll} \text{Maximize} & -x_4 - x_5 \\ \text{Subject to} & x_1 + 3x_2 + x_3 + x_4 \qquad = 4 \\ & 2x_2 + x_3 \qquad + x_5 = 2 \\ & x_1, x_2 x_3, x_4, x_5 \geq 0 \end{array}$$

Exception handling (II)

Unfeasibility: Example

$$
\begin{array}{ll}
\text{Maximize} & x_1 + 2x_2 \\
\text{Subject to} & x_1 + 3x_2 + x_3 = 4 \\
 & 2x_2 + x_3 = 2 \\
 & x_1, x_2 \geq 0
\end{array}
$$

- Basic feasible solution $(0, 0, 0, 4, 2)$. Write Tableau

$$
\begin{array}{l}
x_4 = 4 - x_1 - 3x_2 - x_3 \\
x_5 = 2 \qquad\;\; - 2x_2 - x_3 \\
\hline
z = -6 + x_1 + 5x_2 + 2x_3
\end{array}
$$

Exception handling (II)

Unfeasibility: Example

Maximize	$x_1 + 2x_2$
Subject to	$x_1 + 3x_2 + x_3 = 4$
	$2x_2 + x_3 = 2$
	$x_1, x_2 \geq 0$

- Start pivot steps
  - x_1 in x_4 out

$$\begin{aligned} x_1 &= 4 \quad - 3x_2 - x_3 - x_4 \\ x_5 &= 2 \quad - 2x_2 - x_3 \\ \hline z &= -2 + 2x_2 + x_3 - x_4 \end{aligned}$$

  - x_3 in x_5 out

$$\begin{aligned} x_1 &= 2 \quad - x_2 \quad - x_4 + x_5 \\ x_3 &= 2 \quad - 2x_2 \quad - x_5 \\ \hline z &= 0 - x_4 - x_5 \end{aligned}$$

Exception handling (II)

Unfeasibility: Example

$$\begin{array}{ll}\text{Maximize} & x_1 + 2x_2 \\ \text{Subject to} & x_1 + 3x_2 + x_3 = 4 \\ & 2x_2 + x_3 = 2 \\ & x_1, x_2 \geq 0\end{array}$$

- Basic feasible solution $(2, 0, 2, 0, 0)$, $z = 0$

$$\begin{array}{rcl} x_1 &=& 2 - x_2 - x_4 + x_5 \\ x_3 &=& 2 - 2x_2 - x_5 \\ \hline z &=& 0 - x_4 - x_5 \end{array}$$

- Basic feasible solution for initial problem $(2, 0, 2)$

Exception handling (III)

Degeneracy

In pivot step no increase of entering variable is possible

Allow for some zero increase of the objective function

Example

$$\begin{array}{ll} \text{Maximize} & x_2 \\ \text{Subject to} & -x_1 + x_2 \leq 0 \\ & x_1 \leq 2 \\ & x_1, x_2 \geq 0 \end{array}$$

Tableau:

$$\begin{array}{lllll} x_3 & = & & +x_1 & -x_2 \\ x_4 & = 2 & & -x_1 & \\ \hline z & = & & & +x_2 \end{array}$$

x_2 enters and x_3 leaves, but the value of x_2 can not be increased

Exception handling (III)

Degeneracy

In pivot step no increase of entering variable is possible

Allow for some zero increase of the objective function

Example

$$
\begin{array}{ll}
\text{Maximize} & x_2 \\
\text{Subject to} & -x_1 + x_2 \leq 0 \\
& x_1 \leq 2 \\
& x_1, x_2 \geq 0
\end{array}
$$

- Degenerate pivot step x_2 in x_3 out. Basic feasible solution $(0, 0, 0, 2)$

$$
\begin{array}{llll}
x_2 & = & +x_1 & -x_3 \\
x_4 & = 2 & -x_1 & \\
\hline
z & = & x_1 & -x_3
\end{array}
$$

Exception handling (III)

Degeneracy

In pivot step no increase of entering variable is possible

Allow for some zero increase of the objective function

Example

Maximize	x_2
Subject to	$-x_1 + x_2 \leq 0$
	$x_1 \leq 2$
	$x_1, x_2 \geq 0$

- x_1 in and x_4 out. Basic feasible solution $(2, 2, 0, 0)$

$$\begin{array}{ll|l}
x_1 & = 2 & \quad\quad -x_4 \\
x_2 & = 2 & -x_3 \ -x_4 \\
\hline
z & = 2 & -x_3 \ -x_4
\end{array}$$

- Otput: $(2, 2)$, $z = 2$.

Exception handling (III)

Degeneracy

In pivot step no increase of entering variable is possible

Allow for some zero increase of the objective function

Example

$$
\begin{array}{ll}
\text{Maximize} & x_2 \\
\text{Subject to} & -x_1 + x_2 \leq 0 \\
 & x_1 \leq 2 \\
 & x_1, x_2 \geq 0
\end{array}
$$

- It may happen that there are more than one degenerate pivot steps.
- It may happen that the degenerate pivot steps cycle to some previous point.

Pivot rules

More than one possibility for the entering variable

- LARGEST COEFFICIENT. Choose variable with largest coefficient in z
- LARGEST INCREASE. Largest absolute improvement of z
- STEEPEST EDGE. Moves current feasible solution in a direction closest to c.
- BLAND'S RULE. Entering variable with smallest index. Same rule for leaving variable

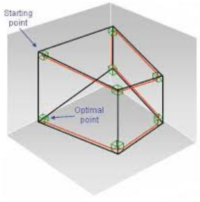
Theorem The simplex method with Bland's rule is always finite (cycling is impossible).

Efficiency of the simplex method

- The simplex algorithm ranges among the most frequently used in real life.

Efficiency of the simplex method

- The simplex algorithm ranges among the most frequently used in real life.
- In some examples it may need exponential running time (Klee-Minty examples, 1972)



Efficiency of the simplex method

- The simplex algorithm ranges among the most frequently used in real life.
- In some examples it may need exponential running time (Klee-Minty examples, 1972)
- In practice it performs very satisfactorily: typically between $2m$ and $3m$ steps to reach the optimum.

Efficiency of the simplex method

- The simplex algorithm ranges among the most frequently used in real life.
- In some examples it may need exponential running time (Klee-Minty examples, 1972)
- In practice it performs very satisfactorily: typically between $2m$ and $3m$ steps to reach the optimum.
- It has been proved that small alterations always brings polynomial time problems (Spielman-Teng, 2001).

Efficiency of the simplex method

- The simplex algorithm ranges among the most frequently used in real life.
- In some examples it may need exponential running time (Klee-Minty examples, 1972)
- In practice it performs very satisfactorily: typically between $2m$ and $3m$ steps to reach the optimum.
- It has been proved that small alterations always brings polynomial time problems (Spielman-Teng, 2001).
- Most mathematical software systems (Maple, Matlab, Mathematica,...) have packages implementing the simplex algorithm.
  In Python the library scipy.optimize.linprog has several implemented algorithms for LP problems
  `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html`

Summary of the simplex method

1. Convert the input linear program to equational form

$$\text{maximize } c^T x \text{ subject to } Ax = b \text{ and } x \geq 0$$

2. If no feasible basis is available, arrange for $b \geq 0$, and solve

$$\begin{array}{lll} \text{Maximize} & -(x_{n+1} + \cdots + x_{n+m}) & \qquad \overline{x} = (x_1, \ldots, x_{n+m}) \\ \text{Subject to} & \overline{A}\overline{x} = b & \qquad \overline{A} = (A \,|\, I_m) \\ & \overline{x} \geq 0 & \end{array}$$

If optimal value negative, the program is infeasible; **stop**.

Otherwise, the first n components of the optimal solution form a basic feasible solution of the original linear program.

3. For a feasible basis $B \subseteq \{1, 2, \ldots, n\}$ compute

$$\begin{array}{llll} X_B & = p & + & QX_N \\ \hline z & = z_0 & + & r^T X_N \end{array}$$

4. If $r \leq 0$,then return $(p, 0)$ as optimal solution; **stop**.

$$\begin{array}{lll} X_B & = p & + \quad Qx \\ \hline z & = z_0 & + \quad r^T X_n \end{array}$$

4. If $r \leq 0$,then return $(p, 0)$ as optimal solution; **stop**.

5. Else select entering variable x_v with positive coefficient in r

6. If the column of the entering variable x_v is nonnegative, the program is unbounded; **stop**.

7. Else select a leaving variable x_u.

In all rows where the coefficient of x_v is negative, divide the component of the vector p by that coefficient and change sign.

Choose the row with minimal ratio.

8. Replace the current basis B by $B - u + v$.

Update the tableau. Go to Step 4.

Simplex in Python

- Optimization library in scipy:
  docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html

Getting started | User Guide | **API reference** | Development | Release notes

Search the docs ...

(scipy.ndimage)

Orthogonal distance regression (scipy.odr)

Optimization and root finding (scipy.optimize)

Nonlinear solvers

Cython optimize zeros API

Signal processing (scipy.signal)

Sparse matrices (scipy.sparse)

Sparse linear algebra (scipy.sparse.linalg)

Compressed sparse graph routines (scipy.sparse.csgraph)

Spatial algorithms and data structures

linprog(method='simplex')

```
scipy.optimize.linprog(c, A_ub=None, b_ub=None, A_eq=None, b_eq=None, bounds=None, method='simplex', callback=None, options={'maxiter': 5000, 'disp': False, 'presolve': True, 'tol': 1e-12, 'autoscale': False, 'rr': True, 'bland': False}, x0=None)
```

Linear programming: minimize a linear objective function subject to linear equality and inequality constraints using the tableau-based simplex method.

Linear programming solves problems of the following form:

$$\begin{aligned} \min_x \ & c^T x \\ \text{such that } & A_{ub} x \leq b_{ub}, \\ & A_{eq} x = b_{eq}, \\ & l \leq x \leq u, \end{aligned}$$

where x is a vector of decision variables; c, b_{ub}, b_{eq}, l, and u are vectors; and A_{ub} and A_{eq}

Simplex in Python

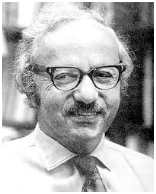
- Optimization library in scipy: docs.scipy.org/doc/scipy/reference/optimize.linprog-simplex.html

jupyter SimplexExample Last Checkpoint: 24 minutes ago (autosaved)

```
In [1]: from scipy.optimize import linprog

In [8]: c = [2, 2]
        A = [[2, 3], [3,1],[1,1]]
        b = [12,10, 4.5]
        x0_bounds = (0, None)
        x1_bounds = (0, None)

In [11]: res = linprog(c, A_ub=A, b_ub=b, bounds=[x0_bounds, x1_bounds],method='simplex')

In [12]: print(res)

     con: array([], dtype=float64)
     fun: 0.0
 message: 'Optimization terminated successfully.'
     nit: 5
   slack: array([12. , 10. ,  4.5])
  status: 0
 success: True
```

Concluding remarks

- The Simplex method was introduced by George Dantzig around 1947.



Concluding remarks

- The Simplex method was introduced by George Dantzig around 1947.
- Implementing the simplex method is far from being trivial.
  Pivot operations require computing the inverse of A_B. This is a costly an unstable computation and requires advanced techniques from matrix theory, such as LU decompositions.

Concluding remarks

- The Simplex method was introduced by George Dantzig around 1947.
- Implementing the simplex method is far from being trivial.
  Pivot operations require computing the inverse of A_B. This is a costly an unstable computation and requires advanced techniques from matrix theory, such as LU decompositions.
- There is a wide range of applications in bioinformatics of Linear Programming (and Integer Linear Programming), including sequence analysis, protein structure, haplotyping, microarrays, evolutionary distances...
  https://www.researchgate.net/publication/26552925_Mathematical_Programming_in_Computational_Biology_an_Annotated_Bibliography