Computational practicum
Lecture 5
Numerical Optimization - Linear & Mixed-Integer Linear Programming

Zoë J.G. Gromotka, Ferdinand Grozema, Artur M. Schweidtmann, Tanuj Karia

Computational Practicum
Dept. Chemical Engineering
Delft University of Technology

TU Delft
Process Intelligence RESEARCH
Delft Institute of Applied Mathematics

Wednesday, 10 December 2025

Recap

- In the previous lecture, we covered:
  - Fundamentals of mathematical optimization – definitions
  - Optimality conditions (1st order sufficient and 2nd order sufficient/necessary conditions)
  - Methods to solve unconstrained optimization (steepest descent & Newton method)

Learning goals of this lecture

- After successfully completing this lecture, you are able to. . . .
  - formulate a linear and mixed-integer linear programming model from an engineering problem statement
  - explain at a high-level different algorithms used to solve linear and mixed-integer linear programming models
  - analyze optimal solutions obtained on solving these models

- Note: No need to be able to implement corresponding solution algorithms from scratch.

Outline

- **Anatomy of an optimization problem**
- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - Analyzing optimal solutions
- **Mixed-Integer Linear Programming (MILP)**
  - Fundamentals
  - Formulating a MILP from an engineering problem statement

Outline

- **Anatomy of an optimization problem**
- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - Analyzing solutions
- **Mixed-Integer Linear Programming (MILP)**
  - Fundamentals
  - Formulating a MILP from an engineering problem statement

Nonlinear unconstraint optimization problem (as discussed in Q2 Lecture 4)

$$\min_{\boldsymbol{x}} f\,(\boldsymbol{x})$$

Objective function

Number of variables: n

Degrees of freedom: n

Chemical engineering challenges often come with constraints - our task is to optimize within these boundaries.

Join the Vevox session

Go to **vevox.app**

Enter the session ID: **191-696-711**

Or scan the QR code

What constraints exist in chemical engineering applications?

What constraints exist in chemical engineering applications?

Anatomy of a mathematical optimization problem

Typically, mathematical optimization problem consists of:

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$$

Objective function

$$\text{s.t. } \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = 0$$

Equality constraints

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0$$

Inequality constraints

$$\boldsymbol{x} \in [\boldsymbol{x}^{\mathrm{L}}, \boldsymbol{x}^{\mathrm{U}}], \boldsymbol{y} \in \{\boldsymbol{y}^{\mathrm{L}}, \ldots, \boldsymbol{y}^{\mathrm{U}}\}$$

Variable bounds

$$\boldsymbol{x} \in \mathbb{R}^{n}, \boldsymbol{y} \in \mathbb{Z}^{k}$$

$$f\colon \mathbb{R}^{n+k} \to \mathbb{R},$$

$$\boldsymbol{h}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_h},$$

$$\boldsymbol{g}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_g},$$

Number of variables: $n + k$

Number of equality constraints: n_h

Number of inequality constraints: n_g

Degrees of freedom: $n + k - n_h$

Equality constraints, $\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = 0$

- A set or a vector of n_h equality constraints → typically defines the mathematical model
- The model equations can be:
  - Linear or nonlinear
  - Algebraic or differential (ordinary or partial) or integral equations
- Determines the degrees of freedom (DOF) of a mathematical optimization model

$$\text{DOF} = n + k - n_h$$

- For all optimization models, degrees of freedom must always be greater than zero.
- Examples:
  - Mass and energy balances
  - Kinetic models determining the rate of a reaction
  - Thermodynamic models providing values of properties of interest

Inequality constraints, $g(x, y) \leq 0$

- Inequality constraints constraint the values that variables can possibly take
- Generally, it is a vector of n_g equations
- **Feasible region** defines the region where all constraints to the optimization problem are satisfied
- Equivalence: $\mathbf{g(x, y) \leq 0 \equiv -g(x, y) \geq 0}$

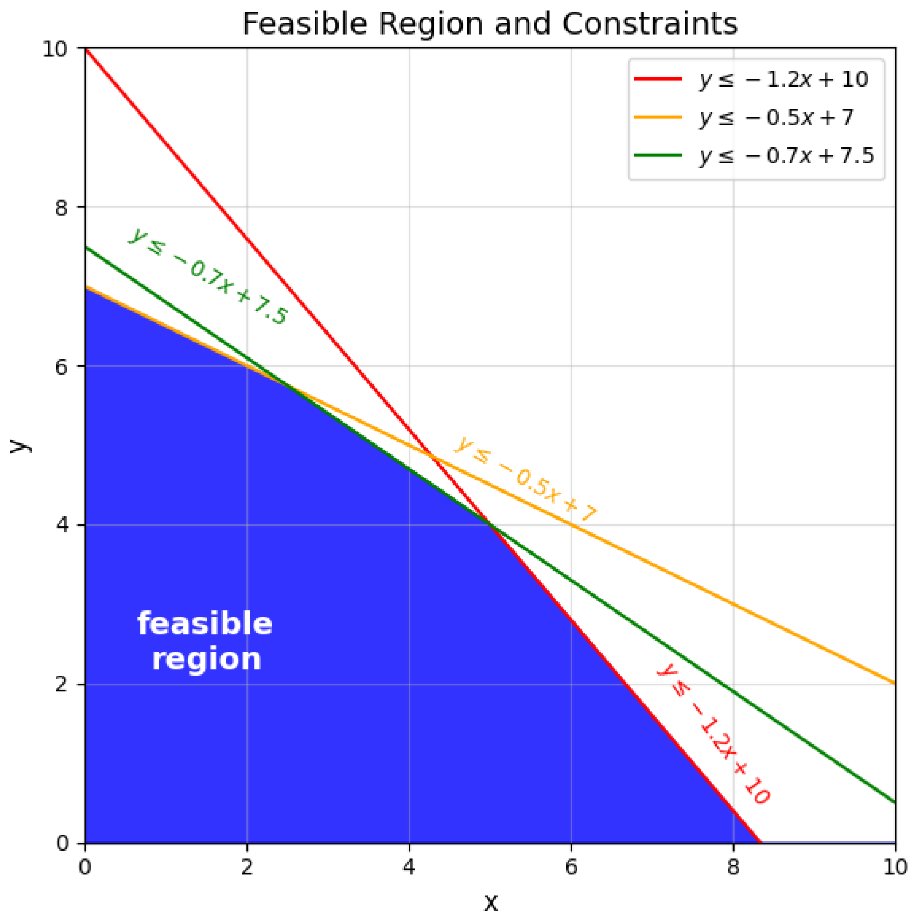


Figure: https://en.wikipedia.org/wiki/Feasible_region#/media/File:Linear_Programming_Feasible_Region.svg

Convexity of a set

The definition of convexity of a function (c.f. Q2 Lecture 4) is different from the definition of convexity for a set

- For a set of constraints defining the feasible set $\Omega = \{\boldsymbol{x} \in D \mid \boldsymbol{g}(\boldsymbol{x}) \leq 0, \boldsymbol{h}(\boldsymbol{x}) = 0\}$
- The set Ω is convex if $\forall\, \boldsymbol{x_1}, \boldsymbol{x_2} \in \Omega$ and $\forall\, \alpha \in [0,1]$ $\alpha \boldsymbol{x_1} + (1-\alpha)\boldsymbol{x_2} \in \Omega$
- Convexity of sets is challenging to establish
- **Sufficient condition for convexity**:
  If equality constraints $\boldsymbol{h}(\boldsymbol{x})$ are linear and $\boldsymbol{g}(\boldsymbol{x})$ are convex functions then Ω is convex
  → a limited case

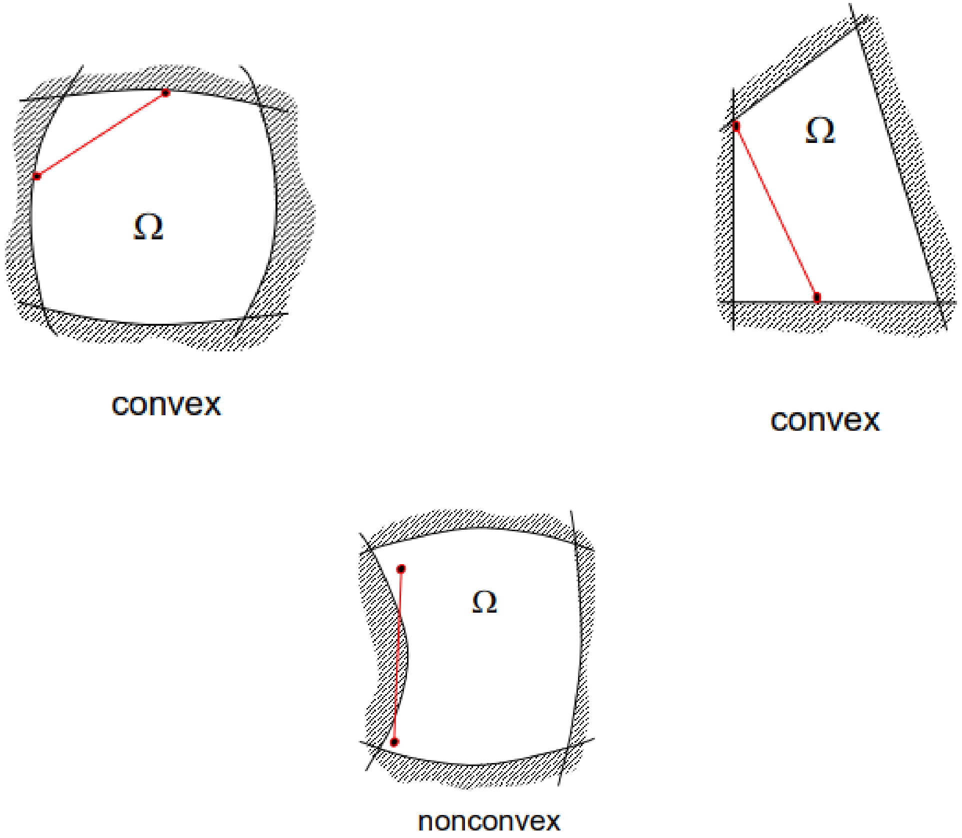


Figure taken from slides by Prof. Alexander Mitsos for his course on Applied Numerical Optimization (RWTH Aachen)

Types of optimization problems

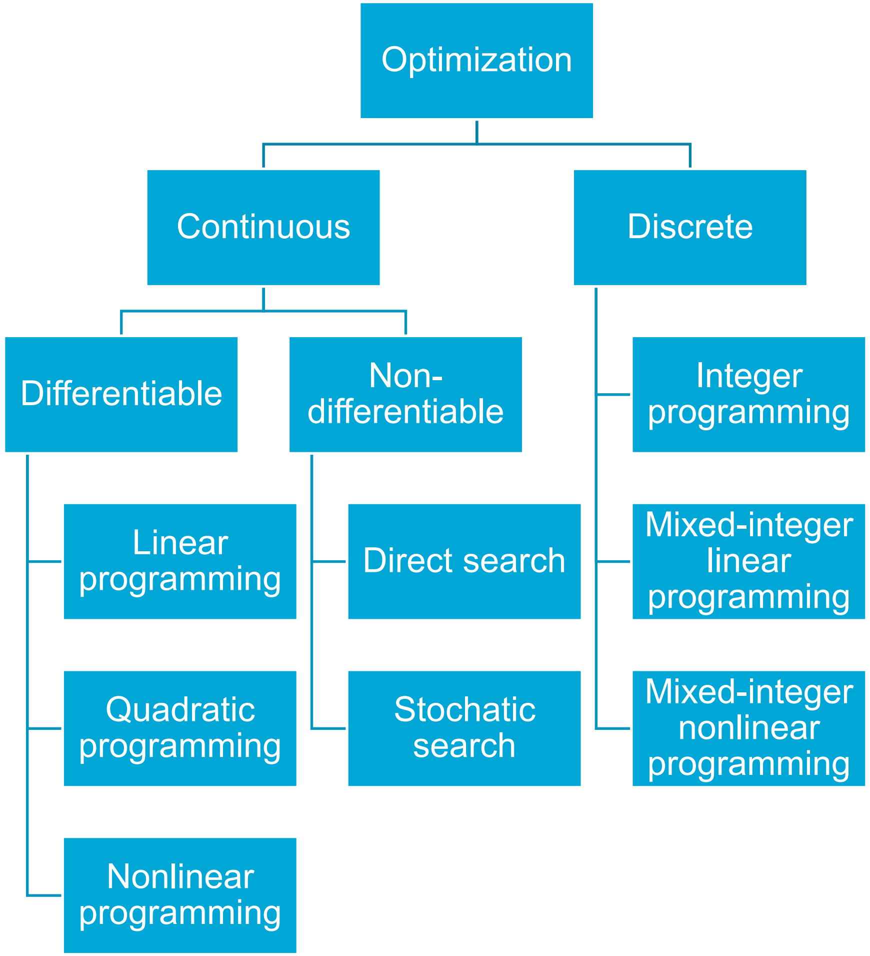


Figure adapted and simplified from https://neos-guide.org/guide/types/

Outline

- Anatomy of an optimization problem
- **Linear Programming (LP)**
  - **Fundamentals**
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - Analyzing solutions
- Mixed-Integer Linear Programming (MILP)
  - Fundamentals
  - Formulating a MILP from an engineering problem statement

What is linear programming (LP)

$$
\begin{aligned}
&\min_{\boldsymbol{x},\boldsymbol{y}} f\left(\boldsymbol{x},\boldsymbol{y}\right)\\
&\text{s.t. } \boldsymbol{h}\left(\boldsymbol{x},\boldsymbol{y}\right) = 0\\
&\quad \boldsymbol{g}\left(\boldsymbol{x},\boldsymbol{y}\right) \leq 0\\
&\quad \boldsymbol{x} \in \left[\boldsymbol{x}^{\mathrm{L}},\boldsymbol{x}^{\mathrm{U}}\right], \boldsymbol{y} \in \left\{\boldsymbol{y}^{\mathrm{L}},\ldots,\boldsymbol{y}^{\mathrm{U}}\right\}\\
&\quad \boldsymbol{x} \in \mathbb{R}^{n}, \boldsymbol{y} \in \mathbb{Z}^{k}\\
&\quad f\colon \mathbb{R}^{n+k} \to \mathbb{R},\\
&\quad \boldsymbol{h}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_h},\\
&\quad \boldsymbol{g}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_g},
\end{aligned}
$$

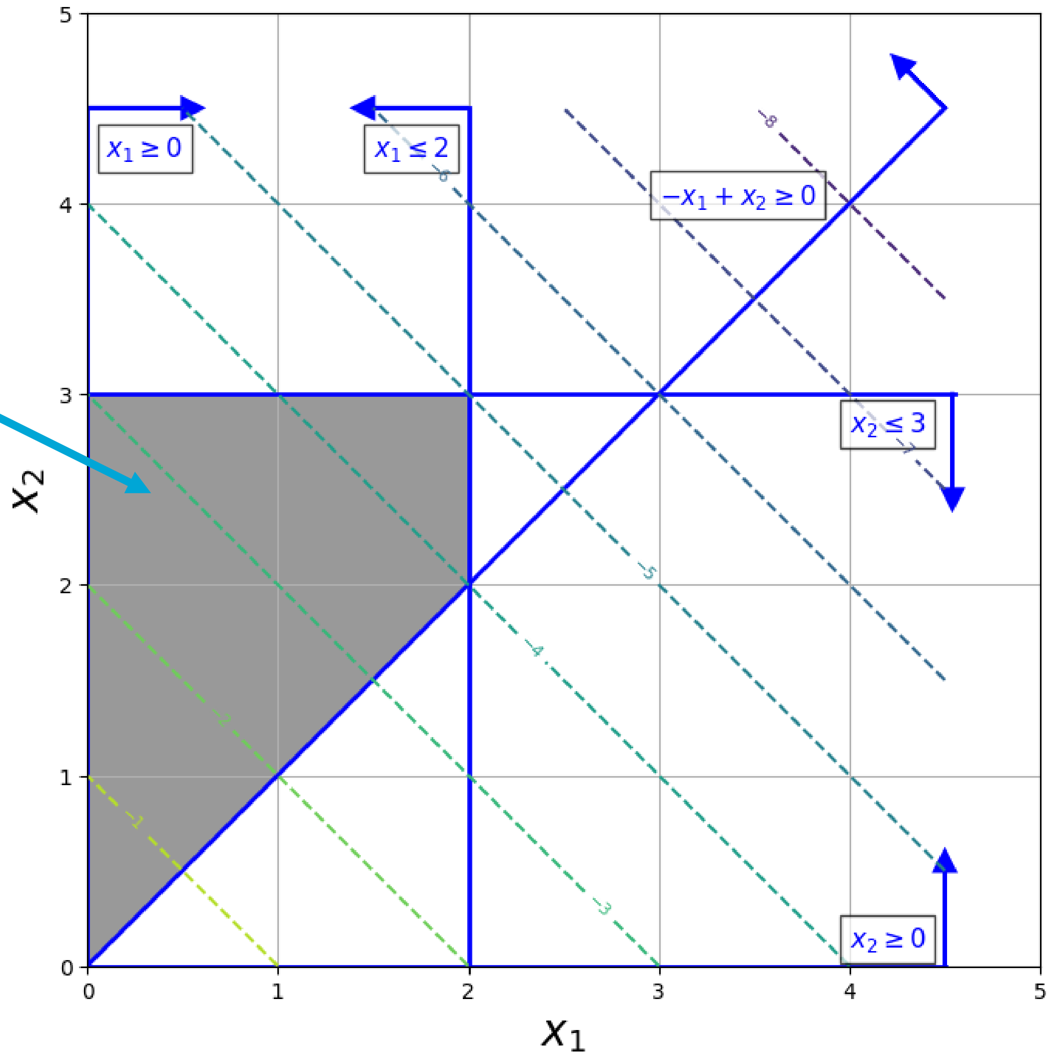
$$
\begin{aligned}
\min_{\mathbf{x}} \quad & f(\mathbf{x}) = \mathbf{c}^{\mathsf{T}}\mathbf{x}\\
\text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}\\
& \mathbf{x} \geq 0\\
& \mathbf{x} \in \mathbb{R}^{n}, \mathbf{c} \in \mathbb{R}^{n},\\
& \mathbf{b} \in \mathbb{R}^{n_g+n_h},\\
& \mathbf{A} \in \mathbb{R}^{(n_g+n_h)\times n}
\end{aligned}
$$

- Formulation above represents the standard form of LP
- All inequalities reformulated to equalities by introducing slack variables: $\mathbf{g}(\mathbf{x}) + \mathbf{s} = 0, \mathbf{s} \geq 0$

What does a feasible region look like?

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \geq 0 \\ & x_1 \leq 2 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

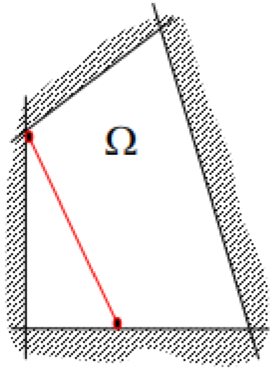
Feasible region



For LP problems, the feasible region is always a *polyhedron*

Fundamental properties of LPs

- Recall *sufficient condition* for a convex set:
  Given $\Omega = \{\boldsymbol{x} \in D \mid \boldsymbol{g}(\boldsymbol{x}) \leq 0, \boldsymbol{h}(\boldsymbol{x}) = 0\}$
- If equality constraints $\boldsymbol{h}(\boldsymbol{x})$ are linear and $\boldsymbol{g}(\boldsymbol{x})$ are convex functions then Ω is convex
- All linear functions are convex
- **→ all LPs are convex optimization problems**

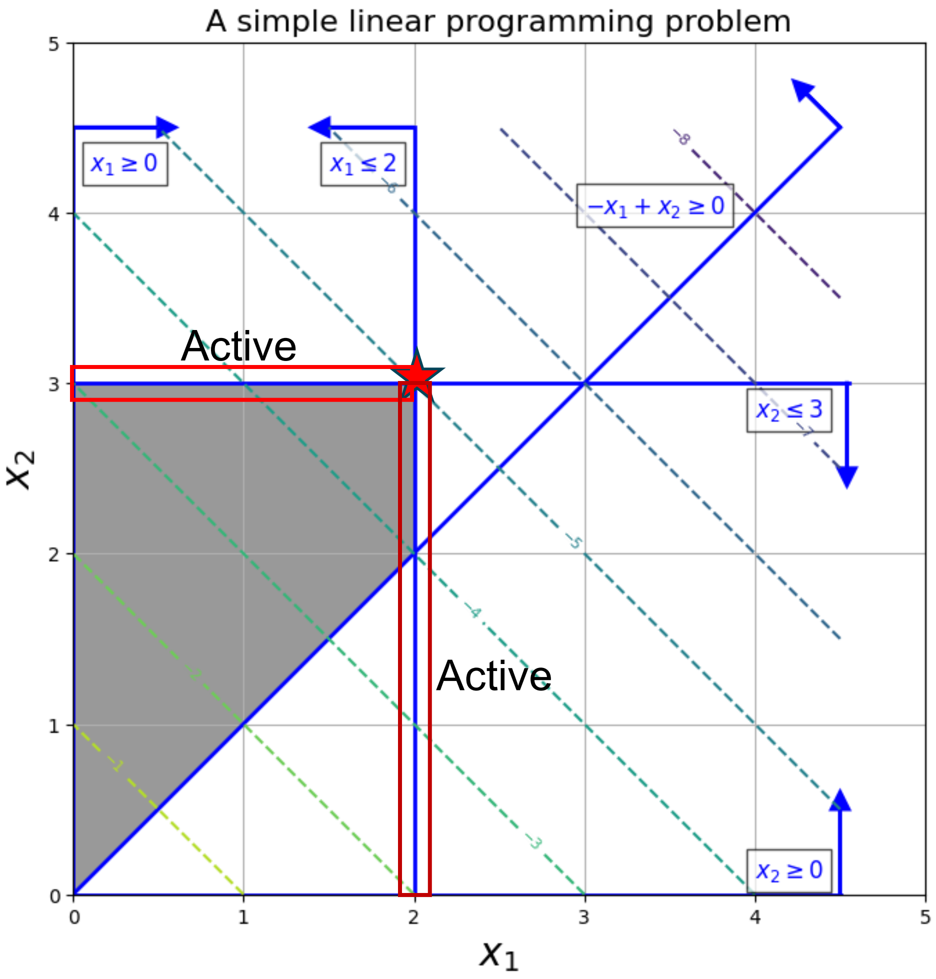


Convex feasible region

Figure taken from slides by Prof. Alexander Mitsos for his course on Applied Numerical Optimization (RWTH Aachen)

Where does the optimum lie?

- Since LPs are convex:
  - Every local optimum for an LP is a *global optimum*
  - If an LP has a non-constant objective function, then an optimal solution of an LP **cannot** lie in the interior of the feasible region
  - If an LP has a unique optimal solution, then the optimal **must** lie at an *extreme* point of the feasible region
  - If an LP has multiple optimal solutions, then one of them **must** occur at an *extreme* point of the feasible region



Outline

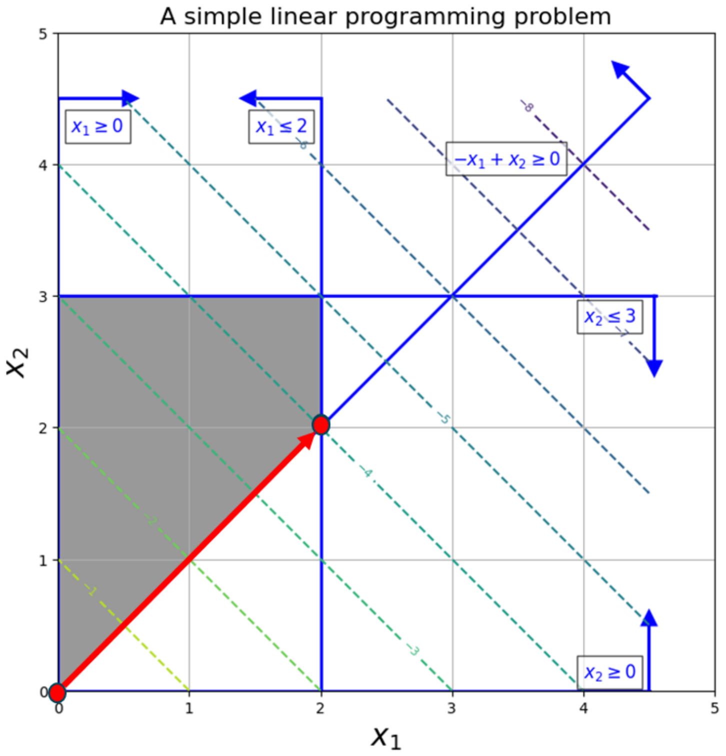
- Anatomy of an optimization problem
- **Linear Programming (LP)**
  - Fundamentals
  - **Solving LPs – conceptual overview of simplex and interior point algorithms**
  - Formulating linear programming models given an engineering problem statement
  - Analyzing solutions
- Mixed-Integer Linear Programming (MILP)
  - Fundamentals
  - Modelling using discrete variables
  - Branch and bound algorithms for solving MILPs

Solving LPs – general concept

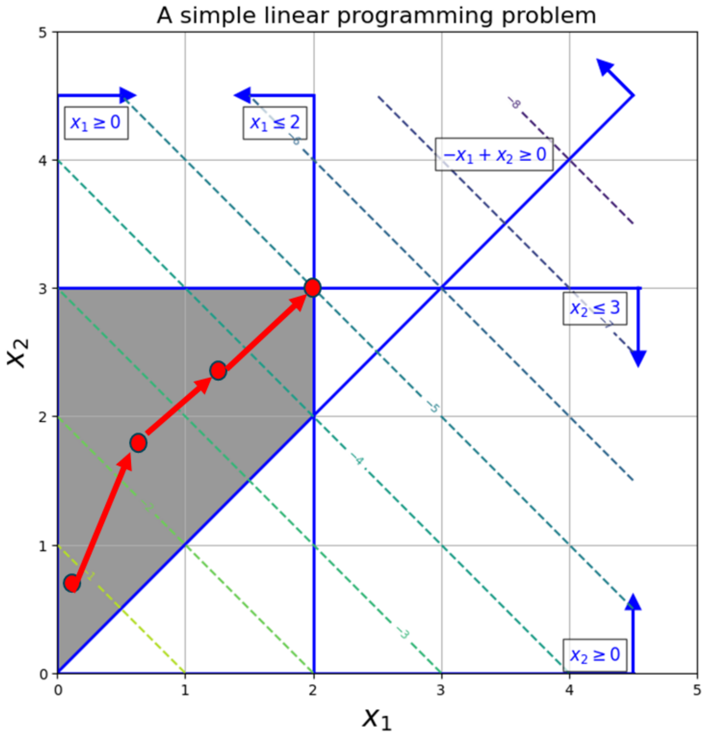
- Typically, the procedure to obtain the optimal solution of a linear programming problem involves:
  - Determining a feasible points in an iterative fashion such that the objective improves at every iteration
  - Terminating the search if no further improvement is possible
- This is popularly referred to as the *improving search principle*

Solving LPs – algorithms overview

- Simplex algorithm

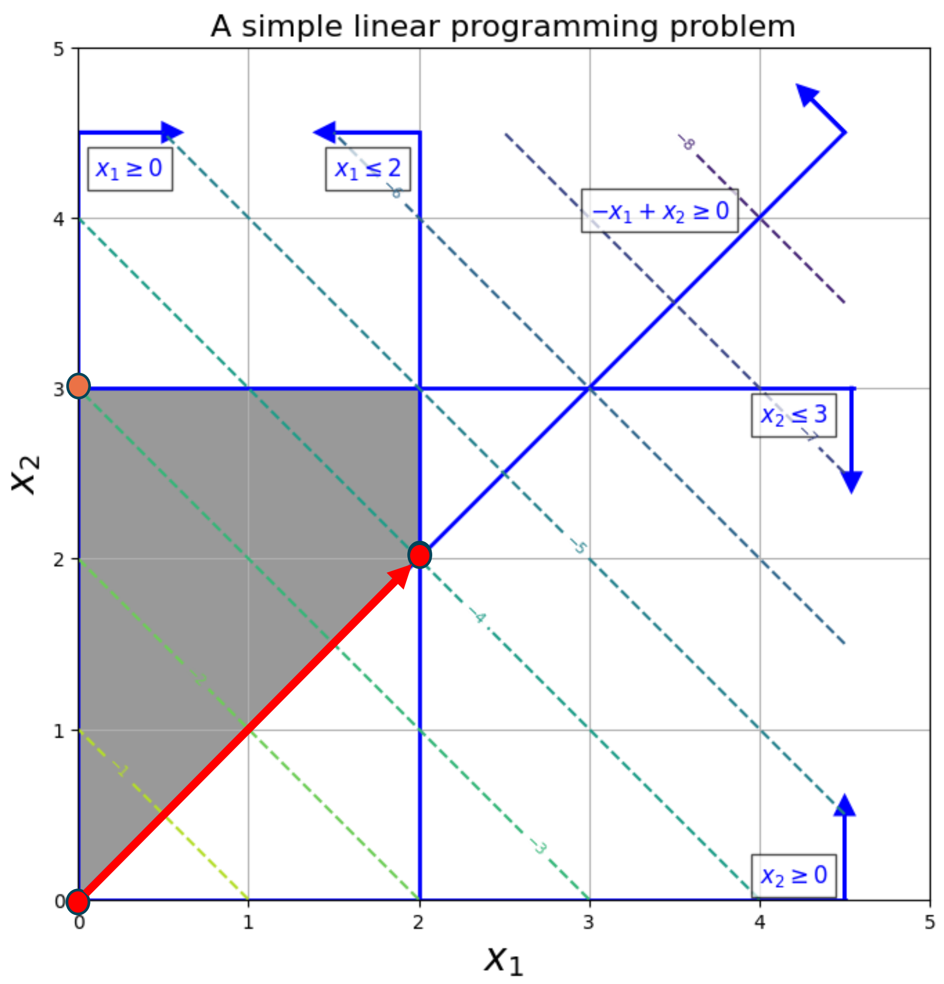


- Interior point algorihtm



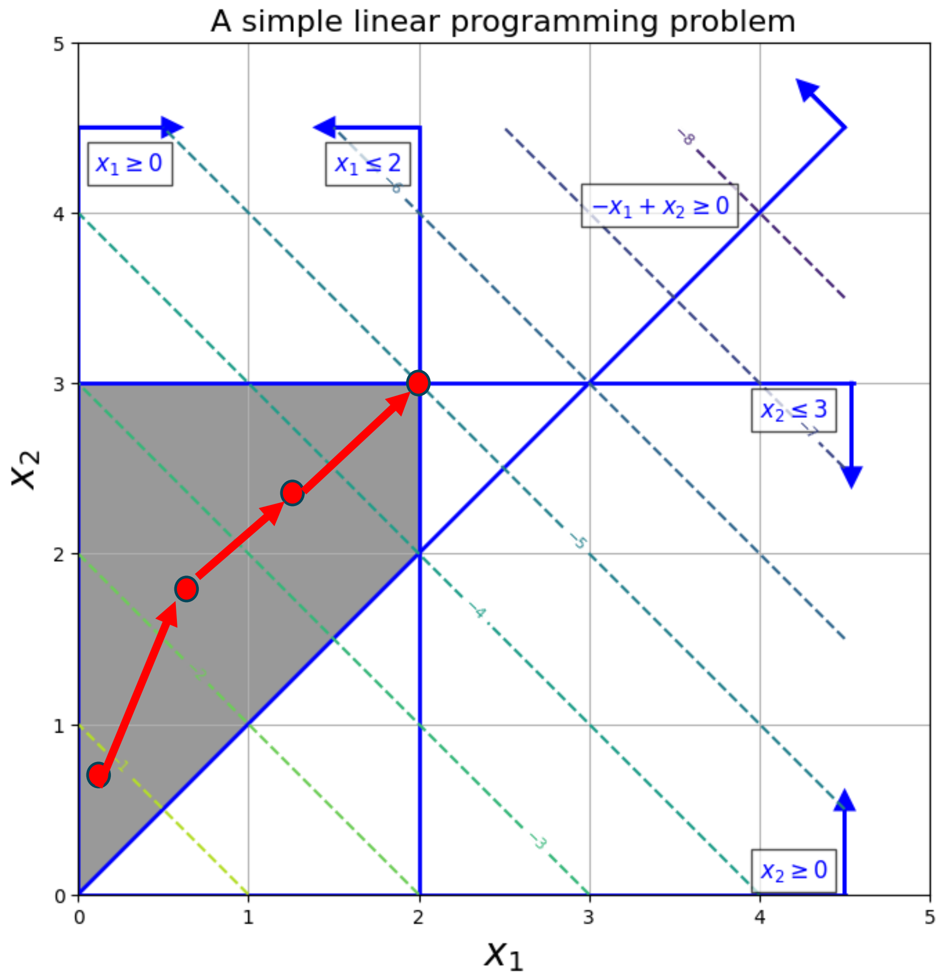
Simplex algorithm

- Only consider adjacent extreme points for improving direction
- Move along the edge that yield largest rate of improvement
- Move until another extreme point has been reached
- Check if further improvement is possible: if 'yes' *continue*; else *terminate*
- Can potentially struggle if there are a high number of extreme points
- High number of extreme points occur when there are many variables and constraints but few degrees of freedom



Interior point algorithm

- Determine an *improving* direction
- Move along that direction, but the feasible point in should *strictly* remain in the interior of the feasible region
- Check of significant change has been made: if ‘yes’ *continue*; else *terminate*
- Works well when degrees of freedom n is large
- For smaller problems simplex algorithm may be more efficient.



Linear programming in SciPy

- Function: Use `scipy.optimize.linprog` for Linear Programming (LP).
- Objective: Minimize $\boldsymbol{c}^T\boldsymbol{x}$ by passing the coefficient vector $\boldsymbol{c}$. For maximization, use - $\boldsymbol{c}$.
- Constraints:
  - Inequalities - ($\boldsymbol{A}_{ub}\boldsymbol{x} \leq \boldsymbol{b}_{ub}$): $\boldsymbol{A}_{ub}$ and $\boldsymbol{b}_{ub}$ represent coefficients of inequality constraints
  - Equalities - ($\boldsymbol{A}_{eq}\boldsymbol{x} = \boldsymbol{b}_{eq}$): $\boldsymbol{A}_{eq}$ and $\boldsymbol{b}_{eq}$ represent coefficients of equality constraints.
- Bounds: Define variable ranges with bounds using. Defaults to non-negative variables.
- Solvers: Default is '**highs**', with alternatives for specific needs.
- Result: Check `result.success`, and retrieve `result.x` (solution) and `result.fun` (optimal value).

$$
\begin{aligned}
\min_{\mathbf{x}} \ & \mathbf{c}^\mathsf{T}\mathbf{x} \\
\text{s.t.} \ & \mathbf{A}_{ub}\mathbf{x} \leq \mathbf{b_{ub}} \\
& \mathbf{A}_{eq}\mathbf{x} = \mathbf{b}_{eq} \\
& \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}
\end{aligned}
$$

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html

Implementing a LP in SciPy

$$\begin{aligned} \min \quad & -x_1 - x_2 \\ \text{s.t.} \quad & -x_1 + x_2 \geq 0 \\ & x_1 \leq 2 \\ & x_2 \leq 3 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Order used is $[x_1, x_2]$

$$\boldsymbol{c}^T := [-1, -1]$$

$$\boldsymbol{A}_{ub} := \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}; \ \boldsymbol{b}_{ub} := [0, 2, 3]$$

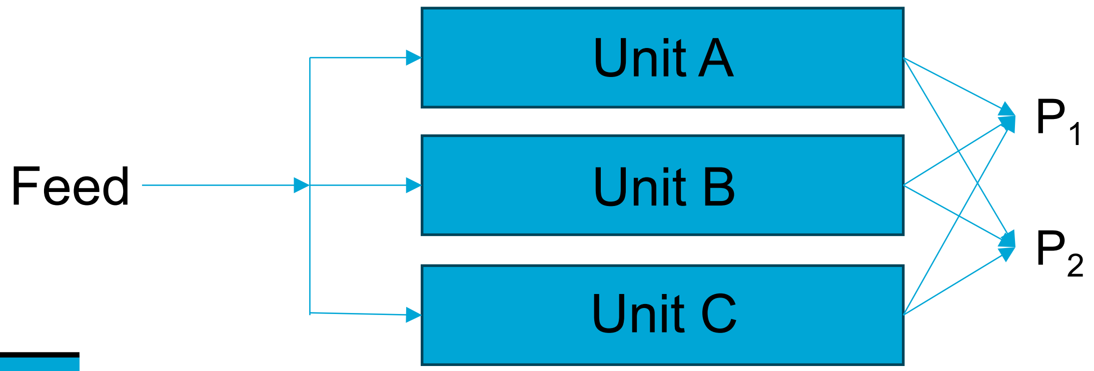
```python
from scipy.optimize import linprog

# Minimize: c^T x
c = [-1, -1]  # Coefficients for the objective function
A = [[1, -1], [1, 0], [0,1]]  # Coefficients for inequality constraints
b = [0, 2, 3]  # Bounds for inequality constraints
x_bounds = [(0, None), (0, None)]  # Variable bounds

# Solve the LP
result = linprog(c, A_ub=A, b_ub=b, bounds=x_bounds, method='highs')
```

Outline

- Anatomy of an optimization problem
- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - **Formulating linear programming models given an engineering problem statement**
  - Analyzing solutions
- Mixed-Integer Linear Programming (MILP)
  - Fundamentals
  - Formulating a MILP from an engineering problem statement

Multi-product problem (1/2) – General problem setup

- Feed to three units is split into three streams: F_A, F_B and F_C.
- Two products are produced: P_1 and P_2,
- Each unit has individual product yields



Yield (wt %)	Unit A	Unit B	Unit C
P_1	40	30	50
P_2	60	70	50

Multi-product problem (2/2) - Additional constraints

- We have some capacity limitations:
  - The plant can handle a maximum feed F of 10,000 kg/day
  - Each unit (A, B and C) cannot handle more than 5,000 kg/day
  - The maximum demand of P_1 and P_2 are 4,000 kg/day and 5,000 kg/day

Costs of feed and product streams

Stream	Value ($/kg)
F	0.40
P_1	0.60
P_2	0.30

Determine the optimal values of F_A, F_B and F_C that maximize the daily profit of the plant?

Formulating the objective function

- Objective function: maximize the profit by selling the products P_1 and P_2
- The profit of the plant is the difference between the amount we earn by selling the products and the amount spent in buying the feed F
- Amount earned by selling the products: unit cost of the product × amount of product produced
- Similarly, amount spent in buying feed: unit cost of the feed × amount of feed bought
- Mathematically,

$$\text{Profit} = C_{P1}P_1 + C_{P2}P_2 - C_F F$$

- P_1, P_2 and F represent the amounts of the products and the feed respectively in kg.
- We know $C_{P1} = 0.6$, $C_{P2} = 0.3$, and $F = 0.4$
- Recall the equivalence of objective function (c.f. Q2 Lecture 4): maximizing profit is equivalent to minimizing negative profit
- Objective function: $-\text{Profit}$

Modeling the material balances

- Feed F is split and sent to three units as $F_A,\ F_B,$ and F_C. Hence,

$$F = F_A + F_B + F_c$$

- Given the yield of each unit, we model the conversion of feed to the products in each unit A, B and C and sum up over all reactors

Yield (wt %)	Unit A	Unit B	Unit C
P_1	40	30	50
P_2	60	70	50

$$P_1 = 0.4F_A + 0.3F_B + 0.5F_C$$

$$P_2 = 0.6F_A + 0.7F_B + 0.5F_C$$

Accounting for capacity and demand

- The maximum capacity of each unit is 5000 kg/day. Hence,

$$F_A, F_B, F_C \leq 5000$$

- Overall, the plant can handle a maximum of 10000 kg/day of feed.

$$F \leq 10000$$

- Due to known demand, we have a maximum limit of the products we can produce giving us an upper bound

$$P_1 \leq 4000 \text{ and } P_2 \leq 5000$$

- All material flows are non-negative,

$$F, F_A, F_B, F_C, P_1, P_2 \geq 0$$

The linear programming formulation

$$
\begin{aligned}
\min\ & 0.4F - 0.6P_1 + -0.3P_2 \\
\text{s.t. } & \left.\begin{aligned}
F &= F_A + F_B + F_C \\
P_1 &= 0.4F_A + 0.3F_B + 0.5F_C \\
P_2 &= 0.6F_A + 0.7F_B + 0.5F_C
\end{aligned}\right\} \text{Equality constraints} \\
& \left.\begin{aligned}
F &\leq 10000 \\
F_A &\leq 5000 \\
F_B &\leq 5000 \\
F_C &\leq 5000 \\
P_1 &\leq 4000 \\
P_2 &\leq 5000 \\
F, F_A, F_B, F_C, P_1, P_2 &\geq 0
\end{aligned}\right\} \text{Inequality constraints}
\end{aligned}
$$

Live coding: Solving the LP in SciPy

- Open Colab: Solving the formulated LP using SciPy

- Find more in the Github repository of the course: https://github.com/process-intelligence-research/computational_practicum_lecture_coding/tree/main

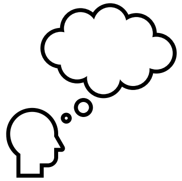
Outline

- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - **Analyzing solutions**
- Mixed-Integer Linear Programming (MILP)
  - Fundamentals
  - Modelling using discrete variables
  - Branch and bound algorithms for solving MILPs
- Introduction to genetic algorithms

Analyzing the optimal solution obtained

- The optimal values are:

Variable	Value
Profit ($/day)	325.0
F (kg/day)	8750.0
F_A (kg/day)	3750.0
F_B (kg/day)	0.0
F_C (kg/day)	5000.0
P_1 (kg/day)	4750.0
P_2 (kg/day)	5000.0



Understanding the output generated (1/5)

```
       message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
       success: True
        status: 0
           fun: -325.0
             x: [ 8.750e+03  3.750e+03  0.000e+00  5.000e+03  4.000e+03
                  4.750e+03]
           nit: 2
         lower:  residual: [ 8.750e+03  3.750e+03  0.000e+00  5.000e+03
                             4.000e+03  4.750e+03]
                marginals: [ 0.000e+00  0.000e+00  2.500e-02  0.000e+00
                             0.000e+00  0.000e+00]
         upper:  residual: [       inf        inf        inf        inf
                                   inf        inf]
                marginals: [ 0.000e+00  0.000e+00  0.000e+00  0.000e+00
                             0.000e+00  0.000e+00]
         eqlin:  residual: [ 0.000e+00  0.000e+00  0.000e+00]
                marginals: [ 4.000e-01 -5.500e-01 -3.000e-01]
       ineqlin:  residual: [ 1.250e+03  1.250e+03  5.000e+03  0.000e+00
                             0.000e+00  2.500e+02]
                marginals: [-0.000e+00 -0.000e+00 -0.000e+00 -2.500e-02
                            -5.000e-02 -0.000e+00]
mip_node_count: 0
mip_dual_bound: 0.0
       mip_gap: 0.0
```

Understanding the output generated (2/5)

- Printing the result object gives a more detailed insight into the solution
- First part of the log shows whether the solver terminated successfully or not.
- **Always** check first if the solver terminated successfully or not before analyzing the solution reported

```
message: Optimization terminated
successfully. (HiGHS Status 7:
Optimal)

success: True

status: 0
```

Understanding the output generated (3/5)

- `fun` reports the value of the objective function of your problem
- x is a vector of the decision variables in your problem and SciPy reports their values in the order you defined
- In the live coding, we defined them in the order:

  ```
  x := [F, F_A, F_B, F_C, P1, P2]
  ```

- nit represents the number of iterations the solver required to optimize the problem

```
fun: -325.0


x: [ 8.750e+03  3.750e+03  0.000e+00
5.000e+03  4.000e+03 4.750e+03]

nit: 2
```

Understanding the output generated (4/5)

- Here, the information about the lower and upper bounds of the variables is reported
- Residual refers to the difference between the value at the optimal solution and the value of the respective bound
- Marginal refers to the dual cost or shadow price and is only non-zero if the value is at the bound → constraint is active
- Marginals tells us how will the value of the objective change on changing the limit of the bound (beyond the scope of this lecture)

```
lower:  residual: [ 8.750e+03
3.750e+03  0.000e+00  5.000e+03
4.000e+03  4.750e+03]

        marginals: [ 0.000e+00
0.000e+00  2.500e-02  0.000e+00
0.000e+00  0.000e+00]

upper:  residual: [       inf
inf        inf        inf inf
inf]

        marginals: [ 0.000e+00
0.000e+00  0.000e+00  0.000e+00
0.000e+00  0.000e+00]
```

Understanding the output generated (5/5)

- Here, marginals and residuals for the constraints are reported
- Equality constraints always have a zero residual
- Inequality constraint only have a zero residual if the constraint is active
- Marginal is non-zero if the residual is zero → constraint is active

```
eqlin:    residual: [ 0.000e+00
0.000e+00  0.000e+00]

          marginals: [ 4.000e-01 -
5.500e-01 -3.000e-01]

ineqlin:  residual: [ 1.250e+03
1.250e+03  5.000e+03  0.000e+00
0.000e+00  2.500e+02]

          marginals: [-0.000e+00 -
0.000e+00 -0.000e+00 -2.500e-02
-5.000e-02 -0.000e+00]
```

Outline

- **Anatomy of an optimization problem**
- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - Analyzing solutions
- **Mixed-Integer Linear Programming (MILP)**
  - **Fundamentals**
  - Formulating a MILP from an engineering problem statement

Mixed-Integer Linear Programming (MILP)

- Building on from linear programming, a mixed-integer linear program also comprises a set of integer (or discrete) variables
- Similar to linear programming, $f(\mathbf{x},\mathbf{y}), \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}), \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y})$ are all linear
- For the case when set of continuous variables $\boldsymbol{x}$ is not present → pure integer linear programming (ILP) problem

$$\min_{\boldsymbol{x},\boldsymbol{y}} f(\boldsymbol{x},\boldsymbol{y})$$

$$\text{s.t. } \boldsymbol{h}(\boldsymbol{x},\boldsymbol{y}) = 0$$

$$\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \leq 0$$

$$\boldsymbol{x} \in [\boldsymbol{x}^{\mathrm{L}},\boldsymbol{x}^{\mathrm{U}}], \boldsymbol{y} \in \{\boldsymbol{y}^{\mathrm{L}},\ldots,\boldsymbol{y}^{\mathrm{U}}\}$$

$$\boldsymbol{x} \in \mathbb{R}^{n}, \boldsymbol{y} \in \mathbb{Z}^{k}$$

$$f\colon \mathbb{R}^{n+k} \to \mathbb{R},$$

$$\boldsymbol{h}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_h},$$

$$\boldsymbol{g}\colon \mathbb{R}^{n+k} \to \mathbb{R}^{n_g},$$

The feasible region of mixed-integer problems

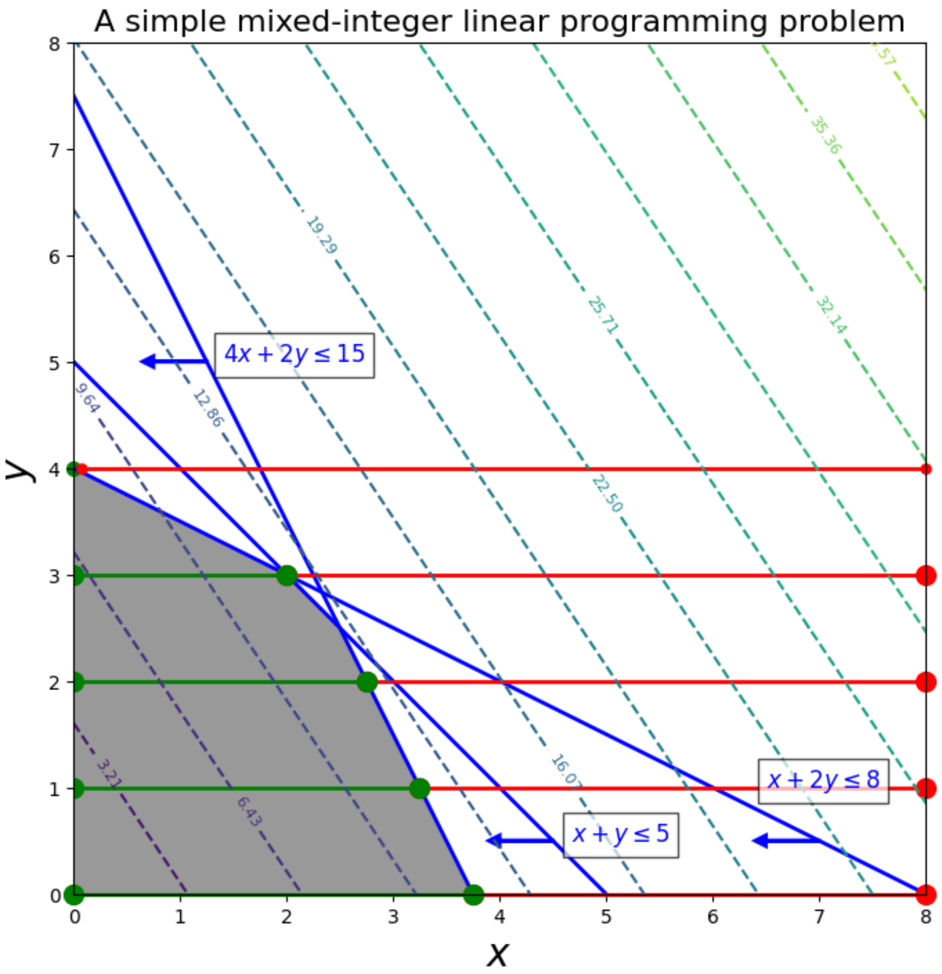
$$\max_{x,y} 3x + 2y$$

$$\text{s.t. } 4x + 2y \leq 15$$

$$x + 2y \leq 8$$

$$x + y \leq 5$$

$$x \in [0, 9], y \in \{0, 1, \ldots, 8, 9\}$$



The feasible region is discontinuous → *non-convex*

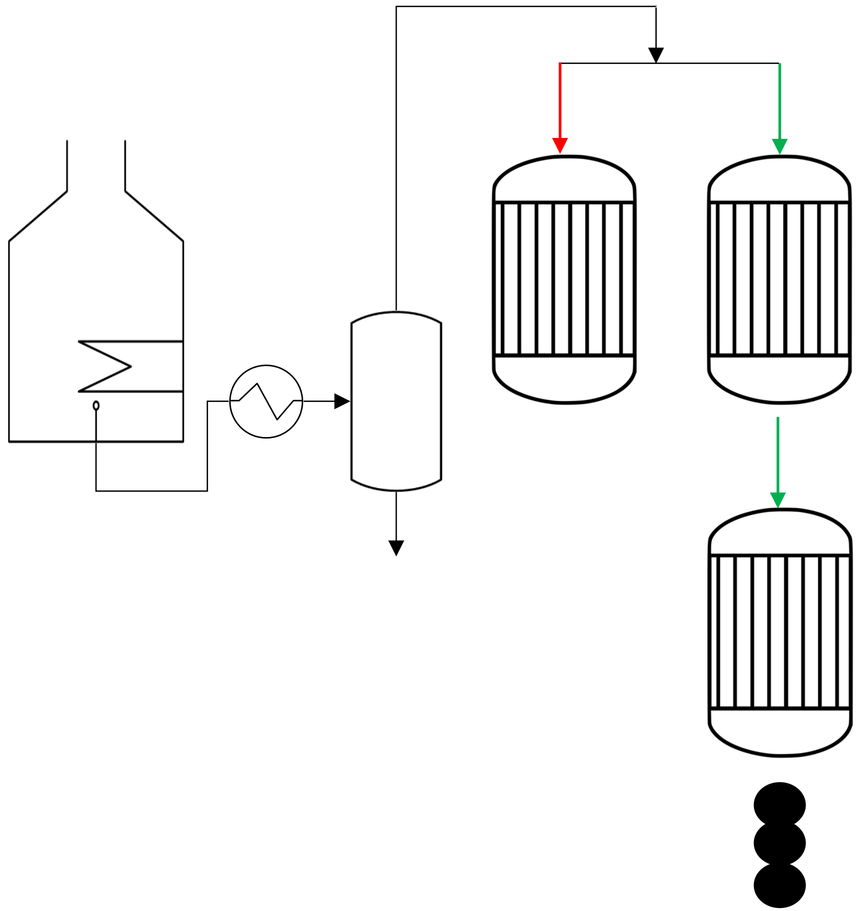
Since MILPs are inherently non-convex, they are more difficult to solve

Discrete variables

- Discrete variables are of two types:
  - **Binary variables** – Can only take values 0 or 1.

    Example: Should feed be sent to the reactor?
  - **Integer variables** – Can take any integer value (e.g., 0,1,2,3,4,5, . . .).

    Example: How many reactors are in series?
- **In this lecture, we will focus on binary decisions (without loss of generality)**



Modelling the logical proposition: OR

- We can model common logical propositions using binary variables
- Assuming J is a set representing multiple options, generally.
- We can model:
  - Selecting at least one component (logically equivalent to inclusive OR) → $\sum_{j \in J} y_j \geq 1$
  - Select exactly one component (logically equivalent to exclusive OR) → $\sum_{j \in J} y_j = 1$
  - Selecting at most one component → $\sum_{j \in J} y_j \leq 1$

Model the logical proposition: For separation you can either pick distillation (y_D) or absorption (y_A), not both

Model the logical proposition: For separation you can either pick distillation (y_D) or absorption (y_A), not both

Correct answers will be shown here

Modeling the logical proposition OR – truth table

Selecting at most one component $\sum_{j \in J} y_j \leq 1$

Pick absorption (y_A)	Pick distillation (y_D)	$y_A + y_D$
0	0	0
0	1	1
1	0	1
1	1	2

Equivalent to inclusive OR ($\sum_{j \in J} y_j \geq 1$)

Equivalent to exclusive OR ($\sum_{j \in J} y_j = 1$)

Modeling the logical propositions – AND & NOT

- Example of AND:
  - [Reactor (y_R)]AND [Separator (y_S)]→ $y_R \geq 1$, $y_s \geq 1$

- Example of NOT:
  - If y_A represents that absorption is selected, $1 - y_A$ represents absorption is not selected

Activating specific constraints only if needed

- Sometimes, we may want to activate certain constraint if a logical condition is met.
- This is particularly useful when values of continuous variables are linked to discrete variables
- Example:

You have two different reactor designs to consider. You can only choose one.

If reactor one is chosen, then pressure P must be between 5 and 10 atm.

If reactor two is chosen, then pressure P must be between 20 and 30 atm.

- How can we mathematically model this?

$$5 - M_1(1-y_1) \leq P \leq 10 + M_1(1-y_1)$$
$$20 - M_2(1-y_2) \leq P \leq 30 + M_2(1-y_2)$$
$$y_1 + y_2 = 1$$

Assuming $M_1 = 100$, $M_2 = 100$

$y_1 = 1$

$$5 - 0 \leq P \leq 10 + 0$$
$$20 - 100 \leq P \leq 30 + 100$$

P is bounded between 5 and 10

$y_1 = 0$

$$5 - 100 \leq P \leq 10 + 100$$
$$20 - 0 \leq P \leq 30 + 0$$

P is bounded between 20 and 30

This approach is called Big-M formulation

Example taken from the book - Biegler, L. T., Grossmann, I. E., & Westerberg, A. W. (1997). Systematic methods for chemical process design.

Choosing appropriate big-M values

- Big-M values should be sufficiently high
- If the Big-M value is too low, *feasible* solution becomes *infeasible*
- Consider the example on previous slide:

Assuming $M_1 = 5$, $M_2 = 5$

$$5 - M_1(1 - y_1) \leq P \leq 10 + M_1(1 - y_1)$$
$$20 - M_2(1 - y_2) \leq P \leq 30 + M_2(1 - y_2)$$
$$y_1 + y_2 = 1$$

$y_1 = 1$

$$5 - 0 \leq P \leq 10 + 0$$
$$20 - 5 \leq P \leq 30 + 5$$

$y_1 = 0$

$$5 - 5 \leq P \leq 10 + 5$$
$$20 - 0 \leq P \leq 30 + 0$$

P has inconsistent bounds → infeasible

- If Big-M value is too large → solve times become slower

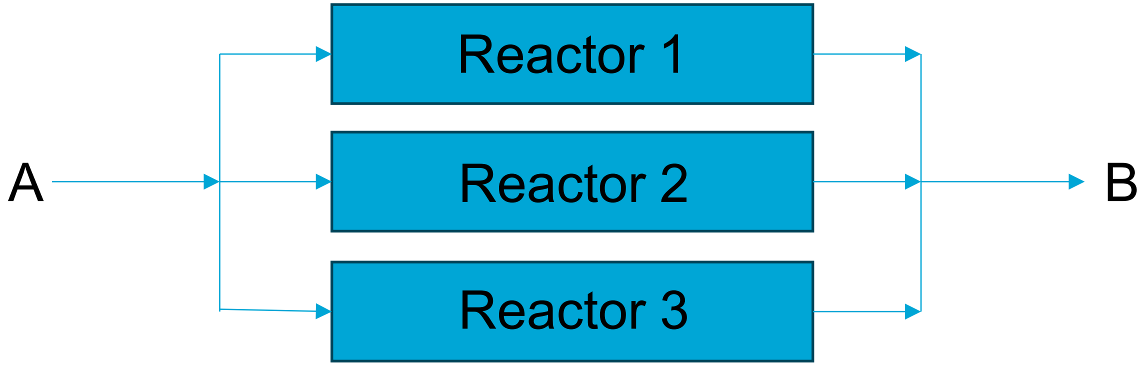
Outline

- **Anatomy of an optimization problem**
- **Linear Programming (LP)**
  - Fundamentals
  - Solving LPs – conceptual overview of simplex and interior point algorithms
  - Formulating linear programming models given an engineering problem statement
  - Analyzing solutions
- **Mixed-Integer Linear Programming (MILP)**
  - Fundamentals
  - **Formulating a MILP from an engineering problem statement**

Reactor selection problem

- Design a process with minimum cost to produce 10 kg/h of B from feedstock A (maximum availability = 15 kg/h). There are three reactors available each with different yields and costs. The details are:

	Yield	Fixed cost ($/h)	Variable cost($/kg)
Reactor 1	0.80	80	35
Reactor 2	0.667	54	30
Reactor 3	0.555	27	25



Formulating the objective function

- Objective function – minimize the cost of the plant by choosing appropriate reactors
- The reactor has a fixed cost and a variable cost
- Fixed cost should only be included if a reactor is selected:

$$80y_1 + 54y_2 + 27y_3$$

- Variable cost is dependent on the amount of feed ($\boldsymbol{x}$) to the reactor:

$$35x_1 + 30x_2 + 25x_3$$

- Thus, total cost representing the objective:

$$80y_1 + 35x_1 + 54y_2 + 30x_2 + 27y_3 + 25x_3$$

- Later we will ensure that $\boldsymbol{x} = 0$ if the reactor is not chosen

Modeling the material balances

- We consider the yield of each reactors when fed A to produce B

	Yield
Reactor 1	0.80
Reactor 2	0.667
Reactor 3	0.555

- Summing up over all reactors to get the total amount of B produced:

$$0.8x_1 + 0.667x_2 + 0.555x_3 = 10$$

- The coefficient on the right-hand side is 10 because we want to produce exactly 10 kg/h of B

Constraint on feed availability

- We have a constraint on maximum amount of feed available to us
- Over all three reactors this is expressed as

$$x_1 + x_2 + x_3 \leq 15$$

- All material flows are non-negative,

$$x_1, x_2, x_3 \geq 0$$

Feed flowrate should be zero if a reactor is not chosen

- We have the option to select among the reactors available.
- The feed to a reactor should only be non-zero only if the reactor is chosen
- We use the Big-M approach to model this:

$$x_i - 15\, y_i \leq 0, i \in \{1,2,3\}$$

- Here, M is 15 → maximum feed availability is 15
- When $y_i = 0$, we get $x_i \leq 0$. Recall we also have a constraint $x_i \geq 0$ to ensure non-negative flow of x_i → enforces $x_i = 0$
- When $y_i = 1$, we get $x_i - 15 \leq 0$, x_i can take any value between 0 and 15

Mixed-integer linear formulation for the problem

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{y}} \quad & 80y_1 + 35x_1 + 54y_2 + 30x_2 + 27y_3 + 25x_3 \\
\text{s.t.} \quad & 0.8x_1 + 0.667x_2 + 0.555x_3 = 10 \\
& \sum_{i=1}^{3} x_i \leq 15 \\
& x_i - 15y_i \leq 0, \ \forall\, i \in \{1,2,3\} \\
& \mathbf{x} \geq 0, \mathbf{y} \in \{0,1\}
\end{aligned}
$$

Live coding: Solving the MILP in SciPy

- Open Colab: Solving the formulated MILP using SciPy

- Find more in the Github repository of the course: https://github.com/process-intelligence-research/computational_practicum_lecture_coding/tree/main

Analyzing the solution obtained

- The optimal values are:

Variable	Value
Cost ($/h)	503.78
y_1	0
y_2	1
y_3	0
x_1	0
x_2	14.992
x_3	0

```
        message: Optimization terminated successfully. (HiGHS Status 7: Optimal)
        success: True
         status: 0
            fun: 503.7751124437781
              x: [ 0.000e+00  1.499e+01  0.000e+00 -0.000e+00  1.000e+00
                  -0.000e+00]
 mip_node_count: 1
 mip_dual_bound: 503.7751124437781
        mip_gap: 0.0
```

Do these results make sense to you?

Superstructure optimization – a brief discussion

- Modeling of such logical propositions widely carried out in Chemical Engineering
- Superstructure optimization is where we systematically chose among many alternatives during conceptual design of chemical processes[1]
- Superstructure optimization offers an alternative to hierarchical design of chemical processes
- Example: Optimal defossilization pathways for industrial cluster at the port of Rotterdam obtained by solving MILPs[2]

ELSEVIER

Chemical Engineering Science

Volume 321, Part A, 1 February 2026, 122783

CHEMICAL ENGINEERING SCIENCE

Superstructure-based optimization framework to assess defossilization pathways in petrochemical clusters ☆

Michael Tan [a], Igor Nikolic [b], Andrea Ramírez Ramírez [c]

Show more

+ Add to Mendeley   Share   Cite

https://doi.org/10.1016/j.ces.2025.122783 ↗   Get rights and content ↗

Under a Creative Commons license ↗   Open access

1. Mencarelli, L., Chen, Q., Pagot, A., & Grossmann, I. E. (2020). A review on superstructure optimization approaches in process system engineering. *Computers & Chemical Engineering*, *136*, 106808.
2. Tan, M., Nikolic, I., & Ramírez, A. R. (2025). Superstructure-based optimization framework to assess defossilization pathways in petrochemical clusters. *Chemical Engineering Science*, 122783.

What if I have larger problems?

- Real-world problems often contain millions of variables and constraints
- Current state-of-the-art MIP solvers are capable of efficiently solving such large problems using branch-and-bound methods
- SciPy is not particularly amenable to model such large problems
- Specialised optimization modelling environments exist

AMPL

GAMS

AIMMS

PYOMO

JUMP

Learning goals of this lecture

- After successfully completing this lecture, you are able to. . . .
  - formulate a linear and mixed-integer linear programming model from an engineering problem statement
  - analyze optimal solutions obtained on solving these models
  - explain at a high-level different algorithms used to solve linear programming models

Thank you very much for your attention!