Introduction to Mathematical Programming

Eren Darici M.S.
eren.darc@wmich.edu

WMU - IEE3110

January 30, 2026

Contents

Who Am I?

- B.S. Eskişehir Technical University (ESTU), Industrial Engineering (2023)
- M.S. Western Michigan University (WMU), Industrial Engineering (2025)
- Currently Ph.D. student at WMU
- Office hours: M 4–5pm, W 10–11am, or via email: `eren.darc@wmich.edu`

Today's Agenda

- Math modeling
- with Excel
- with Python
- *Extra: with GAMS*

Operations Research Modeling (Big Picture)

- **Define the decision:** what do we choose? (decision variables)
- **Objective:** minimize cost / maximize profit / maximize service level
- **Constraints:** limited time, budget, capacity, policy, etc.
- **Data & parameters:** any inputs that drive the decision
- **Solve & interpret:** optimal plan + sensitivity (what changes if inputs change?)

Source: (Stanke, n.d.)

Tools for Mathematical Programming

- **Excel Solver**
  - Great for small LP/MIP models and quick prototyping
  - Easy data entry + sensitivity reports (when available)
  - Limits: scalability, model structure, and reproducibility
- **Python ecosystem**
  - Modeling: `Pyomo`, `PuLP`, `gamspy`, `Google OR-Tools`
  - Pros: automation, reproducibility, data pipelines, visualization
- **Other modeling languages**
  - `GAMS`, `AMPL`, `JuMP` (Julia), etc.
  - Strong for large-scale optimization and industry workflows

Model vs. Language vs. Solver

Model (abstract)

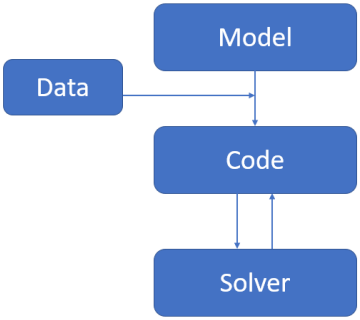
The *mathematical formulation*: decision variables, objective, constraints.

Modeling language / interface (implementation)

How we *write* the model so a computer can understand it (Excel, Pyomo, PuLP, GAMS, AMPL, ...).

Solver (engine)

The algorithmic software that *solves* the instance (e.g. Simplex) and returns an optimal/feasible solution.



Example Problem: Giapetto's Woodcarving, Inc.

Giapetto manufactures two wooden toys: **soldiers** and **trains**.

- **Selling price:** soldier \$27, train \$21
- **Raw materials:** soldier \$10, train \$9
- **Labor & overhead (variable):** soldier \$14, train \$10
- **Labor requirements:**
  - Soldier: 2 finishing hours, 1 carpentry hour
  - Train: 1 finishing hour, 1 carpentry hour
- **Weekly capacity:** 100 finishing hours, 80 carpentry hours
- **Demand:** trains unlimited, soldiers ≤ 40 per week

Goal: maximize weekly profit (revenue $-$ costs).

Mathematical Model (LP)

Decision variables: $x_1 =$ soldiers produced/week, $x_2 =$ trains produced/week.

Maximize

$$\max\ Z = 3x_1 + 2x_2$$

Subject to (s.t.)

$$\begin{aligned} 2x_1 + x_2 &\leq 100 \\ x_1 + x_2 &\leq 80 \\ x_1 &\leq 40 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Excel Follow-Along

Excel Follow Along: Link available

PuLP Follow-Along: Decision Variables

- We define **decision variables** for quantities we want to choose:
- Here: x_1 = soldiers/week, x_2 = trains/week

PuLP (Python) example:

```
import pulp
x1 = pulp.LpVariable("# of soldiers produced per week", lowBound=0)
x2 = pulp.LpVariable("# of trains produced per week", lowBound=0)
```

Tip: use `cat="Integer"` if you need integer variables.

PuLP Follow-Along: Objective

- Create a maximization problem
- Add the objective function: max $3x_1 + 2x_2$

```
prob = pulp.LpProblem("Giapetto", pulp.LpMaximize)
prob += 3*x1 + 2*x2
```

(In PuLP, `prob +=` is used to add the objective/constraints.)

PuLP Follow-Along: Constraints

- Add each constraint, one by one, with its meaning:

```
prob += 2*x1 + x2 <= 100  # finishing hours
prob += x1 + x2 <= 80     # carpentry hours
prob += x1 <= 40          # demand limit (soldiers)
```

- $2x_1 + x_2 \leq 100$ (finishing capacity)
- $x_1 + x_2 \leq 80$ (carpentry capacity)
- $x_1 \leq 40$ (max weekly soldier demand)

PuLP Follow-Along: Solve & Display Results

- Solve the model (uses a solver installed on your system)
- Print status, objective value, and decision variables

```
prob.solve()
print("Status:", pulp.LpStatus[prob.status])
print("Z =", pulp.value(prob.objective))
print("x1 =", pulp.value(x1))
print("x2 =", pulp.value(x2))
```

Tip: you can also loop: `for v in prob.variables(): print(v.name, v.value())`

Extra: GAMS (1/2) — Variables & Equations

```
Variables
    x1   soldiers produced per week
    x2   trains produced per week
    Z    objective value;

Positive Variables x1, x2;

Equations
    obj
    c1
    c2
    c3;
```

Extra: GAMS (2/2) — Model & Solve

```
obj..  Z =E= 3*x1 + 2*x2;

c1..   2*x1 + x2 =L= 100;
c2..   x1 + x2   =L= 80;
c3..   x1        =L= 40;

Model toy /all/;

Solve toy using LP maximizing Z;

Display x1.l, x2.l, Z.l;
```

Generative AI Usage (Guidelines)

- Generative AI is a **tool** — and tools are useless unless you know how to use them.
- You should know what you are doing; that is why we focus on the **math/modeling** part.
- I will show you how to use AI **for your benefit**.
- Please don't use it **to do OR / the math for you** (it can be confidently wrong).
- Example: OpenAI's ChatGPT
- You **can** use it for the **code section** (templates, syntax, debugging ideas).
- **Always check results** before you actually use them.
- Refer to the **syllabus** about **citation** and acceptable use.

References I

Stanke, B. (n.d.). *Operations research overview* [Blog post containing the OR lifecycle figure used in the slides]. Bob Stanke. Retrieved January 29, 2026, from https://www.bobstanke.com/blog/operations-research-overview