# Training Material - UPRC

4. Local Optimization

4.2 Task-Specific Optimization: (IT)

4.2.1 Robot movement optimization for individual tasks

Funded by
the European Union

# Educational Goals of the Training

**What will the user learn?**

1. Understand how robotic movement affects total energy usage in industrial environments
2. Learn how to model and minimize energy consumption using linear optimization techniques
3. Explore practical examples from industrial robotics applications.
4. Gain insights into the role of task scheduling, route selection, and movement control in energy efficiency
5. Analyze simulation results and decision-support tools for evaluating energy-optimized movement strategies
6. Build foundational knowledge that supports sustainable and cost-effective robotic system operation
7. The input and output structure for robotic movements

# Contents

# 1.1 Introduction to Optimizing Energy Consumption of Robotic Movement

**Why Optimize Robotic Energy Consumption?**

Pick-and-Place (PnP) tasks involve a robotic agent moving components from a storage location (e.g., bins or racks) to a target location (e.g., assembly points or kit holders).

**Industrial Context:**

- In modern factories and warehouses, robots are integral to performing repetitive pick-and-place, welding, painting, and sorting tasks
- Each movement — whether linear or rotational — activates motors and actuators that consume electrical energy
- As the number of operations increases, the cumulative energy demand becomes a significant cost factor in production

**Hidden Costs of Inefficient Motion:**

- Default robot paths often prioritize speed or shortest distance without considering energy-efficient acceleration, torque, or trajectory smoothness
- Frequent stop-start motion, sharp turns, or unnecessary path segments increase peak power demand and heat generation, leading to:
  - Shorter equipment lifespan
  - Higher cooling requirements
  - More frequent maintenance

# 1.1 Introduction to Optimizing Energy Consumption of Robotic Movement

**Why Energy Optimization Matters:**

- Cost Reduction: Even a 5–10% energy saving per robot can lead to thousands of euros annually in large-scale plants
- Battery-Operated Systems: In mobile robotics (e.g., AGVs), energy optimization directly extends mission time and reduces charging downtime
- Sustainability Targets: Lower energy use supports carbon reduction goals and regulatory compliance (e.g., ISO 50001, ESG mandates)
- Green Logistics: Eco-efficient robot routing is becoming essential in “green” supply chains and smart factories

**Real-World Insight:**

In one automotive assembly line, reprogramming robotic arms with energy-aware trajectories cut energy consumption by 12% with no loss in throughput.

**The Optimization Challenge:**

How can we redesign robot movement strategies to minimize energy usage, respect operational constraints, and maintain productivity?

# 1.2 Problem Description: Modeling Energy Consumption Industrial Concept

**Industrial Scenario:**

In the use case, roller hemming robotic arms are used for shaping and folding metal edges (e.g., doors, hoods) in automotive manufacturing.

**What Makes This Interesting:**

- Each robot motion is programmable using parameters like:
  - Speed
  - Acceleration
  - Jerk
  - Axis Positioning (A1–A6)
- These parameters affect both:
  - Time to complete the motion
  - Energy needed to execute it

**Multiple Ways to Do the Same Task:**

A single hemming trajectory can be executed:

- Faster with higher energy consumption
- Slower with smoother acceleration and reduced energy draw
- Balanced, using optimal parameter combinations

**Optimization Task in the Pilot:**

Find the best set of parameters (speed, acceleration, jerk) for each robotic task that results in the lowest energy consumption while still meeting production time requirements.

# 1.2 Problem Description: Modeling Energy Consumption Problem Modeling and Constraints

**Problem Scope:**

Traditional energy optimization focuses on path and velocity tuning. Our approach instead explores entire sets of operational scenarios using a Digital Twin (DT) to simulate robotic movements under varying control parameters:

- **Speed**
- **Acceleration**
- **Jerk**
- **Trajectory type**

Each movement is mapped to multiple configurations, with estimated time and energy consumption for each.

**Objective Function:**

Minimize total energy consumption across all selected scenarios.

$$min \sum_{i \in M} \sum_{j \in S_i} E_{ij} * x_{ij}$$

**Constraints:**

The first set of constraints ensures that for each robotic movement, one and only one scenario is selected. The second set of constraints guarantees that the combined duration of all chosen scenarios does not exceed the allowable production cycle time.

$$\sum_{j \in S_i} x_{ij} = 1, \forall i \in M$$

$$\sum_{i \in M} \sum_{j \in S_i} T_{ij} * x_{ij} \leq T$$

Finally, the variables are enforced to be binary.

$$x_{ij} \in \{0,1\}, \forall i \in M, \forall j \in S_i$$

# 1.2 Problem Description: Modeling Energy Consumption Problem Modeling and Constraints

**Preprocessing with Pareto Dominance:**

To reduce complexity:

- Only Pareto-optimal scenarios are kept.
- A scenario is dominated if another performs better or equal in all criteria and strictly better in at least one.
- Theorem 2.1 guarantees optimality is preserved with this filtering step.

This step drastically reduces solution time while maintaining solution quality

$$min \sum_{i \in M} \sum_{j \in S_i} E_{ij} * x_{ij}$$

$$\sum_{j \in S_i} x_{ij} = 1, \forall i \in M$$

$$\sum_{i \in M} \sum_{j \in S_i} T_{ij} * x_{ij} \leq T$$

$$x_{ij} \in \{0,1\}, \forall i \in M, \forall j \in S_i$$

**Theorem 2.1**

*Assuming the IP model (1) is feasible, there exists at least one optimal solution in which, for every robotic movement $i \in M$, the selected scenario $j \in S_i$ is Pareto optimal.*

# 1.3 Pareto Dominance
# Scenario Reduction with Pareto Dominance

**The Challenge:**

Each robotic movement can have dozens or even hundreds of possible configurations (speed, acceleration, trajectory, etc.). Modeling all of them leads to:

- High-dimensional decision space
- Increased computational time
- Possible intractability for large systems

**Why Scenario Filtering?**

We want to reduce the number of considered operational scenarios without losing quality in the optimization

This is achieved by applying Pareto Dominance:

- A concept from multi-objective optimization
- Helps us discard inefficient scenarios early

# 1.3 Pareto Dominance
# Scenario Reduction with Pareto Dominance

**What is Pareto Dominance?**

Given a set of performance metrics (e.g., energy and execution time):

A scenario $OS1$ dominates scenario $OS2$ if:

$fk(OS1) \leq fk(OS2)$ for all $k \in \{1,2,...,n\}$ (no worse in any metric)

$fk(OS1) < fk(OS2)$ for at least one $k$ (strictly better in at least one)

Only non-dominated scenarios are retained – these form the Pareto front

**Example:**

| Scenario | Energy (J) | Time (ms) | Dominated? |
|---|---|---|---|
| **A** | 40 | 160 | ❌ Pareto-optimal |
| **B** | 45 | 180 | ✅ (dominated by A) |
| **C** | 42 | 140 | ❌ Pareto-optimal |

**Scenario B** is dominated (worse in both metrics), so we remove it.

# 1.3 Pareto Dominance
# Scenario Reduction with Pareto Dominance

**Theorem 2.1 – Optimality Preservation**

*"If the full Integer Programming model is feasible, then there exists at least one optimal solution that includes only Pareto-optimal scenarios."*

This guarantees that removing dominated options does not exclude any part of the optimal solution space

**Benefits of Pareto Filtering:**

- Reduces number of decision variables
- Speeds up IP solver significantly
- Ensures energy-time trade-off is still fully respected
- Makes the optimization tractable for real-time use

# 1.4 Optimization Implementation

In modern manufacturing, robot movements are often programmed offline by experienced engineers using their intuition and domain expertise. However, this manual process may not fully exploit opportunities for energy-efficient execution.

To address this, we introduce an automated optimization service integrated directly into the robot design workflow. This service leverages a Digital Twin (DT) and Integer Programming (IP) to improve energy performance with minimal disruption to existing practices.

How Does It Work?

**Step 1: Program Extraction**

- The existing robot motion file is analyzed.
- Core elements like motion points, trajectory type, and axis data are extracted.

**Step 2: Scenario Generation**

- For each movement, multiple operational scenarios are generated using the DT.
- These scenarios vary across: Speed, Acceleration, Trajectory shape,Axis alignment, Motion type (Linear vs. Point-to-Point)

Linear Motion = accurate, path-constrained

Point-to-Point (PTP) = fast, free-form transition

# 1.4 Optimization Implementation

**Step 3: Simulation & Evaluation**

- Each scenario is simulated via the DT, providing:
  - Estimated energy consumption
  - Estimated execution time
- The result is a dataset of diverse, realistic, and feasible robotic movement options.

**Step 4: Pareto Filtering**

- Dominated scenarios are automatically excluded:
  - If a scenario is worse in both time and energy, it is removed.
- Only Pareto-optimal scenarios are retained for optimization.

**Step 5: Integer Programming Optimization**

- The filtered scenarios are passed to the IP model (see Slide: Modeling and Constraints).
- The model selects one scenario per movement, minimizing:
  - Total energy consumption
  - While staying within the allowed cycle time

# 1.4 Optimization Implementation

**Step 6: Reprogramming the Robot**

- The selected optimal scenarios are mapped back to robot control parameters.
- A new energy-aware robot program is generated.
- The result: Same task quality, lower energy, and no disruption to production.

**Summary: End-to-End Optimization Loop**

1. Read original robot program
2. Simulate alternative scenarios using the DT
3. Evaluate energy & time
4. Filter using Pareto dominance
5. Optimize using Integer Programming
6. Rebuild robot task with selected optimal movements

# 1.5 FMU Integration
# Generating Scenarios Using FMU

Goal: Integrate energy optimization into robot programming using Digital Twins

- Method: Generate alternative robot movements, simulate energy consumption
- Tool: Functional Mock-up Unit (FMU) for evaluating energy profiles of different scenarios

## From Robot Code to FMU Input - Key Steps

1. **Extract Input Parameters**:
   1. Robot specs: Max joint current, limits
   2. Axis-level inputs: Position, velocity, acceleration, jerk
2. **Transform Cartesian to Joint Space**:
   1. Inverse Kinematics converts motion into joint angles
   2. Multiple axis-level solutions possible
3. **Resolve Configuration IDs**:
   1. Use S (Status) and T (Turn) bits to ensure correct joint setup
   2. Removes ambiguity for FMU simulation

# 1.5 FMU Integration
# Generating Scenarios Using FMU

## Continuous Motion Handling

- CONT motions reinterpreted as one unified path
- Reduces number of discrete movements in IP model
- Increases number of scenarios per movement for optimization
- Parser and code generator updated accordingly

## Discretization for FMU Compatibility

- Split trajectory duration into equal time steps $\Delta t$
- At each step:
  - Compute position → Interpolate for velocity, acceleration, jerk
- Produces time-series data required by FMU
- Enables accurate energy simulation for every candidate scenario

# 1.5 FMU Integration
# Generating Scenarios Using FMU

## Enhancing Reconfigurability with Tool Modularity

- Modular tools → different TCPs (Tool Center Points)
- Affects:
  - Kinematics
  - Trajectory path
  - Time & energy profiles
- Digital Twin must simulate multiple tool settings
- Makes framework applicable across diverse robotic tasks
- FMU allows simulation of detailed joint dynamics
- Each robot movement is enriched into a set of executable energy-aware scenarios
- Reconfigurable tools increase applicability of optimization engine in flexible manufacturing
- Precise trajectory simulation improves real-world robot performance and energy efficiency

# 1.6 Input and Output Data Structure
# Code-Level View

## Input to the Optimization Process

- The robot program contains movement definitions for each step of a pick-and-place operation.

```
;*************************************************************
;*    Step1_90_plus_90deg
;*    Beschreibung
;*************************************************************

;===============================================
; Positions (if any)
;===============================================

DECL E6POS XGLOB_POS={X 1349.94,Y 1589.73,Z 1475.46,A -179.997,B -0.003,C 79.996,S 2,T 11}
DECL FDAT FGLOB_POS={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL PDAT PPGLOB_POS={VEL 100,ACC 100,APO_DIST 0,GEAR_JERK 100}
DECL E6POS XP10={X 1614.56,Y 1843.9,Z 1051.74,A -179.996,B -0.003,C 89.995,S 2,T 11}
DECL FDAT FP10={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLP10={VEL 3,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXAX
DECL E6POS XP5={X 1564.55,Y 1907.99,Z 971.86,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FP5={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLP5={VEL 0.3,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXA
DECL E6POS XLP1={X 1549.55,Y 1922.08,Z 976.99,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FLP1={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLLP1={VEL 0.3,ACC 100,APO_DIST 0,APO_FAC 0,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXAX_
DECL E6POS XLP2={X 1449.55,Y 1922.07,Z 977,A -179.996,B -0.003,C 89.996,S 2,T 11}
```

# 1.6 Input and Output Data Structure Code-Level View

Each motion includes:

- **Position coordinates (X, Y, Z, A, B, C)** – defines robot's pose.
- **Tool and base settings** – used for coordinate transformations.
- **Motion parameters**:
  - VEL – speed (velocity of the movement).
  - ACC – acceleration.
  - APO_DIST, APO_FAC – approximation values.
  - AXIS_VEL, AXIS_ACC – axis-level velocity and acceleration.
  - GEAR_JERK, JERK_FAC – jerk and gear smoothing values.

```
;*************************************************************
;*    Step1_90_plus_90deg
;*    Beschreibung
;*************************************************************

;===========================================
; Positions (if any)
;===========================================

DECL E6POS XGLOB_POS={X 1349.94,Y 1589.73,Z 1475.46,A -179.997,B -0.003,C 79.996,S 2,T 11}
DECL FDAT FGLOB_POS={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL PDAT PPGLOB_POS={VEL 100,ACC 100,APO_DIST 0,GEAR_JERK 100}
DECL E6POS XP10={X 1614.56,Y 1843.9,Z 1051.74,A -179.996,B -0.003,C 89.995,S 2,T 11}
DECL FDAT FP10={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLP10={VEL 3,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXAX
DECL E6POS XP5={X 1564.55,Y 1907.99,Z 971.86,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FP5={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLP5={VEL 0.3,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXA
DECL E6POS XLP1={X 1549.55,Y 1922.08,Z 976.99,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FLP1={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] " "}
DECL LDAT LLLP1={VEL 0.3,ACC 100,APO_DIST 0,APO_FAC 0,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXAX_
DECL E6POS XLP2={X 1449.55,Y 1922.07,Z 977,A -179.996,B -0.003,C 89.996,S 2,T 11}
```

# 1.6 Input and Output Data Structure
# Code-Level View

- **Output: Optimized Code**
- Output retains the structure and naming of the original robot motion file.
- The VEL values are updated based on the optimal scenario selected:

DECL LDAT LLP1={VEL 0.3,...} ⟶ DECL LDAT LLP1={VEL 0.1,...}

```
;*************************************************************
;*    Step1_90_plus_90deg
;*    Beschreibung
;*************************************************************

;===============================================
; Positions (if any)
;===============================================

DECL E6POS XGLOB_POS={X 1349.94,Y 1589.73,Z 1475.46,A -179.997,B -0.003,C 79.996,S 2,T 11}
DECL FDAT FGLOB_POS={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] ' '}
DECL PDAT PPGLOB_POS={VEL 57,ACC 100,APO_DIST 0,GEAR_JERK 100}
DECL E6POS XP10={X 1614.56,Y 1843.9,Z 1051.74,A -179.996,B -0.003,C 89.995,S 2,T 11}
DECL FDAT FP10={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] ' '}
DECL LDAT LLP10={VEL 0.4,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EX
DECL E6POS XP5={X 1564.55,Y 1907.99,Z 971.86,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FP5={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] ' '}
DECL LDAT LLP5={VEL 0.4,ACC 100,APO_DIST 10,APO_FAC 100,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXA
DECL E6POS XLP1={X 1549.55,Y 1922.08,Z 976.99,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FLP1={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] ' '}
DECL LDAT LLLP1={VEL 0.4,ACC 100,APO_DIST 0,APO_FAC 0,AXIS_VEL 100,AXIS_ACC 100,ORI_TYP #VAR,CIRC_TYP #BASE, JERK_FAC 50, GEAR_JERK 100, EXAX_
DECL E6POS XLP2={X 1449.55,Y 1922.07,Z 977,A -179.996,B -0.003,C 89.996,S 2,T 11}
DECL FDAT FLP2={TOOL_NO 16,BASE_NO 0,IPO_FRAME #BASE,POINT2[] ' '}
```

# 1.6 Input and Output Data Structure Code-Level View

**What the Optimization Alters**

- The optimizer **keeps the motion type and geometry** of the movements unchanged.
- The main optimization **variable is VEL (velocity)** per motion segment.
- The optimizer selects the best VEL values to **reduce energy consumption** while maintaining feasible operation timing.

**Why It Matters**

- Enables **energy-aware robot programming** without disrupting the base motion plan.
- Easy to integrate in existing industrial workflows:
  just replace original velocity values in .src or .dat robot files.

# Further Reading on the TSP

1. **Model Building in Mathematical Programming**
   *H. Paul Williams, 5th Edition (2013)*
   A practical text covering LP, MILP, and real-world modeling problems like production and logistics.
2. **Energy Efficiency in Industrial Robotics**
   *G. Hovland & M. Skogstad, Robotics and CIM (2019)*
   Discusses practical approaches and case studies for reducing energy in robot operations.
3. **Digital Twin-Driven Smart Manufacturing**
   *Tao et al., 2018*
   A foundational paper on how Digital Twins can model, simulate, and optimize industrial operations.
4. **Fundamentals of Optimization Techniques with Applications to Industrial Problems**
   *A. Ravindran et al., 2017*
   Includes integer programming and Pareto optimization applications.

# Q&A (1/2)

1. **Why is optimizing energy consumption important in robotic manufacturing?**
   A. It helps robots move faster
   B. It reduces the need for digital twins
   C. It minimizes operational costs and improves sustainability
   D. It ensures robots take shorter paths
2. **What does the optimization objective aim to minimize in the problem?**
   A. The total number of robot joints
   B. The robot’s battery weight
   C. The total energy consumption across movements
   D. The robot's end-effector torque
3. **What is the purpose of discretizing trajectories for FMU input?**
   A. To simplify robot programming
   B. To comply with FMU’s time-step simulation requirement
   C. To reduce the number of movements
   D. To increase CPU usage
4. **What is the main benefit of Pareto dominance filtering?**
   A. Makes robot paths smoother
   B. Ensures all robot tasks are executed faster
   C. Reduces the number of suboptimal scenarios
   D. Randomizes velocity selection

# Q&A: TSP Understanding (2/2)

- Answers:
  - 1. C
  - 2. C
  - 3. B
  - 4. C

# CONSORTIUM

## meet the team

MOSAPTO

ATC
ATHENS TECHNOLOGY CENTER

Fraunhofer
IOSB

UNIVERSITÉ
DE LORRAINE

UNIVERSITY OF PIRAEUS
RESEARCH CENTER

THE E-BUSINESS RESEARCH CENTER
ELTRUN
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

EKS
InTec GmbH

BOC
www.boc-group.com

AEGIS
IT RESEARCH

LIC
LIETUVOS
INOVACIJŲ
CENTRAS

FFT

SEW
USOCOME

CRF

ILTAR
ITALBOX

Funded by
the European Union

# CONTACT us

MODAPTO


modapto.eu


info@modapto.eu


MODAPTO Horizon Europe Project

@MODAPTO_eu

MODAPTO Horizon Europe Project

Funded by
the European Union