# Turning Torso Optimization Project

This project explores the geometric, visual, and solar optimization of a twisting tower based on the Turning Torso concept. It integrates parametric modeling, data-driven analysis, and machine learning techniques.

---

**Project Structure**
This README file is provided to assist the user in understanding and working with the files included in this project package:

1. **Rhino Turning Torso.3dm**

This is the Rhino file created to provide an environment for running the corresponding Grasshopper file described below.
Important Note: The Rhino file itself appears empty except for the city context because all the main building modeling has been carried out entirely within the Grasshopper environment.

2. **Grasshopper Turning Torso.gh**

This is the main modeling file of the building, developed in Grasshopper. It contains three grouped sections to help users understand the step-by-step modeling process:
- Base Surface Modeling
- Climate and Sky Modeling
- Rotation and Irradiation Analysis

Each group is clearly labeled for easier navigation and understanding of the modeling procedure.

3. **dataset.xlsx**

This Excel file includes the original irradiation result values obtained from the Grasshopper model, along with their corresponding edge lengths and rotation angles of the building.
The dataset contains 92 entries, which are used as the input for the emulator and optimization tasks.

4. **full dataset.xlsx**

This Excel file contains the complete input and output data for all tested design alternatives. Each row represents a unique combination of number of floors, L1 (side length), city, and rotation angle, along with the corresponding irradiation and surface area results

5. **Code file Turning Torso (Visualization, Emulator, Optimization).**ipynb

This Jupyter Notebook file contains the code developed for:
- Visualizing and analyzing the dataset
- Creating an emulator to approximate the surface behavior
- Performing the optimization task based on the emulator results

The code is written in Python and must be run in a Jupyter Notebook environment.
Important: The dataset.xlsx file must be placed in the same directory as the Notebook file for proper execution.

6. **Report Turning Torso.pdf**

This is the final report of the project, documenting all steps of the work — starting from the initial modeling in Grasshopper through to the final optimization and conclusions.

7. **Presentation Turning Torso.pdf (.pptx)**

This file contains the presentation slides that summarize the key milestones, processes, and results of the project. It visually highlights the main steps from geometry generation to emulator development and optimization, offering a concise overview for quick understanding.

8. **Improvements Turning Torso.pdf**

This document summarizes all the modifications made since the last submission. It provides a clear overview of updates in the modeling, code, interface, or documentation, allowing reviewers or team members to quickly catch up on what has been improved or added

9. **Website Interface**

To extend the usability and creativity of the project, a web interface was developed. This website allows users to:

- Select a city
- Choose a building shape (square, pentagon, hexagon)
- Enter values for L1, rotation angle, and number of floors

Upon submission, the system processes the input using interpolation and outputs the estimated irradiation. The result is dynamically shown and saved.
All necessary files—including the Python script (app.py), HTML templates (index.html, results.html), and the Excel dataset—must be located in the same folder

- How to launch the website:
  1) Open your terminal.
  2) Navigate to the folder containing app.py.
  3) Run: python3 app.py
  4) A local URL will be generated (e.g., http://127.0.0.1:5000), which you can open in your browser to use the tool.

Important Notice
The entire project has also been uploaded to GitHub at the following link:
GitHub Link
If there are any issues opening the files or running the code, you can access the original files directly from GitHub.

---

## Key Features

- **Parametric Modeling**: The tower geometry is parametrized using Grasshopper based on the side length `L1` and rotation angle per floor.
- **Data Analysis & Emulator**: A neural network emulator is trained on the dataset to estimate solar radiation output for various input configurations.
- **Optimization**: The model identifies the input configuration (L1 and rotation) that yields as close as possible to the determined value.

---

## How to Use

1. **Requirements**:
   - Python 3.8+
   - Required packages: `pandas, numpy, matplotlib, openpyxl, scikit-learn, tensorflow`

2. **Run the Notebook**:
   - Open the notebook in Jupyter or VS Code.
   - Ensure `dataset.xlsx` is in the same directory.
   - Run all cells in order to train the emulator and perform optimization

3. **Model Files**:
   - Open `Grasshopper Turning torso.gh` with Grasshopper (within Rhino).
   - Use the provided Rhino file to explore or render the final model.

# Emulator-Based Python Workflow

This project includes a Python-based emulator that supports the Grasshopper simulation by analyzing, fitting, and optimizing radiation data. It consists of three main stages:

- **Step 1: Visualization of Original Data**

The first step involves exploring the structure and trends in the dataset through:

- 2D Plots: Visual inspection of how solar irradiation varies with each feature (e.g., L1, rotation, NoFloor) independently.
- 3D Surface Plots: A deeper look at the interaction between two input parameters (e.g., L1 and rotation) and their impact on irradiation.

These plots help to intuitively understand which variables most strongly influence the irradiation and how they behave in combination.

---

- **Step 2: Emulator – Polynomial Regression Model**

To predict irradiation for intermediate or unseen values, a polynomial emulator is constructed:

- Cross-validation is applied to evaluate different polynomial degrees and prevent overfitting.
- Polynomial degrees are tested (e.g., 2 to 6), and the model with the lowest validation error is selected.
- The final emulator captures the nonlinear relationship between inputs (like rotation and L1) and the resulting irradiation.

This step mimics the Grasshopper simulation in a much faster, analytical form.

---

- **Step 3: Optimization – Finding the Best Rotation and L1**

Using the polynomial emulator from Step 2:

- The goal is to find input values that yield irradiation as close as possible to 6000 kWh.
- A cost function is defined:
  Cost = (Predicted Irradiation - 6000)$^2$
- The minimize() function from scipy.optimize is used to solve this optimization problem.
- The result gives optimal values of rotation and L1 to target the 6000 kWh irradiation goal.

# Grasshopper Components and Python Scripts

The Grasshopper definition used in this project consists of several grouped sections. Each of these parts

- **Section 1: Automated Input Iteration**

(Combination Loop – Component: Loop Controller)
This section uses Python to automatically iterate through combinations of L1, the number of floors, and rotation angles. It creates all combinations using the tools library and updates the model one by one by triggering the loop. This allows automated simulation of various geometry configurations for data collection.

- **Section 2: Modeling Based on L1 and L2 –** Finding the Midpoint (Midpoint Calculation – Component: Midpoint Calculator)

The Python code is designed to compute the top vertex of the pentagon by vertically offsetting the midpoint of the bottom horizontal edge (L1). This is based on trigonometric relationships, considering that the bottom three sides of the pentagon have equal lengths (L1), and the two base angles are fixed at 32° and 58°. The computation ensures that the overall perimeter of the pentagon remains constant at 120 meters.

- **Section 3: Total and Each Floor Rotation, Model Volume**

These components calculate the total rotation of the tower and the rotation angle per floor. The model volume is also modeled using the final solid geometry to provide a more complete picture of the overall design.

- **Section 4: Sky Dome, Sun Path, Radiation and Sun Status**

This section visualizes the sun path, irradiation and radiation distribution, and solar vectors using the imported EPW weather datas. It allows users to simulate environmental conditions at different times and dates.

- **Section 5: Calculating Surface Area**

This section calculates the total exposed surface area of the tower by evaluating the mesh faces and summing their surface areas. This is necessary to calculate radiation per square meter.

- **Section 6: Total Incident Irradiation on the Surface**

This component computes the total irradiation received on the geometry using the radiation analysis tools. The result is derived from the geometry's interaction with the generated sky matrix and sun vectors.

- **Section 7: Note on Data Export (Component: Export CSV)**

Although not a core modeling step, this Python script gathers simulation results (L1, number of floors, rotation, city, irradiation, and area) and saves them into a `.csv` file. The path is set to the Grasshopper file location.