JROT – Job Rotation Optimization Tool

Source Code available at: https://github.com/mhenriquezschott/ErgoTools/tree/jrot

JROT zip file at: https://github.com/mhenriquezschott/ErgoTools/archive/refs/heads/jrot.zip

Main ErgoTools Interface

This interface allows users to input, visualize, and save individual ergonomic risk assessments using **LiFFT**, **DUET**, and **The Shoulder Tool (TST)**.

- Estimates can be saved for individual jobs.
- Jobs are saved via the **top right corner** (see **Figure 1**). Users must enter a **Job ID** and may add a **name** and **description**.
- Risk scores for each tool can be saved directly.

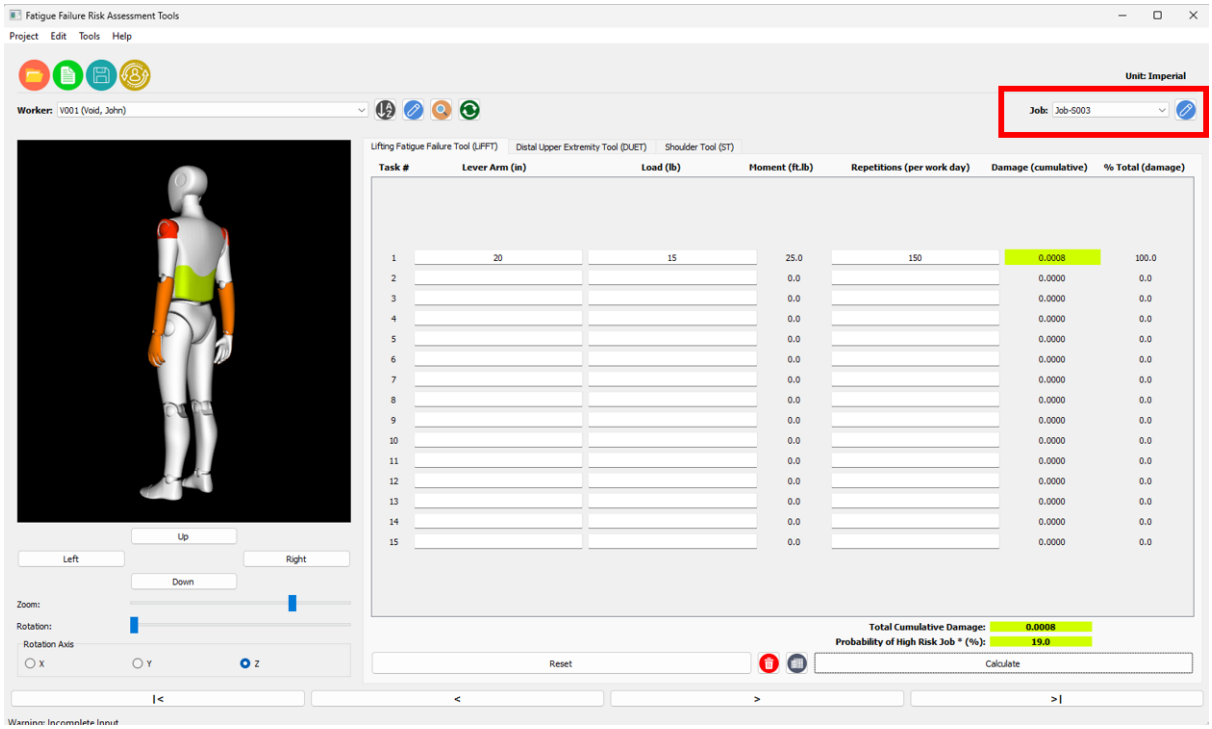

Figure 1. *Main ErgoTools interface with job-saving features highlighted (top right corner).*

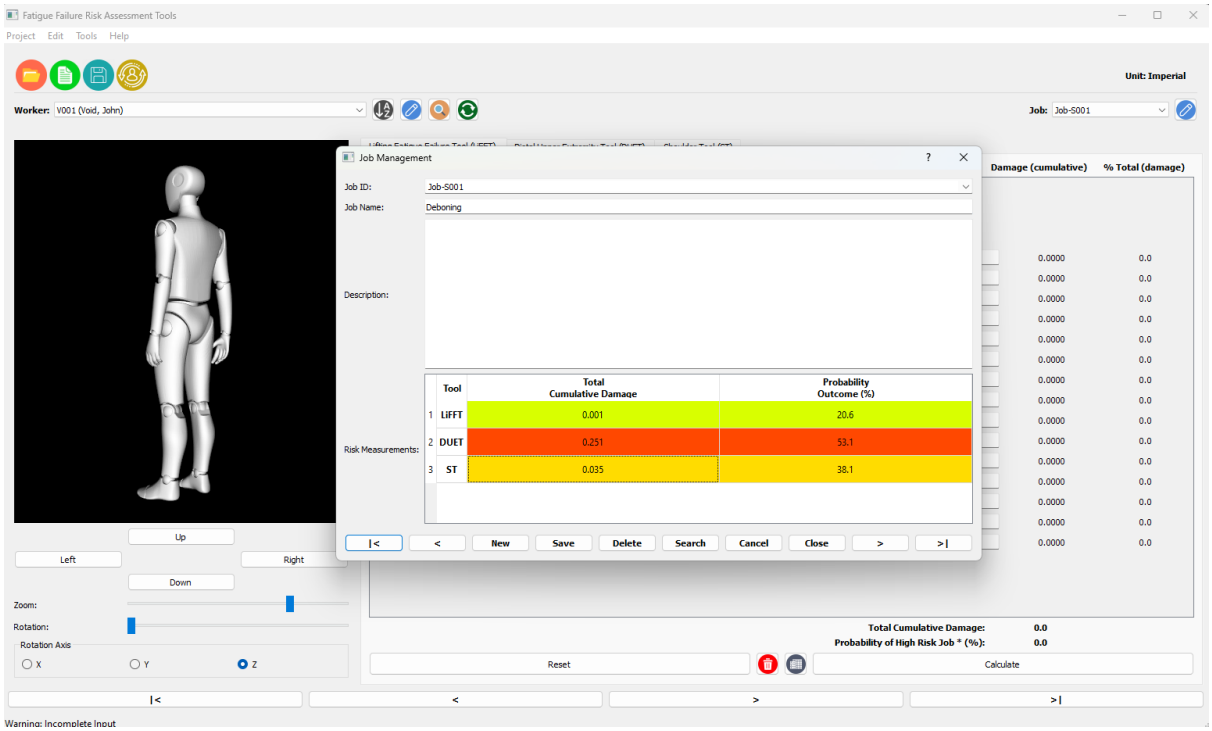

Figure 2. *Job Management interface.*

Data Management

The **top left corner** of the interface provides file management options (see **Figure 3**):

- Create new projects
- Open and save existing projects
- Export data

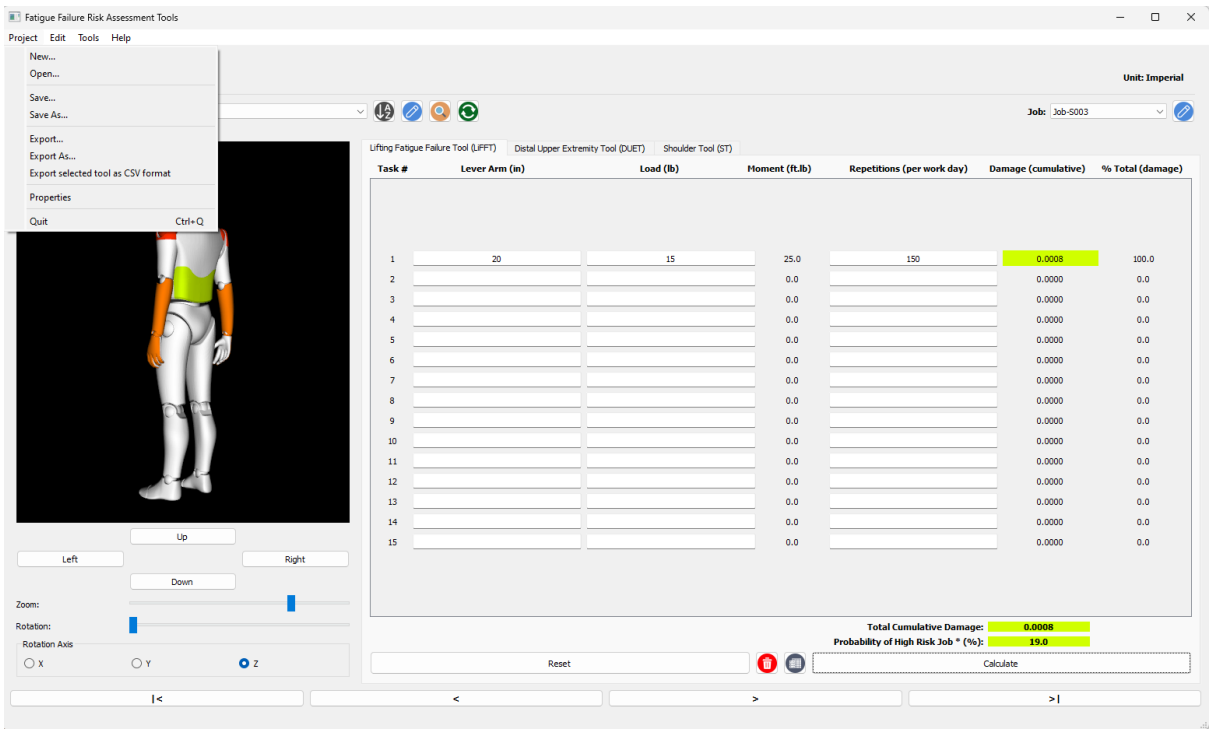

Figure 3. *Top left menu for managing files and projects.*

JROT Blank Interface

This interface is used to define workers and assign jobs into a rotation matrix:

- The matrix size depends on the number of **workers** and **time blocks**, which are configured at the top (see **Figure 4**).
- Previously saved projects with risk data can be loaded and reused (Rotation menu).

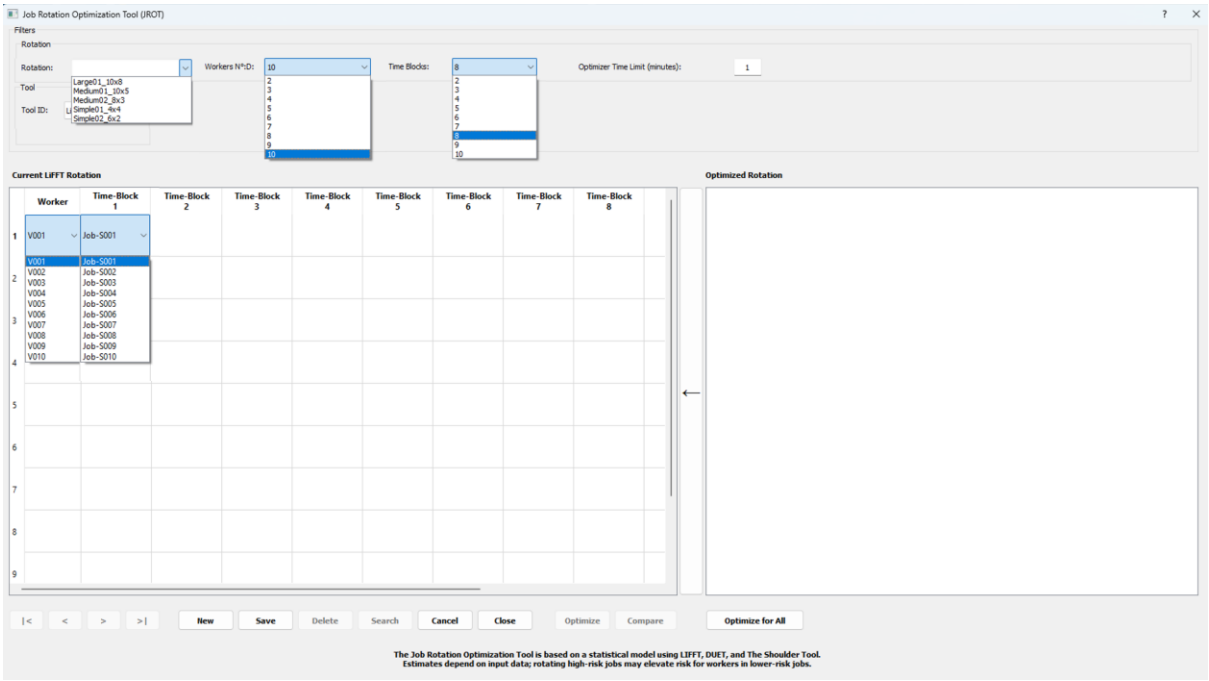

Figure 4. *Blank JROT interface with configurable workers and time blocks.*

JROT Example Interface

In this view, users can assign jobs to workers over time:

- Each job can be added **only once per time block** (see **Figure 5**).
- This ensures that no job is repeated during the same time interval.

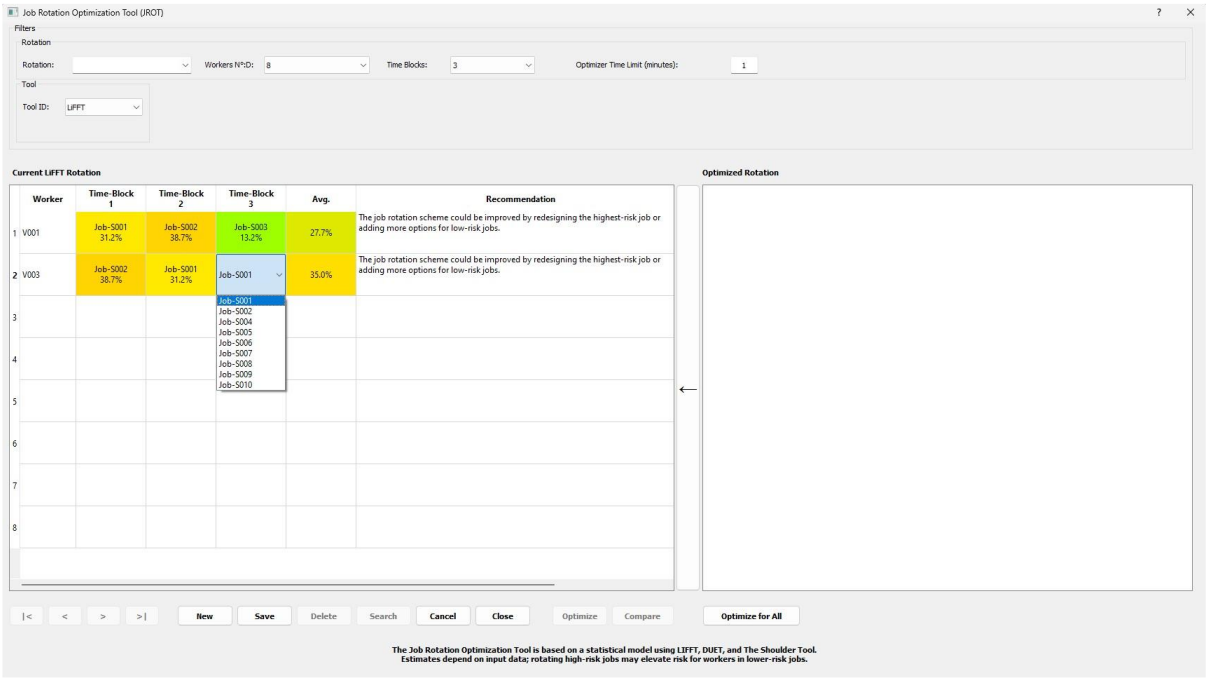

Figure 5. *Example job rotation matrix where jobs are assigned to workers.*

Single-Tool Optimization

Optimize job rotation based on **one tool** (LiFFT, DUET, or TST):

- Ensure tool-specific risk data is saved.
- Select the tool and click **“Optimize”** (see **Figure 6**).
- A loading bar will indicate progress.

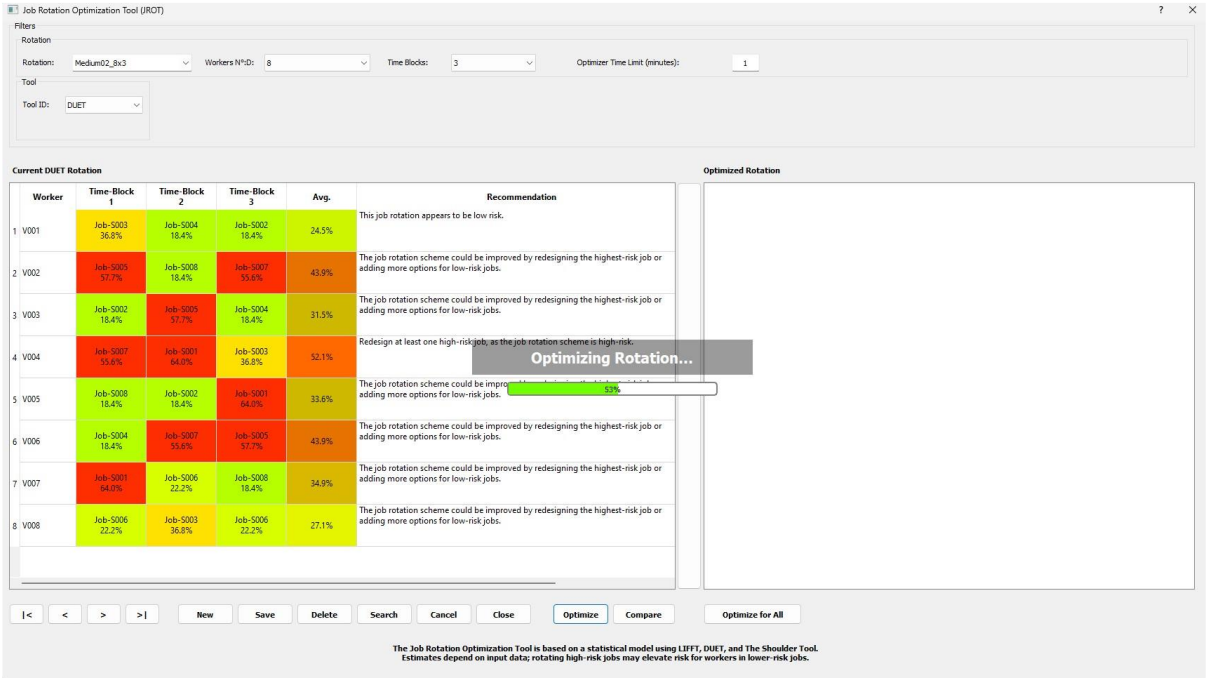

Figure 6. *Single-tool optimizer interface with selected tool and optimize button.*

This optimization uses **MILP**, implemented in Python via **PuLP**, solved using **CBC**. It's fast and ideal for tool-specific fine-tuning.

- After completion, results appear on the **right side** (see **Figure 7**).
- Click the **left arrow button** to overwrite the original with the optimized rotation.

Figure 7. *Post-optimization view showing the optimized solution and apply button.*

Multi-Tool Optimization

This mode performs **multi-objective optimization** using all three tools:

- It aims to reduce both the **maximum average risk per tool** and the **spread between workers**.
- Implemented using **Pyomo** and solved with **GLPK**, this method is more computationally demanding.

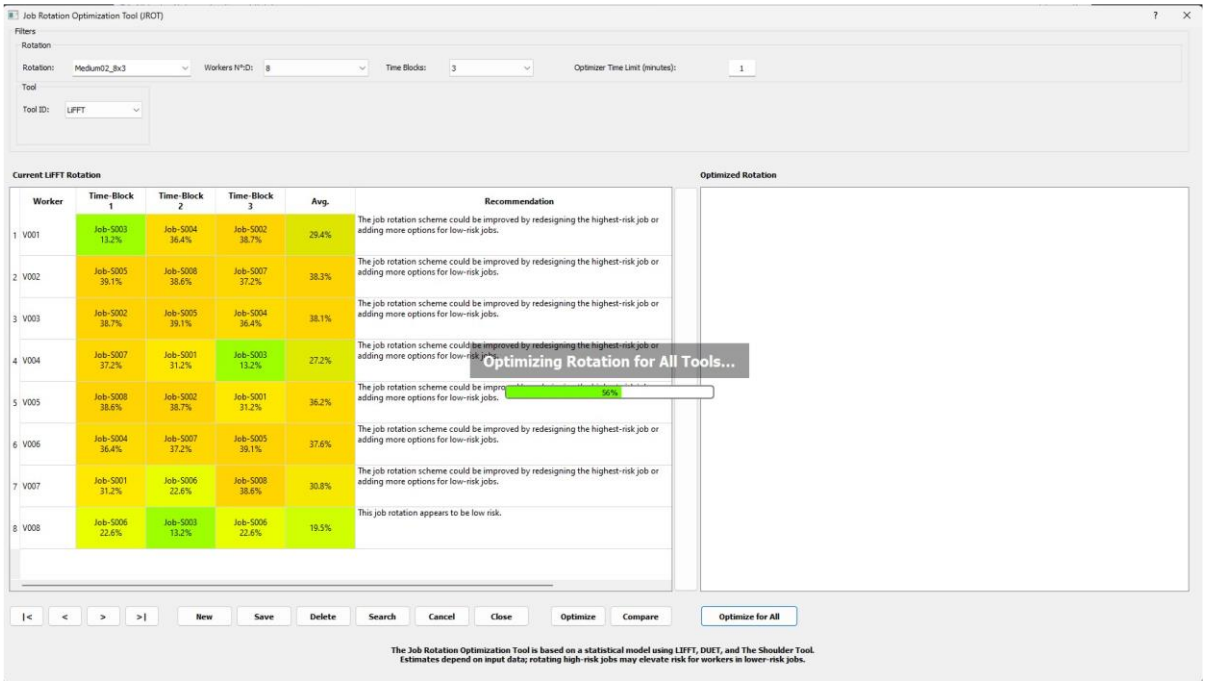

Figure 8. *Multi-tool optimizer interface with progress and results display.*

- Once complete, a window opens to compare results and apply the optimized schedule.

Figure 9. *Comparison screen showing multi-tool optimization outcomes.*