# Pure-Load Tensor (PLT) + Impulse–Response (IR)

Endurance training data are usually collapsed into **single scalars** (e.g., TSS). That destroys the **sequence**, the **physiology** (HRV/sleep/resting HR), and the **structure within sessions**. This repo implements the **Pure-Load Tensor (PLT)** idea and a minimal **impulse–response (IR)** layer to turn daily training into a sequence-aware, physiology-aware signal you can visualize and model.

> **Paper (short)** — We introduce a *Pure-Load Tensor* that combines: **TSS**, **specific energy** (kJ/kg), **heart-rate quartiles** across the session (Q1..Q4), **pre/post-session HRV (rMSSD)**, and an **HRV return-to-baseline rate** ($\lambda_{\mathrm{HRV}}$). An order-aware **carry-over** operator handles **two stimuli on the same day**. Coupled with a simple **impulse–response** layer, PLT outperforms TSS-only baselines on short-term recovery events and 28-day performance deltas.

---

## Contents



---

## Concept

**Why tensors?** Vectors and matrices are special cases of tensors. PLT turns each day into a **multi-feature representation** that keeps:

- *External load*: TSS and specific energy (kJ/kg),
- *Intra-session shape*: HR quartiles (Q1..Q4),
- *Physiological state*: HRV before/after the session and the rate at which HRV returns to baseline,
- *Order*: if there are two sessions in a day, the second **inherits** the effect of the first via an exponential relaxation of HRV.

Then a single scalar **impulse** is computed per day $u_d = \mathrm{softplus}(\beta^\top \tilde{\mathbf{v}}_d^{\mathrm{PLT}})$ (where $\tilde{\mathbf{v}}$ is robustly scaled per athlete) and fed to a **Banister-style IR** filter to obtain a **chronic load** $F_d$ (fitness) comparable in spirit to **CTL**, but physiology-aware.

> This repo ships **Part I (PLT)**. A future **FMT** variant can preserve full intra-session waveforms, but is not needed for the minimal pipeline.

## Repository structure

```
plt_tensor_ir_model.py   # End-to-end PLT → u_d → IR; also computes CTL
```

## Data schema

Pass a CSV with **one row per session** (one per day also works). Case-insensitive column names.

**Required (or strongly recommended):**

- `date` (YYYY-MM-DD or timestamp)
- `tss` (Training Stress Score; any consistent scalar load)
- `hr_q1, hr_q2, hr_q3, hr_q4` (mean HR in time quartiles)
- `hrv_pre_ms` (rMSSD before session, or morning rMSSD)
- `hrv_post_ms` (first post-session rMSSD, \~10–20 min after)
- `hrv_post2_ms` *(optional)* and `dt01_h`, `dt12_h` *(lags in hours)* → used to estimate $\lambda_{\mathrm{HRV}}$
- `kj` (mechanical work, kJ) and `mass_kg` → to derive specific energy `Ekg = kj / mass_kg`
- `duration_s` *(optional)* → used as aggregation weight if there are 2 sessions per day
- `start_time` *(optional)* → to order same-day sessions
- `athlete_id` *(optional but recommended)* → enables per-athlete robust scaling

> Don't have some fields yet? The pipeline still runs: it uses sensible defaults and will switch to a velocity proxy for $\lambda_{\mathrm{HRV}}$ when only one post-HRV exists.

**Minimal daily-only example** (single session per day):

```
date,tss,kj,mass_kg,hr_q1,hr_q2,hr_q3,hr_q4,hrv_pre_ms,hrv_post_ms
2025-01-01,75,650,70,122,126,130,135,58,46
2025-01-02,0,0,70,105,106,107,108,60,58
```

## Quick start

```
# 1) Create/activate a virtual env (recommended)
python -m venv .venv && source .venv/bin/activate

# 2) Install dependencies
pip install numpy pandas matplotlib

# 3) Run the pipeline
python plt_tensor_ir_model.py data_sessions.csv --save-daily daily_plt_ir.csv
```

This will:

1. Build **daily PLT vectors** (per-athlete robust scaling included),
2. Compute the **daily impulse** `u_plt`,
3. Run an **IR fitness–fatigue filter** to get `F_tensor` (chronic load), `G_tensor` (fatigue), `P_hat` (latent performance),
4. Compute **CTL** from TSS for comparison,
5. Save everything to `daily_plt_ir.csv`.

---

## Command-line usage

```
python plt_tensor_ir_model.py path/to/sessions.csv \
  --target-col mmp5            # optional: fit IR params to a target series
  --athlete-col athlete_id     # optional: per-athlete scaling
  --save-daily daily_plt_ir.csv # output file (default shown)
```

### Notes

- If `--target-col` is provided and exists in the data (e.g., `mmp5`, `FTP`, or periodic CP tests), a coarse grid search estimates $\tau_f, \tau_g, k_f, k_g, P_0$ . Otherwise defaults are used ($\tau_f = 42$ , $\tau_g = 7$ , $k_f = 1$ , $k_g = 2$ ).
- When there are **two sessions/day**, the second inherits pre-HRV via an exponential return model (order-aware).

---

## Programmatic usage

```python
from plt_tensor_ir_model import run_pipeline

res = run_pipeline("data_sessions.csv", target_col=None,
save_daily_csv="daily_plt_ir.csv")

daily = res["daily"]      # dataframe with PLT features and u_plt
F = res["F"]               # tensorial chronic load (fitness)
CTL = res["CTL"]           # PMC CTL from TSS (if TSS available)
```

---

## Outputs and interpretation

`daily_plt_ir.csv` adds the following columns:

- `u_plt` — daily **impulse** derived from PLT (softplus of a weighted projection).
- `F_tensor` — **chronic load** (IR fitness) from `u_plt` (fading-memory filter).
- `G_tensor` — **fatigue** (fast branch).
- `P_hat` — latent performance from the IR combination (optional to plot).

- `CTL_TSS` — **CTL** computed from TSS (EWMA with $\tau = 42$ ).

**How to compare**

- CTL (left axis) vs **F_tensor** (right axis) → two curves, two units. Look at **shape and turning points**, not numeric equality. Peaks in `F_tensor` without big CTL moves imply **hard structure** (intervals, high decoupling, HRV hit) at similar TSS.

---

## Plots

A minimal dual-axis plot in Python:

```
import pandas as pd, matplotlib.pyplot as plt

df = pd.read_csv("daily_plt_ir.csv", parse_dates=["date"]) if "date" in
df.columns else pd.read_csv("daily_plt_ir.csv")
fig, ax1 = plt.subplots(figsize=(12,4))
ax1.plot(df["date"], df["CTL_TSS"], label="CTL (TSS)")
ax1.set_ylabel("CTL (TSS)")
ax2 = ax1.twinx()
ax2.plot(df["date"], df["F_tensor"], "--", label="Tensor chronic load F")
ax2.set_ylabel("Tensor F")
ax1.set_xlabel("Day")
ax2.legend(loc="upper left")
fig.tight_layout(); plt.show()
```

---

## Tuning / fitting

- **Impulse weights (β)**: by default we emphasize TSS, Ekg, decoupling (HR_Q4 − HR_Q1), HRV drop (pre−post), and $\lambda_{\mathrm{HRV}}$ . You can replace the default β vector with athlete-specific weights (e.g., via regression to your targets).
- **IR parameters**: use `--target-col` to fit $\tau_f, \tau_g, k_f, k_g, P_0$ against performance markers (e.g., `mmp5`, `FTP`).
- **Two sessions/day**: provide `start_time` / `gap_h_to_prev` to activate order-aware carry-over of pre-HRV for the second session.

---

## FAQ

**Q: Can I run this with daily rows only?** Yes. Use one row per day; the pipeline still computes PLT, `u_plt`, IR, and CTL.

**Q: I don't have** `` `**.**  The code switches to a **velocity proxy** for $\lambda_{\mathrm{HRV}}$ ``

`using` (H1−H0)/Δt`.

**Q: Units differ between CTL and Tensor F — is that OK?** Yes. Use **two y-axes**. Compare *shape*, *lags*, and *turning points*.

**Q: Where do HR quartiles come from?** Split each session **by time** into four equal parts; take the mean HR in each part (Q1..Q4).

---

## Citations


- Banister et al. (1975/76): *A systems model of training for athletic performance*.
- Allen & Coggan (2019): *Training and Racing with a Power Meter*.
- Task Force (1996): *Heart rate variability standards*.
- Skiba et al. (2012): *W′ balance dynamics*.


---

## License