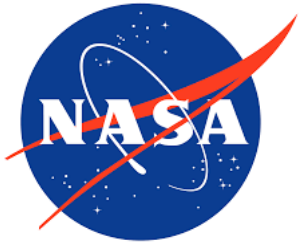


ASME TURBO EXPO 2026
· GT2026-175147

Turbo-Design: Open-Source Radial Equilibrium Turbomachinery Solver

Part I — Turbines. A Python framework for radial equilibrium analysis with extensible loss models.

Dr. Paht Juangphanich, Dr. Arman Mirhashemi (NASA Glenn) · William Andress (Purdue) · June 2026

github.com/nasa/turbo-design

Motivation

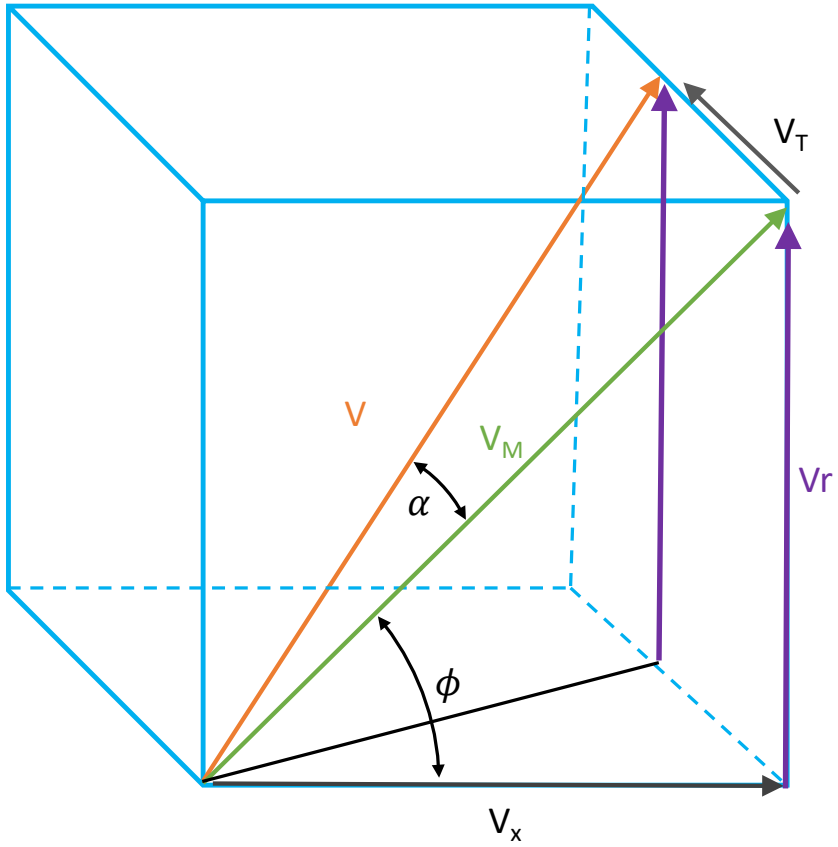
Legacy radial-equilibrium codes are poorly documented; commercial codes are closed-source; neither was built for machine-learning loss models.

NASA's legacy radial-equilibrium tools — TD2, AXOD, and RTD — were written in Fortran 77 in the 1970s–90s. Their assumptions are poorly documented and difficult to extend. Commercial alternatives like AxSTREAM and AXIAL are proprietary, which makes it hard for academia and government to add custom physics or plug in machine-learning inference. Modern designs reduce losses through 3D optimization and advanced cooling, but legacy cascade-based loss models can't capture those gains.

- Open source, written in Python — easy to verify, modify, and extend
- Modular loss-model interface — drop in custom correlations or trained ML inference models
- Handles both axial and radial machines from a unified formulation

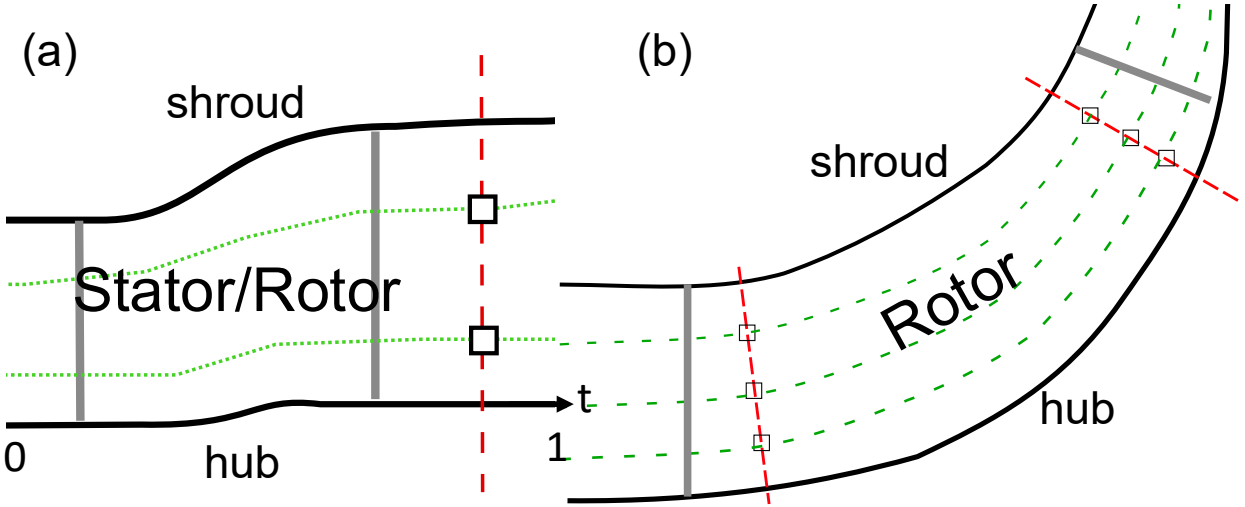
Outline

Coordinate system and geometry



Built for radial flow paths

- Angles are defined relative to the meridional velocity, not the axial.
- Hub and shroud are defined as smooth curves rather than the jagged blade-row endpoints of legacy codes, giving accurate radius-of-curvature and inclination angles that drive the radial-equilibrium balance.

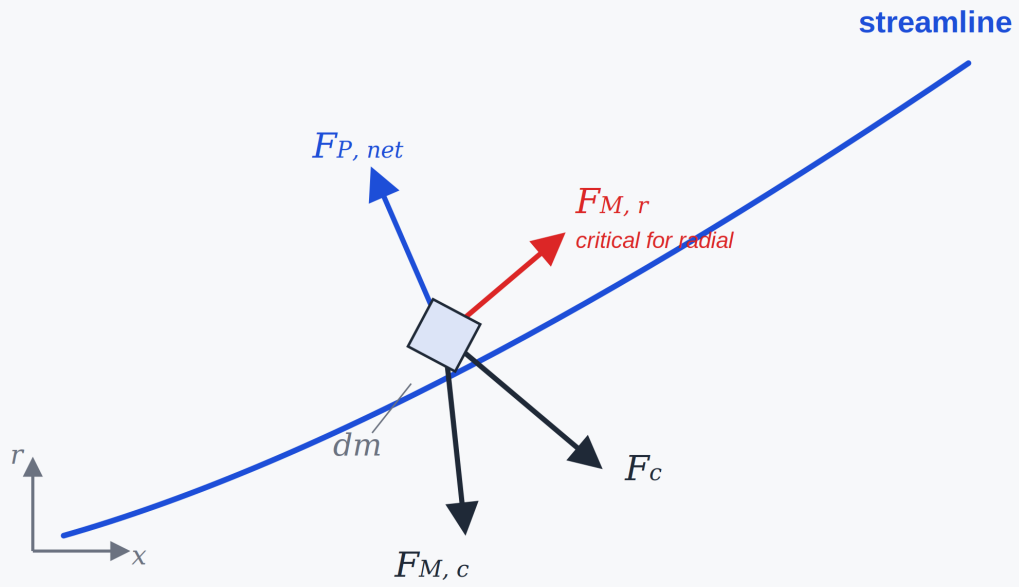


Velocity decomposition. The angle α is measured from V to the meridional V_m.

Radial equilibrium

Four-force balance

on a fluid element along a curved streamline



$$\sum \vec{F} = \vec{F}_{P,net} = \left(F_c + F_{M,c} + F_{M,r}\right)\hat{r} \qquad (5)$$

⋮ Full derivation in Appendix A

$$\hat{r}: \quad \frac{1}{\rho}\frac{dP}{dr} = \frac{V_T^2}{r} - \frac{V_M^2}{r_M}cos(\phi) - V_r\frac{dV_M}{dM} \qquad (15)$$

OTAC Manual [10] and Cumpsty [11]

Derivation

Most 1D analyses drop the last term. For axial machines that is fine — V_r is small. For radial machines, where hub and shroud radii change significantly, dropping it introduces error in turning angles and streamline positions. Turbo-Design retains the term, computed as dM = (dx, dr) along the streamline.

Four-force balance on a fluid element. The radial pressure gradient must keep the element on its streamline.

Solver modes

Pressure Balance mode

- Blade angles are known. The solver balances pressure across all blade rows simultaneously using SciPy SLSQP, expressing pressures as fractional drops bounded between 0 and 1. This forces monotonic pressure decrease and improves convergence.
- Use for: design point analysis and off-design analysis when geometry is fixed.

Angle Matching mode

- Blade angles are unknown — the solver determines them
- Target mass-flow distribution from hub to shroud sets the radial loading
- Adjusting that distribution controls blade twist directly
- Use for: design exploration before geometry is finalized

Derivation

Stability: the kinetic-energy parameter C (Appendix A) must remain below unity. As C approaches 1 the passage chokes; the solver detects this condition and reports it rather than returning NaN.

Loss models as a modular interface

Legacy correlations

Four classical loss models are digitized from their original figures and fit as B-spline surfaces:

- Ainley-Mathieson (1951)
- Kacker-Okapuu (1982)
- Craig-Cox (1970)
- Traupel (1966)

CSV datasets are open in the repository so anyone can refit or audit them. Craig-Cox handles incidence — the others can be extended the same way.

Framework for ML-based loss

- Common base class receives upstream + current row properties
- Any inference model (PyTorch, ONNX, scikit-learn) plugs in by inheriting it
- Designed for interpolation within the training space — extrapolation needs care
- Trained models on real engine data are deferred to a follow-up paper

Legacy correlations and machine-learning models share the same base-class interface; substituting one for the other requires no changes to the solver.

Entropy-based efficiency for radial turbines

The standard total-to-total formula misleads on radial machines — entropy gives a frame-independent answer.

For radial turbines, absolute total-pressure ratio is dominated by centrifugal frame-change rather than aerodynamic loss, so the standard formula reports near-unity efficiency even with significant irreversibilities. Recasting through entropy generation isolates real losses:

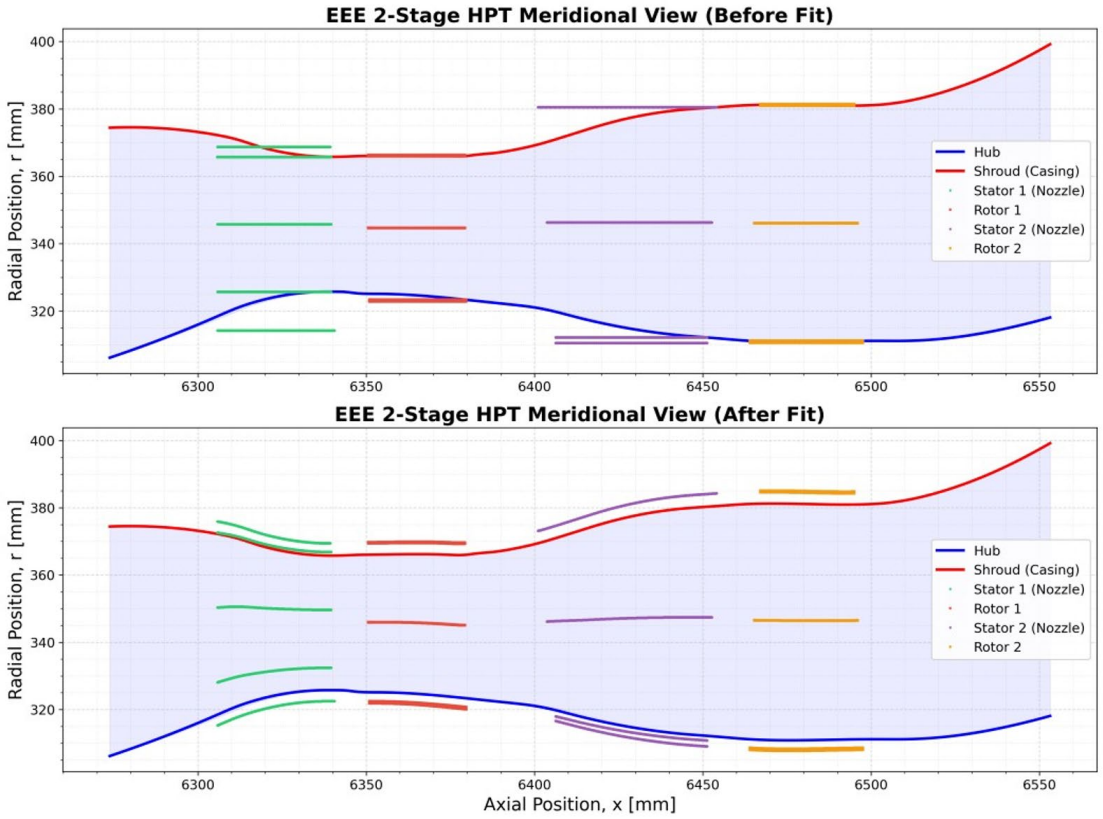
$$\eta_{TT} = \frac{w}{w + T_2 \Delta s} \tag{27b}$$

where $w = c_P (T_{01} - T_{02})$ and entropy generation reduces to:

$$\Delta s = R ln \left(\frac{P_{0R,in}}{P_{0R,out}} \right) \tag{27c}$$

P_{0R} is the relative total pressure — only irreversibilities reduce it.

EEE 2-stage high-pressure turbine



NASA / GE Energy Efficient Engine 2-stage HPT — meridional view: hub, shroud and blade rows (paper Fig. 8). Full-scale warm-air test measured 90.0 % efficiency.

Boundary conditions

Inlet: P_0 = 12.57 bar, T_0 = 1587 K, M = 0.2

Exit: P_s = 2.30 bar

Rotors: 12400 RPM

Fluid: air, variable c_p and γ

CFD reference

ADS solver, Wilcox k–ω, $y^+ < 1$, adiabatic walls, 30k iterations to convergence.

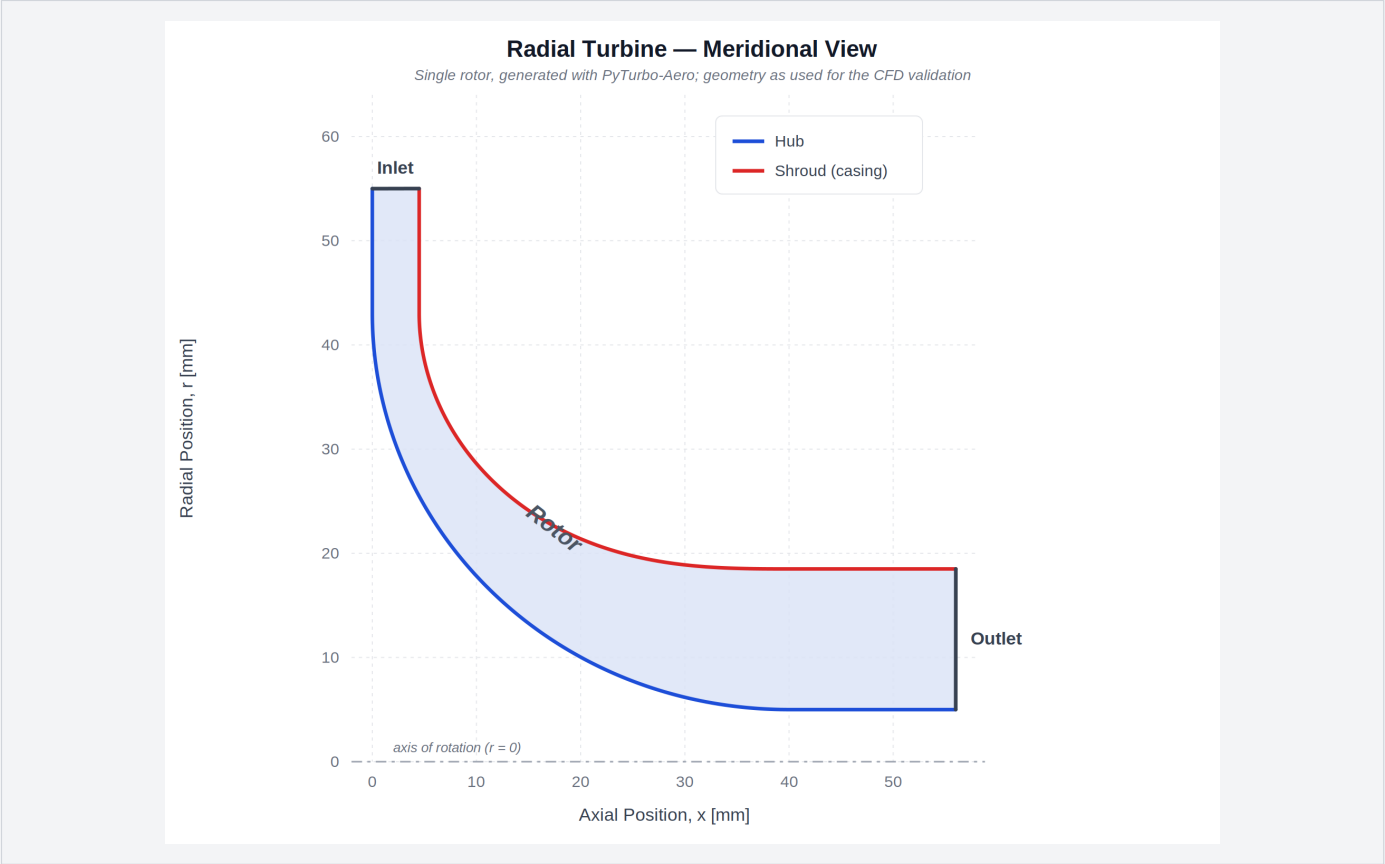
EEE Tutorial

EEE — Turbo-Design vs CFD

Quantity	Source	Stage 1	Stage 2
Efficiency η	CFD	0.962	0.970
	Turbo-Design	0.955	0.986
	Error	0.7 %	1.65 %
Stage loading ψ	CFD	1.61	1.20
	Turbo-Design	1.56	1.26
	Error	3.1 %	5.0 %
Mass flow [kg/s]	CFD	29.46 (total)	
	Turbo-Design	29.4 (total)	
	Error	0.2 %	

CFD-measured total-pressure loss prescribed per blade row (Eq. 22). Stage efficiency and mass flow agree within 2%.

Radial turbine — where the curvature term matters



Radial turbine generated with PyTurbo-Aero; geometry as used for the CFD validation.

Test the $V_r \cdot dV_m/dM$ term

This is the case where dropping the curvature term in Eq 15 would matter most.

$$\frac{1}{\rho}\frac{dP}{dr} = \frac{V_T^2}{r} - \frac{V_M^2}{r_M}cos(\phi) - V_r\frac{dV_M}{dM}$$

Inlet: P_{0R} = 50 bar, T_{0R} = 1200 K

Exit: P_{static} = 25 bar

Rotor: 50000 RPM, γ = 1.35

A zero-loss stator upstream of the rotor enforces α_1 = 0; α_2 is matched to the CFD rotor-inlet angle.

Radial turbine — Turbo-Design vs CFD

Quantity	1D model	CFD	Difference
P_0 loss (Yp)	0.136	0.136	prescribed
Total-total efficiency	0.803	0.797	+0.8 %
Total-static efficiency	0.338	0.334	+1.3 %
Total mass flow [kg/s]	0.299	0.292	+2.2 %
Enthalpy power [kW]	23.23	22.45	+3.5 %
Euler power [kW]	23.23	22.46	+3.5 %
T_0 inlet [K]	1222.9	1222.9	0.0 %
T_0 exit mean [K]	1152.6	1174.1	−1.8 %
P_0 inlet [Pa]	536,657	536,657	0.0 %
P_0 exit mean [Pa]	427,458	428,240	−0.2 %
β_1 avg [deg]	0.01	0.12	~0 %
β_2 avg [deg]	51.2	51.7	−1.0 %

Single-rotor radial turbine, γ = 1.35. Pressure loss coefficient Yp = 0.136 prescribed from CFD.

Conclusions

Summary

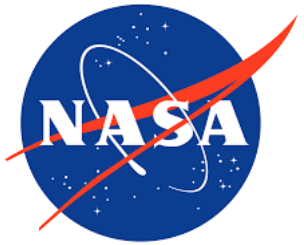
Turbo-Design is an open-source Python radial equilibrium solver that replaces the legacy Fortran codes TD2 and AXOD2. The governing equations are derived from first principles, retaining the meridional curvature terms required for radial machines. Axial and radial turbines are handled with a unified formulation. The solver was validated against CFD for the NASA EEE two-stage HPT and an internally developed radial turbine, with agreement within 2% on mass flow and efficiency when supplied with CFD-measured loss coefficients.

Future work

- Loss models tuned to specific turbomachinery stages
- Machine-learning loss models trained on high-fidelity data
- Compressor support, both axial and radial (Part II)
- Continued validation against experimental datasets

Source code, tutorials, and CFD validation cases are available at **github.com/nasa/turbo-design**

github.com/nasa/turbo-design



NASA GLENN · PURDUE · ASME GT2026-175147

Thank you

paht.juangphanich@nasa.gov | github.com/nasa/turbo-design

Paht’s open-source projects