

Technical Report

propwrap

A Software Library for Liquid Bipropellant
Thermochemical Performance Evaluation

Document type	Engineering software technical report
Software version	0.1.3
Distribution	PyPI package <code>propwrap</code> ; MIT source license
Author	Shanthosh K. V. (shanthoshkv@gmail.com)
Repository	https://github.com/shanthoshkv/propwrap
Project website	https://aboutkvs.vercel.app/propwrap.html
Date	30 July 2026

Disclaimer. This document describes theoretical, one-dimensional chemical equilibrium performance as implemented by NASA CEA (via RocketCEA). Numbers produced by the software are *not* flight-delivered engine performance and do not constitute certification evidence. The project is independent open-source work and is **not affiliated with** ISRO, DRDO, NAL, HAL, NASA, or any other agency or company.

MIT License for `propwrap` source code. Third-party engines (RocketCEA/GPL-family, Cantera, etc.) retain their own licenses.

Contents

Abstract	iv
Nomenclature	v
1 Introduction	1
1.1 Why propellant numbers matter early	1
1.2 The practical problem propwrap addresses	1
1.3 What this report covers	1
1.4 Project identity	2
2 Scope of the software	3
2.1 In scope	3
2.2 Out of scope	3
2.3 Truthfulness of numbers	4
3 Theoretical background	5
3.1 Thrust and effective exhaust velocity	5
3.2 Specific impulse	5
3.3 Characteristic velocity	6
3.4 Thrust coefficient	6
3.5 Nozzle area ratio and pressure ratio	6
3.6 Ideal-gas density check	7
3.7 Mixture ratio	7
3.8 Liquid bulk density and density impulse	7
3.9 Shifting versus frozen expansion	7
3.10 Optional efficiency factors	8
3.11 Note on γ versus c_p/c_v	8
4 Units and metrology	9
4.1 Public SI contract	9
4.2 Pressure inputs	9
4.3 Constants coded in the library	10
4.4 English $\leftrightarrow$ SI (adapter only)	10
5 Software architecture	11
5.1 Design goals	11

5.2	Layered structure	11
5.3	Major modules	12
5.4	Dependencies	12
5.5	Caching	12
6	Public interface and workflows	13
6.1	Installation	13
6.2	Minimal Python usage	13
6.3	Primary API surface (stable for 0.1.x)	13
6.4	Fair multi-pair trades	14
6.5	Command-line interface	14
6.6	Student tooling	14
7	Validation	15
7.1	What is claimed	15
7.2	Primary sources	15
7.2.1	NASA CEA methodology	15
7.2.2	RocketCEA published goldens	15
7.2.3	Metrology	16
7.3	Physics identities (must not drift)	16
7.3.1	Bug fixed during physics audit	16
7.4	Multi-propellant regression matrix	16
7.5	Test suite	17
8	Worked examples	18
8.1	Single-point kerolox	18
8.2	O/F sweep	18
8.3	Fair three-pair trade	18
8.4	Homework pack	18
9	Limitations and responsible use	19
10	Conclusions	20
A	Package layout	22
B	Document control	23

List of Figures

5.1	Logical architecture of <code>propwrap</code> . Public code never re-implements equilibrium chemistry; it orchestrates RocketCEA/Cantera and polices units.	11
-----	---	----

List of Tables

7.1	LOX/LH ₂ golden vacuum I_{sp} (RocketCEA docs conditions).	15
-----	--	----

Abstract

Liquid rocket design begins with propellant thermochemistry: mixture ratio, chamber pressure, and nozzle expansion set the ideal specific impulse, characteristic velocity, product temperature, and molecular weight that every downstream subsystem inherits. NASA’s Chemical Equilibrium with Applications (CEA) program remains the standard way to compute those numbers, but day-to-day use is still awkward—English units leak into SI notebooks, scripts drift, and multi-propellant trades are often run at a single forced mixture ratio that is fair to none of the pairs.

`propwrap` is a Python 3.10+ library that wraps NASA CEA through RocketCEA and exposes a typed, SI-first public API. It computes vacuum and sea-level specific impulse (shifting and frozen chemistry), effective exhaust velocity, characteristic velocity, thrust coefficients, station temperatures and pressures, product γ , bulk liquid density, and density impulse. Higher-level workflows support O/F sweeps, fair multi-pair trades (each pair at its own optimum O/F), multi-component blends, unit conversion, SQLite result caching, a command-line interface, optional Cantera frozen- γ cross-checks, and student-oriented lab report packs.

This report documents the engineering motivation, governing performance relations, software architecture, unit system, public interfaces, validation strategy (including published RocketCEA goldens and physics identity tests), worked examples, and known limitations. The intent is that a propulsion engineer or graduate student can read this document, install the package from PyPI, and reproduce the same theoretical numbers with a clear understanding of what they do—and do not—mean.

Keywords: liquid rocket propellants; NASA CEA; RocketCEA; specific impulse; characteristic velocity; density impulse; SI units; software validation; Python.

Nomenclature

Symbols

A_e, A_t	Exit and throat areas [m ²]
C_f	Thrust coefficient [—]
c^*	Characteristic velocity [m/s]
c_p, c_v	Specific heats at constant pressure / volume
F	Thrust [N]
g_0	Standard gravity, 9.806 65 m/s ²
γ	Isentropic exponent (performance γ from CEA)
I_{sp}	Specific impulse [s]
$\dot{m}$	Mass flow rate [kg/s]
M_w	Molecular weight [kg/kmol]
O/F	Mixture ratio, m_{ox}/m_{fuel} [—]
P_c, P_e, P_a	Chamber, exit, ambient pressure [Pa]
R	Specific gas constant; R_u universal gas constant
T_c, T_t, T_e	Chamber, throat, exit temperatures [K]
v_e	Effective exhaust velocity [m/s]
ε	Nozzle area ratio A_e/A_t [—]
η_{c^*}, η_{C_f}	Optional efficiency factors applied to c^* and C_f
ρ	Density [kg/m ³]
μ, k, Pr	Viscosity, thermal conductivity, Prandtl number

Abbreviations

CEA	Chemical Equilibrium with Applications (NASA)
CLI	Command-line interface
DfAM	Design for additive manufacturing (context only; not in <code>propwrap</code>)
LOX	Liquid oxygen
LH ₂	Liquid hydrogen
MMH	Monomethylhydrazine
N ₂ O ₄	Nitrogen tetroxide
RP-1	Rocket Propellant-1 (kerosene-class)
SI	International System of Units
UDMH	Unsymmetrical dimethylhydrazine

Chapter 1

Introduction

1.1 Why propellant numbers matter early

Before anyone sizes a turbopump or draws a cooling channel, someone has to answer a deceptively simple question: *if this fuel and oxidizer burn at this mixture ratio and chamber pressure, and expand through this area ratio, what ideal performance do we get?*

Those answers— I_{sp} , c^* , T_c , γ , M_w —propagate into every later calculation. A few seconds of I_{sp} error looks small on a slide and expensive on a mass budget. A wrong γ or molecular weight spoils a nozzle contour study before the first CAD feature is drawn. In practice the industry has used NASA CEA for decades precisely because the equilibrium chemistry problem is too messy to reinvent casually [1, 2].

1.2 The practical problem propwrap addresses

CEA itself is solid. The surrounding workflow is often not:

- Scripts and notebook cells accumulate English units (psia, Rankine, ft/s) next to SI formulas.
- Results live as parsed strings or ad-hoc dictionaries.
- “Fair” propellant trades sometimes force one O/F onto hydrogen and kerosene alike—chemically unfair and easy to publish by mistake.
- Re-running last semester’s case means hoping the same CEA deck still exists on someone’s laptop.

`propwrap` exists to make the propellant layer boring and reliable: one importable library, SI public results, typed models, reproducible tests, and workflows that make the right comparison the easy comparison.

1.3 What this report covers

Chapters 3–4 state the performance theory and the unit conventions the software enforces. Chapter 5 describes architecture and major modules. Chapter 6 summarises the public API and CLI. Chapter 7 documents validation claims and automated tests. Chapter 8 walks through representative cases. Chapter 9 states limitations honestly. References and appendices collect sources and package layout.

1.4 Project identity

- **Name:** propwrap (PyPI: propwrap)
- **Language / requirement:** Python ≥ 3.10
- **Core engine:** NASA CEA via RocketCEA [3]
- **Optional cross-check:** Cantera frozen γ [4]
- **Source license:** MIT for propwrap code; RocketCEA is GPL-family—both must be reviewed for proprietary redistribution
- **Affiliation:** Independent open source; not an agency product

Chapter 2

Scope of the software

2.1 In scope

propwrap addresses **liquid bipropellant thermochemical performance** under ideal one-dimensional CEA assumptions:

- Shifting (equilibrium) and frozen composition rocket performance
- Vacuum and sea-level specific impulse
- Effective exhaust velocity $v_e = I_{sp}g_0$
- Characteristic velocity c^* and thrust coefficient C_f
- Chamber / throat / exit temperatures and pressures
- Product γ , molecular weight, selected transport properties
- Liquid bulk density and density impulse ρI_{sp}
- O/F, P_c , and ε parametric sweeps
- Multi-pair trades at each pair's own optimum O/F
- Multi-component weight-percent blends and a propellant registry
- SI converters, CLI, SQLite cache, student homework packs

2.2 Out of scope

The library is deliberately *not* a full engine design code:

- No injector CFD, droplet combustion, or finite-rate kinetics
- No regenerative cooling channel sizing or structural margins
- No turbopump or feed-system geometry
- No trajectory, vehicle Δv , or stage optimisation
- No claim that theoretical I_{sp} equals flight-delivered I_{sp}
- No flight certification or agency endorsement

Those tools can sit *on top* of **propwrap** and pull γ , c^* , T , and I_{sp} from one tested place.

2.3 Truthfulness of numbers

Unless the user applies optional efficiency knobs η_{c^*} and η_{C_f} , every performance number is **ideal theoretical cea**. Real engines deliver less. That distinction is written into the documentation, lab packs, and this report on purpose.

Chapter 3

Theoretical background

This chapter recalls the classical rocket performance definitions that CEA implements and that **propwrap** exposes. Derivations follow standard propulsion texts [5, 6] and the NASA CEA rocket problem statement [1].

3.1 Thrust and effective exhaust velocity

The axial thrust of an ideal rocket in ambient pressure P_a is

$$F = \dot{m} v'_e + (P_e - P_a) A_e, \quad (3.1)$$

where v'_e is the exit velocity relative to the vehicle. Engineers often fold pressure thrust into an *effective* exhaust velocity v_e such that

$$F = \dot{m} v_e. \quad (3.2)$$

In vacuum ($P_a = 0$) with the usual definitions used in equilibrium performance codes, vacuum specific impulse and effective exhaust velocity are related by the conventional standard gravity:

$$\boxed{v_e = I_{\text{sp}} g_0, \quad g_0 = 9.806\,65 \text{ m/s}^2.} \quad (3.3)$$

Equation (3.3) is not a local gravitational acceleration on a particular launch pad; it is the metrological convention that converts I_{sp} in seconds to velocity in SI [7]. **propwrap** uses this constant exactly and tests that v_e fields match $I_{\text{sp}} \times g_0$ bit-for-bit within floating-point expectations.

3.2 Specific impulse

Specific impulse is thrust per unit weight-flow using g_0 :

$$I_{\text{sp}} = \frac{F}{\dot{m} g_0} = \frac{v_e}{g_0}. \quad (3.4)$$

CEA reports shifting (equilibrium composition during expansion) and frozen (composition fixed at the throat or chamber, depending on the option) values. For ideal cases of practical interest one expects

$$I_{\text{sp}}^{\text{shifting}} \geq I_{\text{sp}}^{\text{frozen}}, \quad I_{\text{sp}}^{\text{vac}} \geq I_{\text{sp}}^{\text{SL}} \quad (3.5)$$

for the same P_c and ε . **propwrap** asserts these inequalities on locked regression cases.

3.3 Characteristic velocity

Characteristic velocity measures chamber energy release and throat delivery, independent of the nozzle expansion:

$$c^* = \frac{P_c A_t}{\dot{m}}. \quad (3.6)$$

In SI, kerolox-class theoretical c^* is typically of order 1700 m/s to 1900 m/s, not the numerically similar English value in ft/s. A recurring software bug class is leaving c^* in ft/s while labelling it m/s; `propwrap` converts at the CEA adapter and tests SI bands.

3.4 Thrust coefficient

The thrust coefficient links chamber pressure and throat area to thrust:

$$C_f = \frac{F}{P_c A_t}. \quad (3.7)$$

Combining (3.2), (3.4), and (3.6) yields the vacuum identity used throughout the code base:

$$\boxed{C_f^{\text{vac}} = \frac{v_e^{\text{vac}}}{c^*} = \frac{I_{\text{sp}}^{\text{vac}} g_0}{c^*}}. \quad (3.8)$$

`propwrap` recomputes and checks (3.8) both in unit tests and in an optional runtime sanity check.

3.5 Nozzle area ratio and pressure ratio

The geometric expansion ratio is

$$\varepsilon = \frac{A_e}{A_t}. \quad (3.9)$$

For isentropic flow of a calorically perfect gas one can write closed-form Mach–area and pressure ratios in terms of γ . Real CEA products are multi-species and temperature-dependent; the code solves equilibrium (or frozen) chemistry rather than assuming constant γ . Still, the perfect-gas relations remain useful pedagogy and sanity bounds:

$$\frac{A_e}{A_t} = \frac{1}{M_e} \left[\frac{1 + \frac{\gamma-1}{2} M_e^2}{1 + \frac{\gamma-1}{2}} \right]^{\frac{\gamma+1}{2(\gamma-1)}}, \quad (3.10)$$

$$\frac{P_c}{P_e} = \left(1 + \frac{\gamma-1}{2} M_e^2 \right)^{\frac{\gamma}{\gamma-1}}. \quad (3.11)$$

`propwrap` stores P_c and P_e in pascals and verifies that P_c/P_e matches the ratio reported by the CEA path for the same case.

3.6 Ideal-gas density check

At the chamber, a useful consistency check on molecular weight and temperature is

$$\rho_c \approx \frac{P_c M_w}{R_u T_c}, \quad (3.12)$$

with $R_u = 8314.462\,618\text{ J}/(\text{kmol} \cdot \text{K})$ as coded in **propwrap** [8]. Residual differences can appear when transport or multi-phase bookkeeping differs; the automated test tolerates a small relative error on a standard LOX/RP-1 point.

3.7 Mixture ratio

The oxidizer-to-fuel mixture ratio on a mass basis is

$$\text{O/F} = \frac{\dot{m}_{\text{ox}}}{\dot{m}_{\text{fuel}}}. \quad (3.13)$$

Stoichiometric O/F depends on chemistry; maximum vacuum I_{sp} is usually fuel-rich of stoich for carbon-containing fuels. **propwrap** can mark stoich on O/F plots and can optimise O/F numerically for a fixed P_c and ε .

3.8 Liquid bulk density and density impulse

For tankage- and volume-limited stages, density-weighted performance matters. With liquid densities ρ_f , ρ_o and mass mixture ratio O/F,

$$\rho_{\text{bulk}} = \frac{(\text{O/F} + 1) \rho_f \rho_o}{\text{O/F} \rho_f + \rho_o} \quad (3.14)$$

(standard volume-additive blend of two incompressible liquids). Density impulse is then

$$I_\rho = \rho_{\text{bulk}} I_{\text{sp}}. \quad (3.15)$$

The O/F that maximises I_{sp} is not always the O/F that maximises I_ρ . **propwrap** can report both curves and rank propellant pairs by either metric.

3.9 Shifting versus frozen expansion

Shifting (equilibrium)

Composition re-equilibrates as T and P drop in the nozzle. Dissociation energy can recombine, often raising I_{sp} .

Frozen

Composition fixed after a defined station (commonly throat). More conservative for some systems; lower I_{sp} when recombination would have helped.

Neither model is “the flight answer.” Flight includes kinetics, boundary layers, divergence, finite-rate chemistry, and mixture maldistribution. Shifting and frozen are brackets on the chemistry story, not a substitute for hot-fire data [5].

3.10 Optional efficiency factors

For rough knock-downs toward delivered performance, some workflows multiply

$$c_{\text{eff}}^* = \eta_{c^*} c^*, \quad C_{f,\text{eff}} = \eta_{C_f} C_f, \quad (3.16)$$

and recombine into an estimated delivered I_{sp} . In **propwrap** these factors are explicit knobs, not calibrated engine models. They must not be confused with a validated power-balance simulation.

3.11 Note on γ versus c_p/c_v

CEA publishes a performance γ used for nozzle relations. Chamber transport c_p may include equilibrium dissociation contributions. Therefore

$$\gamma \neq \frac{c_p}{c_p - R} \quad (3.17)$$

exactly is **expected chemistry**, not evidence of a unit-conversion bug. **propwrap** documents this distinction so users do not “fix” a non-problem.

Chapter 4

Units and metrology

4.1 Public SI contract

Outside the CEA adapter, `propwrap` exposes:

Quantity	Unit	Notes
Pressure	Pa	also <code>pc_bar</code> view / input helpers
Temperature	K	
Velocity, c^* , v_e	m/s	
Density	kg/m ³	
I_{sp}	s	rocketry convention
Density impulse	s · kg/m ³	ρI_{sp}
c_p	J/(kg · K)	
Viscosity	Pa · s	

RocketCEA continues to speak English units internally (psia, Rankine, ft/s, etc.). Conversion happens once in `cea_backend`; public models store SI.

4.2 Pressure inputs

Users may enter chamber pressure as

$$P_c = \text{pc} = \text{pc_pa} \quad [\text{Pa}], \quad (4.1)$$

$$P_c = \text{pc_bar} \times 10^5 \quad [\text{bar exact}], \quad (4.2)$$

$$P_c = \text{pc_mpa} \times 10^6 \quad [\text{MPa}]. \quad (4.3)$$

The bar is exact by SI definition:

$$1 \text{ bar} = 10^5 \text{ Pa}. \quad (4.4)$$

A common student error is `pc=70` intending 70 bar. That is 70 Pa—near vacuum—and produces nonsense. `propwrap` rejects pathologically low P_c values that almost always mean “forgot the unit.”

4.3 Constants coded in the library

Symbol	Value	Role
g_0	9.806 65 m/s ²	$v_e = I_{\text{sp}} g_0$
R_u	8314.462 618 J/(kmol · K)	ideal-gas ρ check
1 bar	1×10^5 Pa	<code>pc_bar</code>
1 atm	101 325 Pa	sea-level ambient reference

4.4 English $\leftrightarrow$ SI (adapter only)

Representative conversions applied when reading RocketCEA outputs include

$$T[\text{K}] = T[\text{R}] \times \frac{5}{9}, \quad (4.5)$$

$$P[\text{Pa}] = P[\text{psia}] \times 6894.757293168, \quad (4.6)$$

$$v[\text{m/s}] = v[\text{ft/s}] \times 0.3048. \quad (4.7)$$

Users should not need these at the public API; they are listed for auditability.

Chapter 5

Software architecture

5.1 Design goals

1. **Library, not ritual** — importable, version-pinned, testable.
2. **SI wall** — English units stop at the CEA backend.
3. **Typed results** — Pydantic models; readable `print(result)`.
4. **Fair trades by default** — optimise O/F per pair before ranking.
5. **Fail loudly** — physics identities and pressure guards.

5.2 Layered structure

Figure 5.1 sketches the call path.

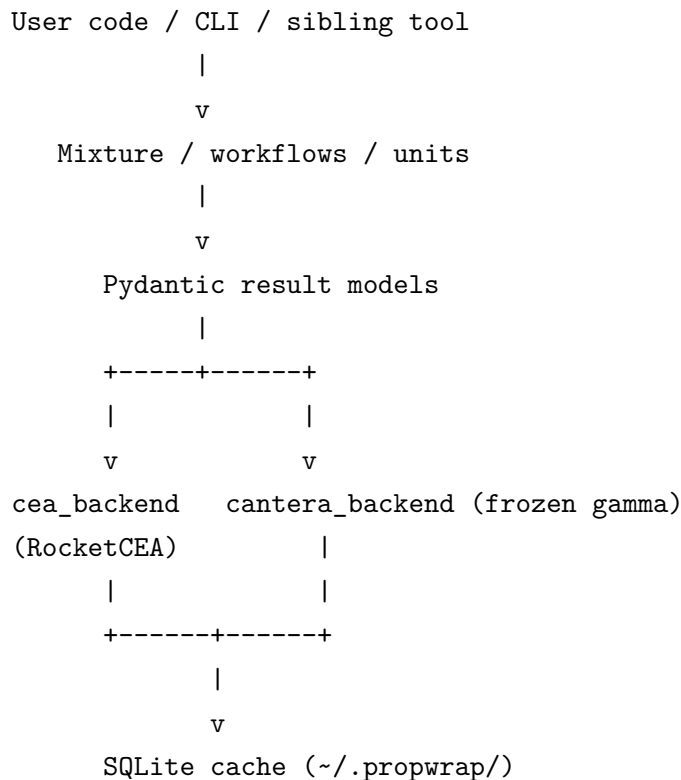


Figure 5.1: Logical architecture of `propwrap`. Public code never re-implements equilibrium chemistry; it orchestrates RocketCEA/Cantera and polices units.

5.3 Major modules

Module	Responsibility
<code>propellant.py</code>	<code>Mixture</code> primary API; evaluate, sweeps, studies
<code>cea_backend.py</code>	RocketCEA calls; English→SI conversion
<code>cantera_backend.py</code>	Frozen γ from mapped major species
<code>models.py</code>	Pydantic result types (<code>PerformanceResult</code> , ...)
<code>units.py</code>	Constants and converters
<code>workflows.py</code>	<code>characterize</code> , <code>compare_propellants</code> , blends
<code>trades.py</code>	Optimum-O/F trades; density-Isp curves
<code>registry.py</code>	Names, aliases, densities, cryo defaults
<code>blends.py</code>	Multi-component weight-percent cards
<code>cache.py</code>	SQLite result cache
<code>cli.py</code>	/ Console entry points
<code>__main__.py</code>	
<code>reports.py</code>	Markdown/CSV lab packs
<code>plotting.py</code>	Optional dark-theme figures (matplotlib)
<code>sanity.py</code>	Runtime physics identity checks

5.4 Dependencies

Hard dependencies of version 0.1.x:

- **rocketcea** — NASA CEA FORTRAN via Python [3]
- **cantera** — optional-use cross-check path; required install [4]
- **pydantic** ≥ 2 — result models
- **numpy**, **matplotlib** — numerics and plotting

5.5 Caching

Repeated identical cases hit a local SQLite database under `~/.propwrap/`. Cache keys include propellant identities, O/F, P_c , ε , and relevant options. Cache invalidation is available via CLI and API. Caching never changes physics; it only avoids re-entering CEA.

Chapter 6

Public interface and workflows

6.1 Installation

From PyPI (recommended):

```
python -m venv .venv
# Windows: .venv\Scripts\activate
pip install propwrap
propwrap run RP-1 LOX --of 2.56 --pc-bar 70 --eps 20
```

Requires Python 3.10+. On multi-Python Windows machines, prefer `py -3.12 -m pip install propwrap` so an old 3.9 pip is not used.

6.2 Minimal Python usage

```
from propwrap import Mixture, compare_propellants, convert

m = Mixture("RP-1", "LOX")
r = m.evaluate(of=2.56, pc_bar=70, eps=20)
print(r.isp_vac_shifting, r.ve_vac_shifting, r.c_star)

sw = m.scan_of((2.0, 3.2, 0.1), pc_bar=70, eps=20)
print(sw.optimum().of_ratio)

print(compare_propellants(
    ["RP-1/LOX", "CH4/LOX", "LH2/LOX"],
    pc_bar=70, eps=40,
))
print(convert(70, "bar", "Pa"))
```

Representative LOX/RP-1 point at $O/F = 2.56$, $P_c = 70$ bar, $\varepsilon = 20$ (theoretical shifting vacuum):

$$I_{sp}^{\text{vac}} \approx 343.7 \text{ s}, \quad v_e \approx 3370 \text{ m/s}, \quad c^* \approx 1798 \text{ m/s}. \quad (6.1)$$

6.3 Primary API surface (stable for 0.1.x)

- `Mixture` / `Propellant` — `evaluate`, `scan_of`, density impulse, gamma profiles
- `compare_propellants`, `characterize`, `define_blend`

- `convert`, `propwrap.units`
- `PerformanceResult` fields documented in the API reference
- CLI: `run`, `scan-of`, `characterize`, `compare-pairs`, `homework`, ...

May still change within 0.1.x: plot styling, Cantera cross-check details, efficiency-knob presentation, internal cache layout.

6.4 Fair multi-pair trades

A trade that ranks LOX/LH₂ against LOX/RP-1 at the same O/F = 2.5 is not a propellant trade; it is a self-inflicted handicap on one chemistry. The default workflow is:

1. Fix P_c and ε (mission-class assumptions).
2. For each pair, scan O/F and take the optimum (e.g. max vacuum shifting I_{sp} or max ρI_{sp}).
3. Rank pairs on the chosen metric.

That is what `compare_propellants` encodes.

6.5 Command-line interface

```
propwrap run RP-1 LOX --of 2.56 --pc-bar 70 --eps 20
propwrap scan-of RP-1 LOX --pc-bar 70 --eps 20 --range 2.0 3.5 0.1
propwrap compare-pairs --combos "RP-1/LOX,LH2/LOX,CH4/LOX" --pc-bar 70 --eps 40
propwrap homework keroxlox --name YourName
```

Module form (0.1.1+): `python -m propwrap ...`

6.6 Student tooling

Coursework environments rarely match research desktops. `propwrap` ships:

- Install notes for lab PCs
- Beginner walkthrough and concept notes (units, frozen/shifting, fair trades)
- One-page cheat sheet and sample lab assignment
- `propwrap homework` packs: summary, assumptions, CSV, figures
- Case presets: student lab / booster / upper stage

Chapter 7

Validation

7.1 What is claimed

Claim	Supported?	Mechanism
Match RocketCEA/CEA for same inputs	Yes	Bit-level I_{sp} tests
SI public units	Yes	Guards + ρ check
$v_e = I_{sp}g_0$	Yes	Identity tests
$C_f^{vac} = v_e/c^*$	Yes	Identity tests
P_c/P_e consistency	Yes	Identity tests
Equal to flight Merlin / RS-25 I_{sp}	No	Explicitly excluded
Cantera γ identical to CEA	No	Approximate frozen compare

7.2 Primary sources

7.2.1 NASA CEA methodology

Gordon and McBride, NASA RP-1311 Parts I and II [1, 2], define the equilibrium/frozen rocket performance problem. `propwrap` does not re-derive the chemistry solver; it wraps the industry-standard implementation and owns API integrity.

7.2.2 RocketCEA published goldens

The RocketCEA QuickStart publishes LOX/LH₂ vacuum I_{sp} at $P_c = 100$ psia, $\varepsilon = 40$ [3]. `propwrap` matches `CEA_Obj.get_Isp` to machine precision for O/F = 2...8 on the locked stack (verified 2026-07-29).

Table 7.1: LOX/LH₂ golden vacuum I_{sp} (RocketCEA docs conditions).

O/F	I_{sp}^{vac} [s]	Role
default <code>get_Isp</code>	374.3036...	install smoke
2	424.36	fuel-rich
4	453.13	
5	453.24	near peak
6	448.19	
8	424.70	ox-rich

7.2.3 Metrology

g_0 and the exact bar follow BIPM/SI conventions [7].

7.3 Physics identities (must not drift)

Automated tests (`tests/test_physics_identities.py`) lock:

1. Equation (3.3)
2. Equation (3.8)
3. P_c/P_e vs stored pressures
4. Chamber density check (3.12)
5. $T_e < T_t < T_c$
6. Typical $\gamma_e \geq \gamma_c$
7. Shifting $\geq$ frozen; vac $\geq$ SL on locked cases
8. c^* SI magnitude band
9. Density impulse product (3.15)

7.3.1 Bug fixed during physics audit

`gamma_and_temp_at_eps` previously returned exit temperature twice instead of chamber and exit. The bug is fixed and covered by tests (audit date 2026-07-29).

7.4 Multi-propellant regression matrix

Representative theoretical anchors (CEA via RocketCEA; not flight):

Pair	O/F	P_c	ϵ	I_{sp}^{vac} (shift.)
RP-1/LOX	2.56	70 bar	20	~ 343.7 s
RP-1/LOX	2.3	~ 1000 psia	40	~ 352.3 s
LH ₂ /LOX	5.0	~ 1000 psia	40	~ 454.9 s
LH ₂ /LOX (RS-25-class)	5.5	100 bar	69	~ 463.7 s <i>theory</i>
CH ₄ /LOX	3.2	~ 1000 psia	40	~ 367.6 s
MMH/N ₂ O ₄	2.0	~ 1000 psia	40	~ 338.6 s

Flight contrast (not used as code targets): RS-25 delivered vacuum I_{sp} is about 452 s; pure CEA at similar class conditions sits higher. Merlin-class flight kerolox I_{sp} is well below pure theoretical CEA. Mixing the two without labelling is a documentation failure; the catalog marks the gap deliberately.

7.5 Test suite

On the development stack, the full suite reports on the order of **132** passing tests spanning physics identities, validation catalog, known values, units, cache, sweeps, registry/trades, student helpers, and UX guards.

```
pytest tests/test_physics_identities.py tests/test_validation_suite.py -v
pytest -q
```

Chapter 8

Worked examples

8.1 Single-point kerolox

Inputs: RP-1 / LOX, O/F = 2.56, $P_c = 70$ bar, $\varepsilon = 20$.

Outputs (theoretical shifting, representative run): $I_{sp}^{vac} \approx 343.7$ s, $v_e \approx 3370$ m/s, $c^* \approx 1798$ m/s, $T_c \sim 3670$ K class, P_c stored as 7×10^6 Pa.

Use: classroom baseline; regression lock REG-RP1-LOX-70bar-eps20.

8.2 O/F sweep

Scan O/F $\in [2.0, 3.2]$ at fixed P_c , ε ; locate maximum vacuum shifting I_{sp} and optionally maximum ρI_{sp} . Expect the I_{sp} peak fuel-rich of stoichiometric for kerolox-class chemistry.

8.3 Fair three-pair trade

Compare RP-1/LOX, CH₄/LOX, and LH₂/LOX at the same P_c and ε , each at its own optimum O/F. LH₂ leads on I_{sp} ; dense hydrocarbons lead on ρI_{sp} . Both rankings are valid—they answer different vehicle questions.

8.4 Homework pack

```
propwrap homework kerolox --name Ada
```

Produces summary markdown, assumptions text, CSV tables, and figures suitable for a lab report without rebuilding a plotting stack from scratch.

Chapter 9

Limitations and responsible use

1. **Ideal 1-D cea** — no kinetics, multiphase losses, injector efficiency, or nozzle divergence losses unless the user adds them.
2. **Shifting/frozen brackets** — flight sits elsewhere.
3. **Efficiency knobs** — rough, not a validated engine model.
4. **Custom propellant cards** — structure-checked, not thermodynamically certified by this project.
5. **Licensing** — MIT for propwrap; review RocketCEA (GPL-family) and other dependencies before proprietary shipping.
6. **Not flight-qualified software.**
7. **No agency affiliation.**

If a number will appear in a design review, label it *theoretical CEA* or *estimated with η* , and keep hot-fire / flight data in a separate column.

Chapter 10

Conclusions

`propwrap` packages a well-known physics engine (NASA CEA via RocketCEA) behind a modern Python interface that prefers SI units, typed results, and fair propellant trades. The contribution is not a new chemistry model; it is engineering hygiene: conversions that do not drift, identities that fail tests when broken, and workflows that make correct comparisons convenient.

Version 0.1.x is suitable for education, early trade studies, and as a propellant backbone under other tools. It is not a substitute for detailed engine design analysis or certification evidence. With that boundary understood, the library does the job it claims: give the same theoretical propellant numbers tomorrow that it gave today, in units a SI codebase can trust.

Bibliography

- [1] S. Gordon and B. J. McBride, *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications, I. Analysis*, NASA Reference Publication 1311, 1994. NTRS: <https://ntrs.nasa.gov/citations/19950013764>.
- [2] S. Gordon and B. J. McBride, *Computer Program for Calculation of Complex Chemical Equilibrium Compositions and Applications, II. Users Manual and Program Description*, NASA Reference Publication 1311, 1996.
- [3] C. Taylor / Applied Python, *RocketCEA* documentation and QuickStart goldens, <https://rocketcea.readthedocs.io/>.
- [4] D. G. Goodwin, H. K. Moffat, I. Schoegl, R. L. Speth, and B. W. Weber, *Cantera: An object-oriented software toolkit for chemical kinetics, thermodynamics, and transport processes*, <https://www.cantera.org>.
- [5] G. P. Sutton and O. Biblarz, *Rocket Propulsion Elements*, 9th ed., John Wiley & Sons, 2017.
- [6] D. K. Huzel and D. H. Huang, *Modern Engineering for Design of Liquid-Propellant Rocket Engines*, AIAA Progress in Astronautics and Aeronautics, Vol. 147, 1992.
- [7] BIPM / CGPM, conventional standard acceleration due to gravity $g_0 = 9.806\,65\,\text{m/s}^2$, International System of Units (SI) documentation, <https://www.bipm.org/>.
- [8] CODATA recommended values of the fundamental physical constants (universal gas constant as used in software checks).
- [9] S. K. V., *propwrap* source repository, <https://github.com/shanthoshkv/propwrap>.
- [10] S. K. V., *propwrap* project website, <https://aboutkvs.vercel.app/propwrap.html>.
- [11] *propwrap* on the Python Package Index, <https://pypi.org/project/propwrap/>.

Appendix A

Package layout

```
propwrap/  
  pyproject.toml  
  README.md  
  LICENSE  
  src/propwrap/      # library  
  tests/              # pytest suite + validation_catalog.json  
  docs/               # user docs + this report  
  examples/           # runnable scripts  
  website/            # single-file HTML site
```

Appendix B

Document control

Version	Date	Notes
0.1	2026-07-30	Initial technical report aligned with software v0.1.3

End of report.