EES Professional (Academic) vs. the Python Scientific Stack

A Comprehensive Comparative Evaluation for University Engineering, Mathematics, and Science Instruction

United States Merchant Marine Academy
Department of Marine Engineering

June 9, 2026

Accuracy Notice: Where exact version numbers, pricing, or feature lists could not be confirmed from authoritative primary sources, this is stated explicitly. No false statements are made; where data is unavailable it is labeled as such.

This document was prepared by:
Steven L. Pike
Associate Professor of Marine Engineering
U.S. Merchant Marine Academy, F318

and

Dr. Joshua Friedman
Math and Science Department
U.S. Merchant Marine Academy

Executive Summary (TL;DR)

> *For a single thermal/fluids course taught to non-programmers, EES Professional Academic is the more efficient teaching tool. For a full curriculum, research, and transferable career skills, Python decisively exceeds EES.*

EES Professional Academic for one department: $4,750 plus mandatory annual Academic Update Service (AUS) renewal (F-Chart, June 2026). The license is suspended if AUS is not renewed annually for licenses purchased after June 1, 2025.

Python scientific stack: Free and open-source, with hidden time/labor costs for environment management and IT support.

Recommended approach: Teach EES early for physical insight and declarative problem-solving; introduce Python progressively for breadth, automation, and career-relevant skills.

1. Overview and Philosophy

EES — Declarative, Equation-Oriented

EES (Engineering Equation Solver, F-Chart Software LLC, Madison WI) is a commercial Windows program that numerically solves thousands of coupled nonlinear algebraic and differential equations. The user types the governing equations in any order; a built-in Newton-type solver finds the values satisfying the entire system simultaneously. Unit consistency is checked automatically across the whole equation set. The Professional Academic license solves up to 12,000 variables and equations simultaneously.

EES's defining characteristic is that it is declarative: the student transcribes physics directly as equations, without sequencing, rearranging, or choosing a numerical algorithm. This dramatically lowers the barrier to entry for engineering students whose goal is thermodynamics or heat transfer — not numerical methods.

Python — Imperative, Library-Based

The Python scientific stack is an open, imperative, library-based ecosystem: Python the language plus NumPy (arrays), SciPy (solvers, integration, optimization), SymPy (symbolic algebra), Matplotlib (plotting), pandas (tabular data), CoolProp (fluid properties), Cantera (combustion/ideal-gas), and domain libraries. There is no single vendor; all components are independently maintained under permissive open-source licenses.

Python requires the student to choose a solution algorithm, write a residual function, supply initial guesses, and manage execution order — a higher cognitive burden initially, but one that builds durable computational literacy.

Learning Curve Summary

A new engineering student can typically solve a basic thermodynamics problem in EES within the first lab session (approximately 1–2 hours of instruction), because equations are entered directly and units are checked automatically. With Python, students without prior programming experience typically require several weeks before independently solving coupled physics problems. The payoff is transferable computational skill across data science, machine learning, and automation. Note: these are reasoned estimates based on tool design; they are not from a controlled time study.

2. Real (Non-Ideal) Thermophysical Properties

EES Built-in Real Fluid Library

EES Professional Academic includes high-accuracy equations of state for a large library of real fluids. The following categories and substances are documented from F-Chart sources:

Steam / Water

- Steam_IAPWS — IAPWS-95 scientific formulation (high accuracy, all regions)
- Steam / Water / R718 — IAPWS-IF97 industrial formulation (faster; used for most engineering calculations)
- Ice properties below 0°C from Hyland and Wexler

Refrigerants — Pure

- Includes: R-134a, R-22, R-744 (CO2), R-717 (ammonia/NH3), R-1234yf, R-1234ze, R-410A, and many others
- Most pure refrigerants use Helmholtz-form reference equations of state (high accuracy across all regions)
- Some older fluids use the Martin-Hou EOS, which F-Chart documents as having ~1% specific-volume accuracy below 1.5× critical density

Refrigerant Blends and Mixtures (Professional Version Includes Additional Blends)

F-Chart documents the following predefined refrigerant mixtures as included in EES Professional (some require Professional license):

- Ammonia-water (NH3H2O) — viscosity and thermal conductivity added for both liquid and vapor phases in v12.119
- R404A, R407C, R407H, R410A, R417A, R422D, R423A, R427A, R438A, R448A, R449A, R450A, R452A, R452B, R453A, R454A, R454B, R454C, R455A, R457A, R460A, R463A, R466A, R469A, R470A, R470B, R471A, R473A, R474A, R475A, R477A, R479A, R507A, R508B, R513A, R513B, R514A, R515A, R515B, R516A
- F-Chart continues to add refrigerant blends in the 12.x series (e.g., R473A, R464A, R477A, R474A, R475A, R479A added in 2026)

Cryogens and Specialty Fluids

- Nitrogen, Oxygen, Hydrogen, Helium, Helium-3 (Professional), Argon
- Sodium and Potassium (liquid/vapor, Professional license only) — for liquid-metal applications

Hydrocarbons

- Methane, Ethane, Propane, Butane, and others available as real fluids
- Natural Gas is recognized as an ideal-gas mixture (v11.357+); default composition can be modified

Brines / Incompressible Substances

- Water-based secondary refrigerants as a function of temperature and mass concentration; correlations largely from Melinder (IIR, 2010)
- Solid materials including metals (e.g., Haynes 282) with thermal conductivity, density, and specific heat
- Solid CO2 and solid CH4

Moist Air / Psychrometrics

- AirH2O — ideal-gas psychrometric functions: Wetbulb, Dewpoint, RelHum, HumRat, etc.
- Optional ASHRAE LibHuAirProp add-in for real moist-air properties

Seawater

- A Sea Water Property Library is listed in the EES help contents table of contents (fchart.com/ees/eeshelp/cnt.htm). Full details of the formulation used were not confirmed in this review.

NASA Ideal Gas / Condensed Library (Optional Add-in)

- 1,262 ideal gases and 762 condensed substances
- Gordon-McBride/NASA polynomial thermodynamic data
- Molar mass, specific heat, enthalpy, and entropy; the $Reference RXN directive sets enthalpy to the enthalpy of formation at 25°C/101.3 kPa for combustion calculations

Python Equivalents for Real Fluid Properties

EES Property Domain	Python Library	License / Cost	Notes
Steam IAPWS-95 / IF97	CoolProp (Water), iapws, pyXSteam	Open source / Free	CoolProp uses high-accuracy Helmholtz EOS; iapws is pure Python IAPWS-95/97 implementation
Refrigerants — pure	CoolProp (R134a, R410A, NH3, CO2, etc.), thermo	Open source / Free	CoolProp covers 122 components with EOS and transport; thermo covers a broader chemical database
Refrigerant blends	CoolProp mixture calculations, thermo	Open source / Free	CoolProp supports user-defined mixtures; fewer predefined blends than EES
Cryogens / hydrocarbons	CoolProp, thermo, chemicals	Open source / Free	CoolProp includes N2, O2, H2, He, Ar, CH4, C2H6, C3H8, etc.
Brines / incompressible	CoolProp INCOMP:: backend	Open source / Free	e.g., INCOMP::MEG-50% ethylene glycol-water
Moist air / psychrometrics	psychrolib (ASHRAE-based), CoolProp HAPropsSI	Open source / Free	psychrolib follows ASHRAE Handbook-Fundamentals; multi-language
Metals / solids	ht library (390 materials), thermo	Open source / Free	Caleb Bell; insulation/material property database

NIST gold-standard reference	REFPROP via ctREFPROP or CoolProp REFPROP backend	Commercial — $325 perpetual single-user (NIST SRD23V10)	Same EOS database EES_REFPROP uses; academic pricing not separately listed by NIST

CoolProp Note: CoolProp (open source, MIT license) ships Helmholtz equations of state for 122 components (per coolprop.org), humid air via HAPropsSI, and incompressible/brine fluids. Both EES (Steam_IAPWS) and CoolProp (HEOS::Water) implement IAPWS-95 with the same triple-point reference state; numerical results match to the digits shown in the USMMA worked problems document.

3. Ideal Gas Properties

EES Ideal Gas Substances

In EES, substances named by chemical formula (N2, CO2, CH4, O2, CO, H2O, C3H8, etc.) are modeled as ideal gases with enthalpy and entropy from JANAF table references. The enthalpy datum is elements = 0 at 298 K; entropy is from the Third Law. This supports combustion energy balances. Air and AirH2O are modeled as ideal-gas mixtures.

The optional NASA library (loadable add-in) provides thermodynamic data for 1,262 ideal gases and 762 condensed substances via Gordon-McBride/NASA polynomial fits. Functions include h(T), s°(T), Pr(T), and vr(T). The $Reference RXN directive sets the enthalpy at 25°C/101.3 kPa to the enthalpy of formation, enabling combustion calculations even for real fluids.

Python Equivalents for Ideal Gas / Combustion

Capability	Python Library	License / Cost	Notes
Ideal gas thermo, combustion, kinetics	Cantera	Open source / Free (BSD-3, NumFOCUS)	Premier combustion/kinetics tool; chemical equilibrium, flames, reactor networks; current stable v3.2.0
Ideal gas properties, NASA polynomials	thermo (Caleb Bell)	Open source / Free (MIT)	20,000+ chemicals; NASA 7/9-coefficient polynomials
Chemical property database	chemicals (Caleb Bell)	Open source / Free (MIT)	Broad chemical database companion to thermo
Symbolic thermodynamic derivations	SymPy	Open source / Free (BSD)	No EES equivalent; full computer algebra
Unit handling	pint	Open source / Free (BSD)	Explicit unit tracking; not automatic like EES

4. Special Built-in Libraries and Functions

EES Professional Academic auto-loads multiple specialty function libraries (stored as .LIB files in the USERLIB folder). The following are documented from F-Chart sources and the EES help index:

a) Heat Transfer Library (Nellis & Klein)

Developed in conjunction with Klein & Nellis heat-transfer textbooks. Auto-loaded at startup in Commercial and Professional versions. Provides:

- Conduction shape factors (ht.conduction analogs)
- Radiation view factors between surface geometries
- Forced convection Nusselt correlations (internal flow: PipeFlow, PipeFlow_ND, PipeFlow_local; external flow; free convection)
- Boiling and condensation correlations
- Fin efficiency calculations
- Blackbody radiation functions and CO2/H2O gas emittance (VoltTC thermocouple curve fits also listed)

Python equivalent: ht library (Caleb Bell, MIT). Modules: ht.conduction, ht.conv_internal, ht.conv_external, ht.conv_free_immersed, ht.boiling_flow, ht.boiling_nucleic, ht.condensation, ht.conv_tube_bank, ht.conv_plate, ht.radiation (Stefan-Boltzmann, blackbody spectral radiance), ht.hx (effectiveness-NTU, P-NTU, LMTD, TEMA methods), insulation material database (390 materials).

> *Gap: Radiation view factors are listed in EES's Heat Transfer Library. Their presence in the Python ht library was not confirmed in this review — a potential EES advantage for radiation problems.*

b) Fluid Flow / Pipe Flow Functions

- PipeFlow, PipeFlow_f, DuctFlow_f — fully developed internal flow with friction factor and Nusselt number
- Moody chart / Colebrook implicit friction factor (Professional: no rearrangement needed)
- Minor Losses and Flow Measurement library — fitting K-values, orifice/venturi meters
- Drag Coefficient library

Python equivalent: fluids library (Caleb Bell, MIT): friction_factor (Colebrook, Churchill, etc.), Moody, K_from_f, K_from_L_equiv, fitting loss correlations for 100+ fitting types, compressible flow functions.

c) Psychrometric Functions

- AirH2O fluid: Wetbulb, Dewpoint, RelHum, HumRat, Enthalpy, Volume — ideal-gas
- Optional ASHRAE LibHuAirProp for real moist-air properties

Python equivalents: psychrolib (MIT, follows ASHRAE Handbook-Fundamentals); CoolProp HAPropsSI for humid air properties.

d) Combustion Functions

- JANAF/NASA ideal gas data for CO2, H2O, CO, CH4, C3H8, O2, N2, air, etc.
- $Reference RXN directive for enthalpy-of-formation-based combustion balances
- Combustor3_CL (Component Library, Professional) — high-level combustor model

Python equivalent: Cantera (BSD-3, free). Full chemical kinetics, equilibrium, flame modeling, and reactor networks. Substantially more capable than EES for combustion, at the cost of a steeper learning curve.

e) Mechanical Design Library

A Mechanical Design Library is listed as an auto-loadable EES library in the F-Chart help index. The exact inventory of functions (stress analysis, fatigue, gear design, etc.) was not fully confirmed in the sources reviewed for this document. Professors who use this library should verify current contents with F-Chart.

Python equivalent: No single equivalent library. scipy.optimize, sympy, and custom code are used; specialized packages such as pyFEA exist but are not as integrated as the EES library.

f) Electrical Properties Functions

A dedicated EES electrical/semiconductor material-properties library was not confirmed in sources reviewed. The EES help index lists VoltTC (thermocouple curve-fit functions) but not a general electrical materials library. Treat as: data unavailable / not confirmed.

g) Component Libraries (Professional)

The Professional license provides access to the Component Library — high-level component models for energy systems. F-Chart documentation confirms the following components:

- HeatExchanger4_CL through HeatExchanger6_CL — heat exchanger models
- Combustor3_CL — combustor model
- Evaporator1_CL — evaporator model
- Condenser2_CL — condenser model
- Fan, compressor, and pump models (exact model names not fully enumerated in reviewed sources)
- Shock and ejector models listed in the help index

Python equivalents: ht.hx provides heat exchanger sizing (effectiveness-NTU, LMTD, P-NTU). For full system simulation with component models, open-source process simulators such as DWSim (MIT) provide Python scripting. No bundled component library as convenient as EES Professional's Component Library exists in the Python ecosystem.

h) Thermomechanical and Structural Material Properties

- EES Incompressible Substances library includes solid metals (e.g., Haynes 282), solid CO2, solid CH4, and other solids with density, thermal conductivity, and specific heat
- Sodium and Potassium (liquid/vapor) in Professional version for liquid-metal applications

Python equivalents: ht library (Caleb Bell): 390-material insulation/structural database including metals, polymers, and insulation with thermal conductivity and density. The thermo library also provides material property data.

i) Peng-Robinson and Redlich-Kwong-Soave Property Functions

The EES help index lists dedicated Peng-Robinson (PR) and Redlich-Kwong-Soave (RKS) cubic EOS function libraries as separate loadable items, in addition to the default high-accuracy Helmholtz EOS data. These allow calculations with these simpler cubic equations of state, which are widely used in chemical engineering.

Python equivalents: thermo library (Caleb Bell) implements PR, SRK, and many other cubic and reference EOS natively. chemicals provides underlying property data. Both are MIT-licensed.

5. Report Writing with Live Data and Graphics

EES Professional Report Window

The Professional license provides a Report Window that operates like a word processor: choice of fonts, font sizes, text styles, subscripts/superscripts, color, tabs, and indents. Key features confirmed from F-Chart:

- Graphics can be pasted from the Formatted Equations window (rendered LaTeX-style equations), from Plot windows, from Diagram windows, or from other applications
- EES variables and plots can be embedded in the Report Window and are automatically updated as their values change — 'live' data integration
- Equations in the Formatted Equations window can be converted to LaTeX or MathType format (Professional); MathType equations paste directly into Microsoft Word
- Reports are inherently tied to the EES file; they are not standalone reproducible documents

Limitation: EES's report features are convenient for self-contained homework write-ups but are not a general-purpose document system. Reports cannot be versioned independently or rendered in multiple output formats.

Python Report Writing Equivalents

Tool	Capability	License / Cost	Notes for Instructors
Jupyter Notebooks	Interactive code + prose + equations + plots in one document; nbconvert to PDF/HTML/LaTeX	Open source / Free (BSD)	Industry standard for reproducible science; excellent for teaching; nbconvert requires LaTeX for PDF
Quarto	Scientific publishing platform: HTML, PDF, Word, ePub, slides from one source; executable Python/R/Julia code, LaTeX equations, cross-references	Open source / Free (MIT)	Next-generation R Markdown; superior multi-format output; strong academic adoption; replaces pweave/knitr for Python
python-docx	Programmatic Word document generation from Python	Open source / Free (MIT)	Used to generate this document; full table, heading, image, and style support
ReportLab	Programmatic PDF generation with full layout control	Open source (BSD) + commercial option	More powerful than nbconvert for custom-formatted PDFs; steeper API
Plotly + Dash	Interactive web-based reports and dashboards	Open source / Free (MIT)	For interactive results exploration; requires web server or localhost
LaTeX via subprocess	Full LaTeX compilation from Python-generated .tex files	Open source / Free	Maximum typographic quality; highest learning curve

Verdict: EES's Report Window is convenient for live, self-contained engineering reports tied to a single .ees file. Python/Quarto/Jupyter produce publication-grade, reproducible, multi-format output that can be version-controlled and reused across courses — at the cost of a steeper learning curve.

6. Diagram Window, Child Diagrams, and GUI Interface (Professional)

> *This section covers one of EES Professional's most distinctive features — and one for which there is no close Python equivalent without substantial custom GUI development.*

6.1 The Diagram Window — Overview

All EES Commercial and Professional licenses include a Diagram window. It serves two distinct purposes: (1) displaying a schematic diagram of the system (state-point locations, component labels) to help interpret the Equations window; and (2) functioning as a live, interactive input/output GUI and report interface linked directly to EES variables.

In the Commercial version, a picture can be imported and drawing objects, text, input variables, and calculated output variables can be superimposed on it. A Calculate button can be placed on the Diagram window to initiate the solver from within the GUI rather than from the menu.

6.2 Professional License Diagram Window Enhancements

The Professional license greatly expands the Diagram window's capabilities. All of the following are confirmed from F-Chart's official feature documentation:

Hot Areas and Child Diagram Windows

- 'Hot areas' can be defined in any Diagram window — clickable regions that, when clicked, open additional Child Diagram windows
- Child Diagram Windows have all the capabilities of a parent Diagram window, including the ability to launch further Child Diagram windows (unlimited nesting depth)
- Navigation between parent and child windows is provided; a Home button returns to the parent
- Hot areas can be repositioned after creation

Interactive Input Controls

- Slider controls for numerical input (in addition to text edit boxes)
- Edit boxes can be configured to accept full EES equations involving variables and numerical values — not just scalar entry
- Radio button groups and check box groups can be placed on any Diagram or Child Diagram window; selecting a button executes specific EES equations or code branches
- The $INPUT directive (Professional) can prompt the user to enter a variable value when solving

Buttons and Links

- Calculate buttons to initiate solver from the GUI
- Link buttons to start external programs, run an EES macro, show a movie, open another EES file, or perform other tasks
- Save/load input buttons to preserve and restore the state of input variables
- Print buttons for the Diagram window
- Help buttons that open an HTML help file
- Hyperlinks: text items and buttons can act as hyperlinks to open a browser to a specified URL, open a PDF file, or start a program

Live Plots in the Diagram Window

With the Professional license, 'live' plots from the Plot window can be copied and embedded in the Diagram window. These plots are automatically updated when the underlying data changes. Plots can be made transparent so that text and other graphics in the Diagram window show through them.

Animation

The Professional license supports full animation of the Diagram and Plot windows:

- Graphical objects and text in the Diagram window can have attributes (location, size, angle, color) controlled by EES variables
- Text items can be assigned to string variables that are updated dynamically in an EES program
- A red circle indicator tracks the current value of a variable in a Plot window as calculations proceed
- Animations created with a Professional license can be played in Distributable and Commercial versions

Drawing Tools (Professional)

- Polyline and polygon construction tools in the toolbar
- Polylines can be smoothed to cubic Bezier curves
- Polygons can be partially transparent for colored shading while allowing underlying objects to remain visible
- Text in the Diagram window can be displayed at any angle between -90° and 90°

6.3 Make Distributable — The Professional Deployment Feature

A critically important Professional license capability for academic use is Make Distributable:

- Creates a special-purpose EES executable that runs up to 6 preselected .ees problems
- EES, the problem files, and all supporting files are packaged into a single executable that can be freely distributed
- The end user does NOT need to own an EES license to run calculations, create/change/view plots and tables in the distributable
- Equations can be securely hidden from the end user, if desired — enabling faculty to distribute working problem solvers to students without exposing the solution logic

Pedagogical implication: a professor can build a complete interactive problem set — with a schematic diagram, slider inputs, calculated results displayed on the diagram, live plots, and embedded help text — and distribute it as a standalone executable. Students interact with a polished, textbook-like interface without needing an EES license.

6.4 EES AI Librarian (Professional, v12.x)

The Professional license includes an AI Librarian: an AI assistant trained on all EES documentation, YouTube tutorials, and the Mastering EES book. It provides answers to EES-related questions, debugs code, and gives hints. This is a distinct feature not replicated by the Python ecosystem's general-purpose documentation tools.

6.5 Python GUI Equivalents — Comparison

There is no single Python tool that replicates the EES Diagram window's integrated schematic + live-variable + GUI-control paradigm for non-programmers. Comparable capabilities require explicit GUI development:

EES Feature	Python Approach	Difficulty	Gap Assessment
Schematic with superimposed live variables	Matplotlib figure with annotated text updated via callbacks, or Plotly Dash layout	High — requires GUI programming	EES requires no code; Python requires explicit layout and callback design
Slider input controls	ipywidgets (Jupyter), Dash sliders, Tkinter/PyQt sliders	Medium–High	ipywidgets is usable in Jupyter; requires programming
Radio buttons / check boxes	ipywidgets, Dash, Tkinter/PyQt	Medium–High	Functional but requires coding
Hot areas / child windows	Custom Tkinter/PyQt dialogs, Dash multi-page	High	No out-of-the-box equivalent
Animation of diagram objects	Matplotlib FuncAnimation, Plotly animation frames	High	Functional but requires significant code
Live plot embedded in schematic	Plotly Dash figure in layout, or Matplotlib subplot	Medium	Less integrated than EES; requires web server for Dash
Make Distributable (self-contained exe)	PyInstaller packaging of a Tkinter/Dash app	High	Feasible but requires substantial development effort
Distributable without source code visible	PyInstaller with bytecode, Cython compilation	High	Possible but complex; not a one-click workflow

Verdict: The EES Professional Diagram window — particularly the combination of interactive schematics, child windows, slider/radio-button controls, live variable display, animation, and Make Distributable — provides a polished GUI capability that would require weeks to months of Python GUI programming to replicate. For building interactive teaching tools for non-programmers, EES Professional has no practical Python equivalent at comparable development effort.

7. Tables, Batch Calls, and External Application Interfaces

7.1 EES Table Types

Parametric Table

The Parametric Table is EES's primary tool for parametric studies. It is similar in concept to a spreadsheet: the user specifies values for one or more independent variables across rows, and EES solves the entire equation system for each row, filling in the dependent variables. Confirmed features:

- Any number of variables can be included as columns
- Professional license: unlimited rows (limited only by available memory); Commercial license: limited number of rows
- The Solve Table command (Calculate menu) runs the full Newton solver for every row
- Results can be plotted directly from the Parametric Table (New Plot Window), creating publication-quality X-Y, parametric, and other plot types
- Multiple Parametric Tables can exist in a single EES file
- Data can be copied to/from Excel with headers and in E-format (12 significant figures) to avoid rounding loss
- The Alter Values command and column-header dialog allow equation-based fill of column values (Professional: accepts EES variable names in place of numbers)
- Professional: alternate unit sets for column display; values can be entered and plotted in either unit set
- Professional: crosshair tool in the Plot window can identify the table row corresponding to a selected plot point
- Min/Max Table command runs optimization across Parametric Table rows
- Uncertainty Propagation Table command applies uncertainty analysis across all rows

Array Table (Arrays Window)

EES supports indexed array variables using bracket notation (e.g., T[1..10], h[3]). Array variables are displayed and edited in the Arrays window, which is separate from the Solution window.

- Array variables can be indexed with integer expressions
- DUPLICATE loops allow compact equation entry over arrays: DUPLICATE i=1,N: h[i] = ... END
- The $CopyToLookup directive (Professional) copies calculated scalar and array variables to a Lookup table column; that data can then be written to an external file
- Array data can be plotted from the Arrays window
- Arrays are particularly useful for discretized problems (finite difference, cycle-state-point sequences)

Lookup Table

The Lookup Table stores user-supplied tabular data that can be interpolated and used in EES equations via Interpolate, Interpolate2D, and table-access functions. Confirmed features:

- Data can be imported from text files, Excel, or other external sources
- Any column's data can be entered using an equation referencing data in other cells, EES variables, or EES functions (via the column-header dialog)
- Professional: the calculated data in Lookup Table columns can be automatically updated, making the Lookup table operate similarly to a spreadsheet

- Professional: unlimited rows
- Professional: Fast Fourier Transform (FFT) transform available on Lookup table column data
- Data in Lookup tables can be written to external files in multiple formats
- Two-dimensional interpolation (Interpolate2D) is supported
- NIST REFPROP .FLD (fluid coefficient) files can be read by EES Professional and used as property data in the same manner as built-in fluids

Integral Table

When EES solves differential equations via its Integral function, the Integral Table captures the state variable values at each integration step — analogous to the dense output of scipy.integrate.solve_ivp. This provides a tabular time-history of ODE solutions for plotting or export.

7.2 External Application Interfaces

Historical Note: DDE Removed in v12.000

Dynamic Data Exchange (DDE) was the original EES mechanism for communicating with external programs. Per F-Chart's official documentation: 'DDE is not supported in EES versions 12.000 or newer.' DDE was dropped because MATLAB and other programs were phasing it out. The replacement mechanisms are described below.

Current Mechanism: Windows SendMessage (v12.131+, Professional)

Starting with version 12.131 (released September 13, 2025), the Professional license responds to Windows SendMessage commands, enabling any external application to call EES:

- EXCEL, MATLAB, LabView, and Python can all call EES using SendMessage to return property data or other calculated results
- Chapter 22 of the Mastering EES book covers this in detail
- Both numerical values and plots can be transferred from EES to Excel using this mechanism
- EES then writes output files (via $Export or $SaveTable directives) that the calling application reads

Command-Line / Macro File Interface (All Versions)

Any external program can call EES via the operating system command line:

- EES can be started with a .ees file and /Solve parameter: EES.EXE myfile.EES /solve starts EES in a hidden state, opens and solves the file, writes output, and quits
- Alternatively, a .emf macro file can be provided as the command-line parameter
- This approach causes EES to quit after execution; for repeated calls it is less efficient (EES must reload each time)
- Used in this manner, EES can be directed to solve equations and write results to a text file without appearing on the screen

Macro Language (Professional)

The EES macro language (stored in .emf files or typed in the Macro window) provides a full scripting capability:

- Looping and conditional logic (Repeat...Until, IF...THEN...ELSE including multi-line IF constructs added in v12.133)
- All menu commands can be implemented via macro instructions

- Macro commands can be automatically recorded using the Build Macro command in the File menu
- Macros can be called from within EES (CallMacro) or triggered externally

Specific macro command groups documented by F-Chart:

- EXCEL commands: EXCEL.Cell, EXCEL.Copy, EXCEL.FileNew, EXCEL.FileOpen, EXCEL.FileSaveAs, EXCEL.Hide, EXCEL.Paste, EXCEL.RANGE, EXCEL.Quit, EXCEL.Sheet, EXCEL.Show
- MATLAB commands: MATLAB.Open, MATLAB.Execute, MATLAB.GetVariable, MATLAB.GetTable, MATLAB.SetVariable, MATLAB.Quit
- Microsoft Word communication via macro commands
- Serial port communication: Serial.Open, Serial.Close, Serial.ReadChar, Serial.ReadNumber, Serial.ReadString, Serial.ReadtoLookup, Serial.WriteChar, Serial.WriteNumber, Serial.WriteString — enabling two-way communication with any device using serial (RS-232/USB-serial) communication, including laboratory instruments and data acquisition hardware

EES ↔ Python (Bidirectional, Professional)

The Professional license supports bidirectional communication with Python, documented on F-Chart's website:

Calling Python from EES: The CALL Python('filename.pyw', ..., In1, In2, ... : Out1, Out2, ...) syntax calls a Python script from within EES equations. The script accepts numerical inputs and returns numerical outputs. Units of inputs and outputs can be specified so EES's unit-checking algorithms apply. The $Python directive specifies the Python interpreter path. Any installed Python version is acceptable.

Calling EES from Python: Python can call EES via the Windows SendMessage command (v12.131+) or the command-line /Solve mechanism. The Python script passes inputs via text files read by EES's $Import directive; EES writes results via $Export; Python then reads the output file.

Use case: EES can serve as a property-calculation engine called from a Python data pipeline — combining EES's built-in IAPWS-95, refrigerant, and ideal-gas property databases with Python's data processing and ML ecosystem. Conversely, a Python machine-learning model can supply inputs to an EES thermal simulation.

LabView Interface

LabView can call EES using the SendMessage mechanism (v12.131+) or the command-line macro file approach. F-Chart provides an example showing LabView calling EES to return property data. Serial port macro commands additionally allow EES to communicate directly with National Instruments DAQ hardware.

7.3 Python Equivalents for Tables and Batch Calls

EES Feature	Python Equivalent	Comparison
Parametric Table	NumPy arrays + pandas DataFrames for vectorized sweeps; scipy.optimize for table-level optimization	Python vectorizes entire sweeps in one expression (np.linspace + array math); no GUI, but scriptable, reproducible, and easily exported to CSV/Excel
Array Table / DUPLICATE loops	Python lists, NumPy arrays, for loops	Python's array handling is more general; NumPy broadcasts

		naturally without DUPLICATE syntax
Lookup Table with interpolation	scipy.interpolate (interp1d, RegularGridInterpolator), pandas DataFrames	Python interpolation is flexible and fast; no built-in live-update spreadsheet feel
Lookup Table FFT	numpy.fft, scipy.fft	Equivalent functionality; more control over windowing and output format
Integral Table (ODE history)	solve_ivp dense_output=True, sol.sol(t) for any t; pandas DataFrame for tabular output	Functionally equivalent; Python also returns error estimates
2D interpolation	scipy.interpolate.RegularGridInterpolator, griddata	Functionally equivalent
Batch property calls	CoolProp PropsSI / PropsSImulti with NumPy arrays; REFPROP ctREFPROP vectorized calls	Python batch calls are typically faster for large arrays
Serial port / instrument interface	pyserial library (MIT)	pyserial provides serial communication; Python ecosystem for DAQ includes NI-DAQmx Python API
Excel integration	openpyxl, xlsxwriter, pandas to_excel, xlwings (bidirectional COM automation)	xlwings enables two-way Excel↔Python communication similar to EES macro EXCEL commands
MATLAB integration	MATLAB Engine for Python (MathWorks); scipy.io for .mat file exchange	MATLAB Engine API is the direct equivalent; bidirectional variable passing

Key observation: EES's Parametric Table GUI is unmatched for quick what-if studies without writing code. Python's NumPy vectorization is more scalable and automatable. The serial-port macro commands in EES make it a plausible laboratory data-acquisition front-end — a less commonly discussed but practically useful feature for experimental courses.

8. Batch / Programmatic Property Calls

EES

Parametric Tables run the full solver across many rows. Built-in property procedures return multiple outputs per call. The macro language and (in v12.131+) the SendMessage interface allow external programs to drive EES programmatically. The $Import/$Export directives exchange data with text files. These mechanisms make EES usable as a property-calculation server callable from other applications.

Python

CoolProp.PropsSI accepts NumPy arrays directly for vectorized batch evaluation. PropsSImulti computes multiple properties in a single call, reducing overhead. REFPROP Python bindings (ctREFPROP) similarly support vectorized calls. pandas DataFrames organize tabular property sweeps. For large batch property evaluations, Python is typically faster than EES (which must run its solver for each table row), particularly for pure property lookups that do not require solving an equation system.

9. Complex Numbers, Vector Calculations, and 2D/3D Plotting

9.1 Complex Numbers in EES

Overview and Scope

EES supports complex arithmetic throughout the entire equation system, controlled by the $Complex ON/OFF directive or the Preferences dialog. All confirmed from F-Chart documentation:

- Complex mode applies to the main program and to Subprograms. Functions, Modules, and Procedures cannot operate in complex mode, but they can be called from the main program or Subprogram regardless of whether complex mode is active.
- When complex mode is enabled, all variables are assumed to have real and imaginary components unless specifically designated as real with the $Real directive.
- The choice of i or j to represent the imaginary unit is user-selectable: $Complex ON i or $Complex ON j — matching either the mathematics or electrical-engineering convention.

Entry Formats

- Rectangular form: entered as a + b*i (or a + b*j), e.g., Z = 3 + 4*i
- Polar form: entered as magnitude < angle, e.g., Z = 5<53.13deg. The < symbol is rendered as the angle symbol □ in the Formatted Equations and Solution windows, supporting the phasor notation used in AC circuit analysis.
- The CIS function provides an alternative polar entry: Z = 5*CIS(53.13deg) is equivalent to Z = 5<53.13deg
- Angles can be designated in degrees (deg) or radians (rad); using the unit label is recommended because the degree symbol then appears in the Formatted Equations window

Internal Representation and the Solver

When EES parses a complex equation, it automatically generates two real equations — one each for the real and imaginary parts. These are identified with (r) and (i) in the Residuals and Computational Flow windows. Each complex variable X is internally renamed X_r and X_i, and these component names appear in Variable Information dialogs, Parametric Table column headers, and elsewhere. The user can set a component directly — for example, omega_i = 0 to force the imaginary part of omega to zero — but EES will report an error if a subsequent equation conflicts with an already-set component.

- The Newton-type simultaneous solver operates on the doubled real system transparently — the user still writes physics equations involving complex variables without any manual decomposition
- Because EES doubles the equation count (one per complex variable per equation), complex mode consumes the license's variable/equation budget at twice the rate

Mathematical Functions in Complex Mode

Standard mathematical functions have been modified to accept and return complex numbers where mathematically valid, including:

- Trigonometric: sin, cos, tan, and inverses — accepting complex arguments per Euler's formula
- Exponential and logarithm: exp and ln — supporting complex arguments
- The ANGLE function returns the argument (phase angle) of a complex number; ANGLEDEG and ANGLERAD return the angle in degrees and radians respectively

- The CONJ function returns the complex conjugate
- The CIS function: CIS(θ) = cos(θ) + i·sin(θ) — the unit phasor
- ABS returns the modulus (magnitude) of a complex number
- Some functions, such as Min and Max, are not accessible in complex mode (not meaningful for complex quantities)

Display and Output

- The Solution window can display complex variables in rectangular form (a + bi) or polar form (r□θ) — the display format is toggled per-variable by right-clicking and selecting Format Variable
- Polar display angle units (degrees or radians) can be set independently of the entry units
- Complex variables are displayed in both the main Solution tab and any Key Variables tab
- The Formatted Equations window renders complex equations in typeset notation, including the □ symbol for polar entries

Pedagogical Applications

Complex number support is particularly valuable for the following course contexts at USMMA and similar institutions:

- AC circuit analysis: impedances, phasors, power factor, three-phase systems
- Control systems: transfer functions, poles and zeros, frequency response
- Wave mechanics and vibrations: complex exponential solutions to ODEs
- Signal processing: Fourier analysis with complex coefficients

9.2 Python Complex Number Capabilities

Python handles complex numbers natively at the language level — no library import is required for basic complex arithmetic. NumPy and SciPy extend this to arrays and scientific functions.

EES Complex Capability	Python Equivalent	Assessment
Rectangular entry (a + b*i or j)	Python native: z = 3 + 4j (j only; i not a keyword)	Parity; Python uses j exclusively
Polar entry (r<θ, CIS)	cmath.rect(r, theta); numpy: r*np.exp(1j*theta)	Functional parity; less natural notation than EES's r<θ
Complex conjugate (CONJ)	z.conjugate() or np.conj(z)	Parity
Modulus / ABS	abs(z) or np.abs(z) or cmath.polar(z)[0]	Parity
Phase angle (ANGLE)	cmath.phase(z) in radians; math.degrees(cmath.phase(z)) for degrees	Parity; manual unit conversion
Complex trig / exp / ln	cmath.sin, cos, exp, log; numpy equivalents	Parity; cmath and numpy both support complex arguments
Automatic real/imaginary decomposition of equations	Not available — must manually decompose or use SymPy	EES advantage: no decomposition needed

Complex simultaneous equation solver	scipy.optimize.fsolve with complex arrays (cast as real 2N system); or direct complex Newton	Functional; requires manual residual construction
Phasor notation (□ display)	No built-in phasor display; custom formatting required	EES advantage for AC/circuit pedagogy
Unit checking on complex quantities	Not available in Python (pint does not handle complex units natively)	EES advantage
Symbolic complex algebra	SymPy: full symbolic complex arithmetic, conjugate, re(), im(), simplify()	Python advantage: symbolic capability EES lacks entirely
Complex FFT	numpy.fft, scipy.fft — industry-standard complex FFT	Python advantage: far more capable than EES's Lookup Table FFT

EES complex number advantage: the simultaneous solver handles complex equations without the user decomposing them into real and imaginary parts. A phasor circuit equation like Z_total = Z_1 + Z_2 is entered exactly as written; EES solves the doubled real system automatically. Python's equivalent requires either manual decomposition, casting to a real 2N array, or using SymPy for symbolic solution.

9.3 Vector Variables and Calculations in EES

Native 2D and 3D vector support was added to EES in version 11.834 (April 2024) and is confirmed from F-Chart's feature documentation and help pages. This is a relatively recent addition to EES.

Declaration and Assignment

- $Vector A B C — declares A, B, C as 3D vectors; each vector V has components V_x, V_y, V_z (shorthand for V = V_x·i + V_y·j + V_z·k)
- $Vector2D A B — declares A, B as 2D vectors; each vector V has components V_x, V_y (shorthand for V = V_x·i + V_y·j)
- Vector variables and functions that return vectors are displayed with a bold underline font in the Equations window
- 2D and 3D vectors cannot be mixed in the same equation set
- VectorAssign(Vx, Vy, Vz) — assigns component values to a declared 3D vector; units can be set with the x-component
- VectorAssign(Vx, Vy) — for 2D vectors
- VectorAssignPolar(magnitude, angle_x) — assigns a 2D vector from polar form (magnitude and angle from positive x-axis)
- VectorAssignPolar(magnitude, angle_x, angle_y) — assigns a 3D vector from polar form (magnitude plus two directional angles)

Vector Functions (Confirmed from F-Chart Documentation)

EES Function	Description	Applies to
VectorDot(A, B)	Scalar dot product A·B	2D and 3D
VectorCross(A, B)	Cross product A×B — returns a 3D vector for 3D inputs; for 2D vectors, returns the scalar representing the signed area of the parallelogram (z-component of the 3D cross product)	2D (scalar) and 3D (vector)
VectorCross_i(A,B), _j, _k	Individual i, j, k components of the 3D cross product vector	3D only
VectorMag(A)	Magnitude (Euclidean norm) of vector A	2D and 3D
VectorUnit_i(A), _j, _k	Components of the unit vector in the direction of A	3D
VectorAngle_x(A)	Angle between vector A and the positive x-axis	2D and 3D
VectorAngle_y(A)	Angle between vector A and the positive y-axis	3D
VectorAngle_z(A)	Angle between vector A and the positive z-axis	3D
VectorPolarAngle(A,B)	Angle between vectors A and B	2D and 3D

- Standard algebraic operations apply: vectors can be added, subtracted, and multiplied or divided by scalars using normal EES equation syntax
- Scalars cannot be added to or subtracted from vectors (EES enforces this dimensionally)
- Unit checking applies to vector quantities — all components carry the same units
- Vector variables participate fully in EES's simultaneous equation solver — a vector equation is automatically split into its component equations

Vector Plotting

- $VectorPlot2D Name='...' — creates a 2D vector diagram in the Plot window from declared 2D vector variables
- $VectorPlot Name='...' (equivalently $VectorPlot3D) — creates a 3D vector diagram from declared 3D vector variables
- Vectors are plotted one at a time; origin (tail) of each vector is specified; the 'Head of last vector as origin' option enables tip-to-tail chain plots
- Label vector option shows an identifying label for each plotted vector
- 2D vector plots support Auto Update — they refresh automatically when the underlying data changes
- The VectorPlot macro command automates vector plot creation from a macro or external script (added v11.855)

Pedagogical Applications for USMMA

- Statics and dynamics: force vectors, moments, free-body diagrams in 2D and 3D
- Fluid mechanics: velocity fields, pressure gradients

- Navigation: course and current vectors, resultant track
- Electromagnetics: electric and magnetic field vectors
- The vector functions integrate with EES's property and unit-checking system — a force vector in Newtons participates in energy balance equations naturally

9.4 Python Vector Calculation Equivalents

Python has mature, powerful vector and linear algebra support through NumPy. The comparison below is for vector (geometric/physics) calculations specifically, not matrix/array operations:

EES Vector Capability	Python Equivalent	Assessment
$Vector / $Vector2D declaration	numpy arrays: A = np.array([Ax, Ay, Az])	Python uses plain arrays; no special declaration syntax needed
VectorAssign / VectorAssignPolar	Direct array construction or polar: r*np.array([np.cos(theta), np.sin(theta)])	Parity; slightly more verbose
VectorDot(A,B)	np.dot(A, B) or A @ B	Parity
VectorCross(A,B)	np.cross(A, B)	Parity; np.cross handles both 2D (scalar) and 3D (vector) automatically
VectorMag(A)	np.linalg.norm(A)	Parity
Unit vector	A / np.linalg.norm(A)	Parity; no dedicated function but one-liner
VectorAngle_x, _y, _z / VectorPolarAngle	np.arccos(np.dot(A,B)/(np.linalg.norm(A)*np.linalg.norm(B)))	Parity; requires manual formula
Vector arithmetic (+, -, scalar mult/div)	NumPy array arithmetic with broadcasting	Parity; arguably more natural in NumPy
Simultaneous vector equations	scipy.optimize.fsolve with vector residuals	Requires explicit residual construction; no declarative entry
Unit checking on vectors	pint: Quantity arrays (opt-in, not automatic)	EES automatic; Python manual
Symbolic vector algebra	SymPy: Matrix, sympy.vector (CoordSys3D, divergence, curl, gradient)	Python advantage: full symbolic vector calculus not available in EES

Vector field operations (div, curl, grad)	SymPy.vector, scipy (numerical gradient)	Python advantage: EES has no vector calculus operations
Batch / array-indexed vector operations	NumPy broadcasting over arrays of vectors	Python advantage: EES handles one set of vector equations at a time

Assessment: NumPy provides fully equivalent vector arithmetic and is more powerful for batch/array operations and symbolic vector calculus (SymPy). EES's advantage is the integration with its simultaneous solver and automatic unit checking — a statics problem with unknown force vectors is solved by writing the equilibrium equations directly, with no scripting.

9.5 2D Plotting Capabilities

EES 2D Plot Types (Confirmed from F-Chart Documentation)

EES produces publication-quality 2D plots from data in any table window (Parametric, Lookup, Arrays, Integral). All plots can be printed, exported as Windows Metafile graphics, or embedded live in the Diagram window (Professional). Plot templates can be saved and reapplied (Professional).

EES 2D Plot Type	Description	License
X-Y plot	Standard Cartesian plot; multiple Y series; linear or log axes; spline interpolation option; auto-update from table	All
Bar plot	Side-by-side bar plots; Professional adds high-quality grouped bars	All; Professional for side-by-side
Area plot	Line plot with area below the line filled with a specified color	Professional only
Parametric plot (X-Y with symbol color)	X vs Y plot with symbol color representing a third variable Z	Professional only
Property plot	Plots thermodynamic property diagrams (T-s, P-h, P-v, etc.) with state points overlaid — directly from EES property databases	All
Contour-Lines	Lines of constant Z in X-Y space; auto-labeled; up to 25 contour lines	All
Contour-Bands	Color-band representation of Z=f(X,Y); up to 250 color bands; gray or spectral color schemes;	All

	interactive Z-value read from mouse position	
Contour-Gradient	Arrows showing gradient direction superimposed on a color-band plot	All
2D Vector plot	Plots declared EES vector variables as arrows in 2D space with specified origins	All (requires $Vector2D declaration)
$NewPlot / $OverlayPlot directives	Create and overlay plots from text commands in Equations window (no GUI required)	Professional
$PropertyPlot directive	Generate property diagram from Equations window text command	Professional

Additional 2D plot features confirmed from F-Chart: cubic spline curve fitting through data points; auto-update linking plot to live table data; red circle indicator tracking current variable value during iterative calculation; crosshair tool identifying table row for a selected plot point (Professional); Plot Thumbnails window (Professional); plot templates (.EPT files, Professional); transparent live plots embeddable in the Diagram window (Professional).

Python 2D Plotting Equivalents

Matplotlib is the standard Python 2D plotting library. It is significantly more powerful and customizable than EES's plot windows for programmatic use, at the cost of requiring code to configure each plot.

- Line/scatter: plt.plot(), plt.scatter() — full axis control, multiple series, any color/marker
- Bar charts: plt.bar(), plt.barh() — grouped, stacked, horizontal
- Area plots: plt.fill_between()
- Contour lines: plt.contour() with plt.clabel() for auto-labeling
- Contour bands (filled): plt.contourf() with colorbar
- Gradient arrows: plt.quiver() for 2D arrow fields
- Thermodynamic property diagrams: CoolProp + Matplotlib can generate T-s, P-h, and P-v diagrams; the PropsSI function populates saturation curve data for overlay
- Plotly: interactive versions of all above plot types; web-embeddable
- seaborn: statistical visualization built on Matplotlib — no EES equivalent
- Python plots are saved as vector (SVG, PDF, EPS) or raster (PNG, JPEG) files programmatically; no GUI required

> *Key Python advantage: Matplotlib/Plotly plots are version-controlled, reproducible, and scriptable. EES plots are GUI artifacts; they cannot be regenerated from a script without re-running EES interactively (unless the $NewPlot directive is used in Professional).*

9.6 3D Plotting Capabilities

EES 3D Plot Types (Confirmed from F-Chart Documentation)

EES's 3D plotting uses a modified GLScene open-source 3D graphics package. The X-Y-Z Plot menu provides the following 3D options:

EES 3D Plot Type	Description	License
3D Surface plot	Rotatable 3D surface Z=f(X,Y) with orthogonal or perspective projection; a 4th variable can be displayed as a color map on the surface (color variable selectable separately from Z)	All (requires 3D plotting option at setup)
3D Polygon/Points plot	3D representation of tabular data in three columns; up to 6 overlaid polygons with configurable color, line type, symbol, and filled polygon option; rotatable	Professional only
3D Vector plot	Plots declared EES 3D vector variables as arrows in 3D space	All (requires $Vector declaration)
Symbol Color plot	2D X-Y plot with symbol colors representing a third (Z) variable; color legend included	Professional only

Data gridding for surface and contour plots uses one of two algorithms: Radial Basis Function (RBF, exact interpolation at data points with optional smoothing; computationally $O(N^3)$) or Bi-Quadratic Polynomial (local weighted least squares; 2,500–10,000 grid points). The resolution and smoothing are controlled by slider bars in the dialog. After construction, 3D plots are rotated interactively by dragging the mouse; auto-update (v11.862+) keeps 3D plots synchronized with changing table data. Only one 3D Surface plot can appear per plot window, but Professional allows 3D Polygon/Points overlays on a surface.

Python 3D Plotting Equivalents

Python's 3D plotting ecosystem is substantially more powerful and flexible than EES for publication-quality and interactive 3D visualization:

EES 3D Capability	Python Equivalent	Assessment
3D Surface (rotatable)	Matplotlib mpl_toolkits.mplot3d: Axes3D.plot_surface(); full rotation in GUI or animated in scripts	Parity in basic capability; Python allows programmatic rotation and animation
3D Surface with 4th variable color map	Matplotlib facecolors=cm.viridis(normalize(data)); Plotly go.Surface(surfacecolor=)	Parity; Python's colormapping is more flexible
Contour-Lines (2D projection)	plt.contour() with clabel()	Parity
Contour-Bands (2D projection)	plt.contourf() with colorbar()	Parity
Gradient arrows (2D)	plt.quiver(X, Y, U, V)	Parity
3D vector arrows	Matplotlib ax.quiver(x,y,z,u,v,w); Plotly cone plot for 3D vector fields	Python advantage: full vector field visualization over a grid
3D Polygon/Points	Matplotlib ax.plot3D(), ax.scatter3D(), Poly3DCollection	Parity

Interactive 3D (web)	Plotly go.Figure(); Plotly Express — fully interactive, zoomable, rotatable in browser	Python advantage: EES 3D plots are GUI-only
Thermodynamic surface plots (T-s, P-v)	CoolProp + Matplotlib / Plotly — 3D phase surface diagrams	Python advantage: full programmatic generation
Animation of 3D plots	Matplotlib FuncAnimation; Plotly animation frames	Python advantage: EES 3D animation requires Professional and is GUI-only
Data gridding / interpolation	scipy.interpolate: RBFInterpolator (same method EES uses), LinearNDInterpolator, griddata	Parity; scipy.interpolate.RBFInterpolator is the direct equivalent of EES's RBF gridding
Rendering quality	GLScene (EES); Matplotlib/Plotly (Python)	Plotly produces higher-quality interactive 3D; Matplotlib produces publication-quality static

Overall 3D plotting assessment: EES's 3D Surface and Polygon plots are adequate for engineering course use and are tightly integrated with the solver output — a Parametric Table sweep automatically feeds the 3D plot with no additional code. Python's Matplotlib and Plotly provide equal or superior 3D visualization quality, with far more flexibility, scriptability, and interactivity, at the cost of requiring code to configure each plot.

9.7 Comprehensive EES vs. Python: Complex, Vector, and Plotting

Capability	EES	Python
Complex arithmetic	Native; i or j selectable; no import needed	Native (j only); cmath for math functions; no import for arithmetic
Complex equation solving	Automatic real/imaginary decomposition; Newton solver	scipy.optimize or manual decomposition; SymPy for symbolic
Phasor notation (r<θ, □)	Built-in entry and display	cmath.rect(); custom formatting
Unit checking on complex quantities	Automatic	Not supported (pint does not handle complex units)
Symbolic complex algebra	None	SymPy: full CAS (re, im, conjugate, simplify, factor over □)
2D vector declaration	$Vector2D directive (v11.834+)	NumPy array; no declaration syntax
3D vector declaration	$Vector directive (v11.834+)	NumPy array
Dot product	VectorDot()	np.dot() or @ operator
Cross product	VectorCross() — 2D scalar, 3D vector	np.cross() — same behavior
Vector magnitude	VectorMag()	np.linalg.norm()

Angles between vectors	VectorAngle_x/y/z, VectorPolarAngle	np.arccos(dot product formula)
Vector calculus (div, curl, grad)	Not supported	SymPy.vector; scipy numerical gradient
Standard 2D plots	X-Y, bar, area, contour, property; GUI-based	Matplotlib: all types and more; scriptable
Property diagrams (T-s, P-h)	Built-in Property Plot with EES fluid databases	CoolProp + Matplotlib; requires code
2D vector plots	$VectorPlot2D; auto-update	plt.quiver(); manual update
3D surface plots	3D Surface (rotatable GUI); color 4th variable	Matplotlib plot_surface(); Plotly go.Surface()
3D vector plots	$VectorPlot; GUI-based	ax.quiver(); Plotly cone — vector field grids
Interactive/web 3D plots	Not available	Plotly: fully interactive in browser
Scripted/automated plot generation	$NewPlot/$OverlayPlot (Professional)	Matplotlib/Plotly: fully scriptable
Animation	Professional Diagram Window only	Matplotlib FuncAnimation; Plotly frames

10. Printing Capabilities and Formatted Equation Output

EES's printing system is closely integrated with its Formatted Equations window — a unique feature that renders algebraic equations in typeset mathematical notation directly from the plain-text Equations window, with no manual typesetting required by the user.

9.1 The Formatted Equations Window

The Formatted Equations window renders every equation entered in the Equations window in standard mathematical notation — fractions, subscripts, superscripts, Greek letters, and operator symbols — automatically. Confirmed features from F-Chart documentation:

- Equations are displayed in typeset mathematical notation. Variables whose names contain underscores are rendered with subscripts; vertical-bar | introduces superscripts. Greek symbol names (alpha, beta, gamma, etc.) render as Greek characters. Dot-notation variants (_dot, _bar, _hat, _tilde, _ddot, _prime, _dprime) render as expected mathematical decorations.
- Comments enclosed in double quotes are displayed in the Formatted Equations window; comments in braces {} or starting with // are suppressed, allowing the user to control what narrative text accompanies printed equations.
- Display Units for Constants and Display Units for Variables options show unit labels alongside numerical constants and variables directly in the formatted display.
- Hyperlinks embedded in comments (http:, https:, file: prefixes) are active in the Formatted Equations window and in any PDF generated from it.
- Equations cannot be edited in the Formatted Equations window; clicking any equation returns focus to the corresponding line in the Equations window for editing.
- The entire window or selected equations can be selected and copied to the clipboard as Windows Metafile pictures (vector graphics), suitable for pasting into Word, PowerPoint, and other applications.

9.2 Formatted Equation Export Options

Three distinct clipboard copy formats are provided, confirmed from F-Chart's Formatted Equations help page:

Copy as EES Picture (All Licenses)

Selected equations are copied to the clipboard as Windows Metafile (WMF) vector pictures. These can be pasted directly into Word, PowerPoint, or any OLE-capable application. The images are not editable after pasting; they render at screen resolution and, per F-Chart's own note, 'generally do not look as good as the MathType equations.'

Copy as LaTeX (Professional License)

Selected equations are translated to LaTeX markup and placed on the clipboard as plain text. The LaTeX code can be pasted into any LaTeX editor or document. This is the highest-quality export path for academic paper and report writing, producing typeset-quality output through any LaTeX compiler.

Copy as MathType (Professional License)

Selected equations are translated to MathType OLE objects (Windows Object Linking and Embedding format) and placed on the clipboard. Requirements: MathType must be installed at a

specific path (C:\Program Files (x86)\MathType\System\ in 32 or 64-bit variant). After pasting into Word, double-clicking the equation opens MathType for further editing. MathType equations can also be pasted into the EES Diagram window (but cannot be edited there after pasting).

9.3 Print Command (All Licenses)

The standard File > Print command (keyboard shortcut Ctrl+P) prints the currently active EES window to the system printer. Individual windows that can be printed include:

- Equations window — plain-text source as entered
- Formatted Equations window — typeset mathematical notation (the primary printed equation output for student work)
- Solution window — all variable values and units, organized in alphabetical order; Key Variables tab printed separately if defined
- Arrays window — indexed array variable values
- Residuals window — equation residuals and calculation order
- Parametric Table — all columns and rows
- Lookup Table — tabular data contents
- Integral Table — ODE integration history
- Plot windows — publication-quality X-Y, parametric, bar, area, contour, 3D, and vector plots
- Diagram window — schematic with superimposed values and graphics
- Report Window (Professional) — word-processor-formatted report with live variables and plots
- Macro window — macro script text

The Printer Setup command (PrintSetup macro command) controls printer selection, page orientation, and margins. The font used for printing is set separately from the screen display font via the Printer Display tab in Preferences — confirmed from F-Chart's General Display documentation.

9.4 LaTeX/PDF Print (File Menu, All Licenses)

The LaTeX/PDF Print command in the File menu is EES's integrated report-generation pathway. Confirmed from F-Chart's LaTeX/PDF Print help page:

Purpose: 'To provide a high-quality printout of all EES windows including the Diagram, Equations, Solutions, Tables and Plots' (F-Chart documentation verbatim).

- Generates a LaTeX2e .tex source file (ASCII) from the EES file's current state; a LaTeX compiler (MikTeX, which F-Chart provides installation instructions for) processes this into a .pdf
- Color option: if selected, the Diagram window, plots, and comments appear in the same colors as on screen; comments in the Equations window are rendered in their assigned highlight colors
- Number of columns for Solution and table display can be specified to control page layout
- Display PDF document checkbox: if checked, the default PDF reader is opened automatically to display the result
- Delete TEX document checkbox: if checked, the intermediate .tex file is deleted, leaving only the .pdf
- Include Hyperlinks checkbox: visible hyperlinks (http:/https:) in the Report window are active in the resulting PDF
- Limitation: only the main Diagram window is included; child Diagram windows are not printed via this command

9.5 LaTeXPDF Macro Command (Professional License)

The LaTeXPDF macro command provides programmatic, scriptable PDF generation — all confirmed from F-Chart:

Syntax: LaTeXPDF 'Filename' /Show -D -E -S -T -A -P -L -I

- /Show — displays the created PDF immediately after generation
- -D — excludes the Diagram window from the output
- -E — excludes the Equations window content
- -S — excludes the Solution window(s)
- -T — excludes all Parametric tables
- -A — excludes the Arrays table
- -P — excludes all plot windows (by default all plots are included)
- -L — excludes all Lookup tables (by default all Lookup tables are included)
- -I — excludes the Integral table

This macro command enables automated report generation: a macro can solve the equations, run a Parametric Table sweep, and call LaTeXPDF to produce a finished PDF — entirely unattended. The macro can be called from Python, MATLAB, Excel, or LabView via the SendMessage interface, making fully automated engineering report generation from an external workflow feasible.

Requirement: MikTeX LaTeX software must be installed on the system. This is a non-trivial dependency for student machines.

9.6 Print Button in the Diagram Window

A Print button can be placed on any Diagram or Child Diagram window (confirmed from F-Chart). When clicked, it prints the current Diagram window. This enables student-facing GUI interfaces — built with the Diagram window — to include a one-click print function without requiring access to the File menu.

9.7 Report Window Print (Professional)

The Report Window (Professional license) functions as an integrated word processor within EES. Its print button produces a formatted printed report containing:

- Formatted prose text with full font, size, color, tab, and indent control
- Graphics pasted from the Formatted Equations window (typeset equations), Plot windows, Diagram windows, or external applications
- Live EES variables — name, value, and/or units — that are automatically updated to current solver values before printing
- Live EES plots — automatically updated to the current Plot window state
- Hyperlinks to EES windows (\EES_Solution, \EES_Plot, etc.) and to external URLs and local files
- Import from and Export to .RTF and .TXT files; RTF export preserves graphics, enabling use of letterhead templates

Implication: a complete, formatted engineering homework solution — with typeset equations, numerical results, and embedded plots — can be printed directly from EES in one step, without leaving the application.

9.8 Comparison: Python Equation Formatting and Printing

Python has no built-in equivalent to EES's Formatted Equations window. Equation rendering requires an explicit toolchain chosen and set up by the instructor or student:

EES Capability	Python / Open-Source Equivalent	License / Cost	Gap Assessment
Formatted Equations window (auto-render)	No direct equivalent; SymPy sp.latex() generates LaTeX from symbolic expressions; sympy.pprint() renders in terminal	Open source / Free	EES auto-renders from plain-text equations; Python requires equations to be re-expressed as SymPy objects — significant extra work
Copy as LaTeX (from solver equations)	SymPy .latex() on symbolic expression; manual LaTeX authoring	Free	SymPy requires equations to be written symbolically; cannot derive LaTeX from a CoolProp or SciPy call automatically
Copy as MathType (OLE)	No equivalent	—	Windows OLE integration not replicated in Python ecosystem
Print any window (Ctrl+P)	No 'windows' to print; Jupyter: File > Print or nbconvert; Matplotlib: plt.savefig(); pandas: DataFrame.to_string()	Free	Python has no single-window print model; each output type uses a different export method
LaTeX/PDF full-file report (all windows)	Quarto: renders code + prose + equations + plots to PDF via LaTeX/Typst; Jupyter nbconvert --to pdf	Free (requires LaTeX or Typst installation)	Quarto/Jupyter are more powerful and flexible but require explicit document authoring; EES assembles the report automatically from the active file
Live variable values in printed report	Quarto/Jupyter: code output cells update when re-executed	Free	Functionally equivalent but requires re-execution; EES updates live within the Report Window
Print button on interactive GUI	Matplotlib print_figure(); ipywidgets button calling plt.savefig(); Dash callback	Free	Requires custom programming; EES provides this with no code
Automated PDF from macro/script	Quarto CLI (quarto render); nbconvert CLI; LaTeX via subprocess	Free	Python automated PDF generation is mature and powerful; LaTeX installation required for PDF output in both EES and Python paths
Equation color coding in print	LaTeX color package; Jupyter CSS styling	Free	Both tools support color in printed output

Printer font/margin control	Matplotlib rcParams; LaTeX geometry package	Free	More granular control available in Python/LaTeX, but requires configuration

Key verdict: EES's Formatted Equations window is a genuinely unique capability — it automatically renders the solver's plain-text equation input as typeset mathematics, with no additional work by the student or instructor. No Python tool does this automatically; SymPy can generate LaTeX from symbolic expressions, but requires the equations to be re-entered as Python SymPy objects, not derived automatically from property-function calls or solver equations. For producing a clean, typeset printed solution from a homework problem, EES requires zero additional effort beyond solving; Python requires explicit document authoring in Quarto or Jupyter.

11. Custom Procedures, Functions, and Modules

EES

- FUNCTION blocks: user-defined functions returning a single value; can be called from anywhere in the Equations window
- PROCEDURE blocks: user-defined procedures returning multiple outputs; similar calling format to built-in EES procedures
- MODULE and SUBPROGRAM: independent equation-solving units within an EES file; can have their own variable namespaces
- Internal Library files (.LIB): EES files containing functions/procedures that auto-load from the USERLIB folder at startup
- External compiled routines: .DLF (dynamic-link functions, C/C++/Pascal/Delphi), .FDL (FORTRAN array-passing procedures), .DLP (linked-list procedures); 64-bit variants (.DLF64, .FDL64, .DLP64) for the 64-bit Professional license
- NIST REFPROP .FLD coefficient files can be read directly (Professional) to access any REFPROP fluid
- Tabbed Equations windows (Professional) partition large programs into logical sections with $InsertTab
- Variable Information files (.VAR) store and restore guess values, units, limits across files

Python

Python's extensibility is the foundation of its entire ecosystem: standard functions, classes, modules, packages, and pip-installable libraries cover every use case. Compiled code integration: ctypes and cffi call C shared libraries directly; f2py wraps FORTRAN 77/90/95 subroutines with NumPy array interfaces; Cython and Numba provide JIT compilation of Python code. PyInstaller packages applications into standalone executables (analogous to Make Distributable). Python's extensibility model is far more general and powerful than EES's.

12. Textbook Example Libraries (EES Academic Advantage)

A genuine EES strength in academic settings is its tight integration with specific engineering textbooks, several authored by EES's own creators (Klein, Nellis). Confirmed from F-Chart documentation:

- Heat Transfer Library — developed with Klein & Nellis, Heat Transfer (Cambridge, 2009) and Introduction to Engineering Heat Transfer (Cambridge, 2020); auto-loaded
- Klein & Nellis, Thermodynamics (Cambridge, 2012) — companion EES problems
- Mitchell & Braun, HVAC in Buildings — companion EES files
- Duffie & Beckman, Solar Engineering of Thermal Processes — downloadable SETP4 library via 'Load Textbook' menu
- The 'Load Textbook' menu command loads textbook-specific function libraries and problem files
- EES Professional Academic also includes EESyGrader — an automated grading tool for EES homework assignments (noted in F-Chart Academic license page)

Python: No single bundled textbook library equivalent. Open educational resources include CoolProp-based online thermodynamics texts, and many textbook authors (e.g., Cengel) publish companion code repositories on GitHub. The absence of a bundled, auto-loading textbook library is a real gap compared to EES for courses built around Klein/Nellis materials.

13. Additional EES Professional Capabilities Without Python Equivalents

Automatic System-Wide Unit Checking

EES checks the dimensional consistency of every equation in the system and reports the result. This is automatic — not opt-in — and requires no programmer effort. Python's pint library checks units only where the programmer explicitly wraps quantities in Quantity objects; there is no automatic, whole-program unit audit.

Uncertainty Propagation (Professional)

The EES Uncertainty Propagation dialog computes the uncertainty of up to 50 calculated variables in terms of the uncertainties of up to 200 measured variables, using automatic partial derivatives. The percentage contribution of each measurement to the total uncertainty is displayed. The Professional license can save uncertainty configurations to .UNC files for reuse. This is a one-dialog workflow with no programming.

Python equivalent: the uncertainties package (open source, BSD) propagates uncertainties through expressions analytically using the linear error propagation formula. It requires the programmer to explicitly wrap each measured variable. The result is functionally equivalent but requires more programmer effort and has no GUI.

Optimization (Professional Additional Methods)

The Professional license adds three optimization algorithms beyond the Commercial version's Variable Metric and Direct Search methods: Genetic (global, slow), Nelder-Mead Simplex, and DIRECT (global, efficient). Results can be logged to a text file. A Sensitivity table shows how the optimum depends on each independent variable.

Python equivalent: scipy.optimize provides Nelder-Mead, genetic algorithm variants (via scipy.optimize.differential_evolution), and many other methods. The Python ecosystem exceeds EES for optimization breadth.

Complex Variables

EES supports complex-variable arithmetic throughout the equation system — all built-in property and mathematical functions accept complex arguments where mathematically valid. Python's NumPy natively handles complex arrays; SymPy handles complex symbolic expressions. This is parity between the tools.

14. Learning Curve and Time-to-Proficiency

Dimension	EES Professional Academic	Python Scientific Stack
New student — first problem	~1–2 hours (estimated); equations entered as written; units auto-checked	Several weeks to independent problem-solving for students without programming background
New student — full thermal/fluids course	Low ramp; can focus on physics rather than algorithms	Significant investment; pays back as transferable skill across courses
Professor — course setup	Low; install on department machines or student PCs; load textbook libraries	Higher; environment management (conda/pip), JupyterHub or lab setup, version conflicts
Professor — creating teaching tools	Moderate; Diagram window GUI building requires learning EES's drawing/control system	High; custom GUI (ipywidgets, Dash, Tkinter) requires programming expertise
Ongoing student support	Low; single Windows application; consistent behavior	Higher; heterogeneous student environments (Windows/Mac/Linux), package conflicts
Career transferability	Limited to EES-using organizations	High; Python is among the most widely used languages in engineering and science careers
Platform	Windows only (officially)	Cross-platform (Windows, macOS, Linux)

Time-to-proficiency estimates are reasoned approximations based on tool design and documentation; they are not from controlled time studies. Actual learning times vary substantially with instructor support and student backgrounds.

15. Maintenance Costs: Time and Money

EES Professional Academic — Current Pricing (F-Chart, confirmed June 2026)

License Type	Price	Category	Notes
32-bit Academic Professional (single department)	$4,750	Commercial (Academic)	Includes 1 year AUS; mandatory annual renewal if purchased after June 1 2025; license suspended if AUS lapses
Academic Commercial site license	$3,000	Commercial (Academic)	For educational institutions; site-wide, not department-restricted
Student License	$160/year	Commercial (Student)	~3,000 equation limit; 1-year duration; similar to single-user Commercial
AUS renewal — expired < 1 year	$1,000	Commercial (Academic)	Annual Update Service renewal
AUS renewal — expired 1–4 years	$2,000	Commercial (Academic)	Cannot renew after 4 years; new license required
64-bit Professional upgrade	Additional cost; price not confirmed	Commercial	Requires current IUTS subscription
EES_REFPROP interface	Price not confirmed	Commercial	Interface to link EES with NIST REFPROP
Mastering EES eBook (single)	$25	Commercial	Textbook; 5-copy dept. package $125
EES Demo	Free	—	50-equation limit; 30-day expiry

Python Stack — License Costs

Component	Cost	License Type	Notes
Python (CPython)	Free	Open source (PSF)	
NumPy, SciPy, Matplotlib, pandas, SymPy	Free	Open source (BSD-style)	Core scientific stack
CoolProp	Free	Open source (MIT)	122 components; IAPWS-95, refrigerants, cryogens, humid air

Cantera	Free	Open source (BSD-3, NumFOCUS)	Combustion, kinetics, ideal gas; stable v3.2.0
thermo / chemicals / fluids / ht (Caleb Bell)	Free	Open source (MIT)	Heat transfer, fluid flow, thermodynamic properties
psychrolib	Free	Open source (MIT)	ASHRAE psychrometrics
iapws / pyXSteam	Free	Open source (MIT/LGPL)	Steam/water properties
uncertainties	Free	Open source (BSD)	Uncertainty propagation
Jupyter / JupyterHub	Free (server infra cost for hosted deployment)	Open source (BSD)	Lab deployment requires server or cloud
Quarto	Free	Open source (MIT)	Scientific publishing
NIST REFPROP	$325 perpetual single-user	Commercial (NIST SRD23V10)	Gold-standard property database; separate academic pricing not listed by NIST
xlwings (Excel integration)	Free (open source) / Pro: price not confirmed	Open source / Commercial	Bidirectional Excel↔Python
MATLAB Engine for Python	Requires MATLAB license	Commercial (MathWorks)	Academic MATLAB pricing varies by institution

Hidden Costs: Python

- IT staff time for environment setup, JupyterHub server administration, and troubleshooting student environment issues
- Faculty time for debugging package version conflicts across student machines (Windows/Mac/Linux heterogeneity)
- Potential cloud compute costs if JupyterHub is hosted on AWS/GCP/Azure rather than on-premises
- These are real costs but are time/labor costs, not software license fees

16. Areas Where EES Has No Practical Python Equivalent

- Formatted Equations window: automatic typeset rendering of plain-text solver equations as mathematical notation, with LaTeX and MathType export — no re-entry in a different system required
- Complex phasor entry and display: r<θ angle notation entry with □ display in the Solution window, unit checking on complex quantities, and automatic real/imaginary equation decomposition — no Python tool provides all three together
- Automatic system-wide unit consistency checking across the entire equation set — no opt-in required
- Truly declarative, order-independent equation entry — the student does not sequence or rearrange equations
- Built-in Uncertainty Propagation GUI dialog (up to 50 results × 200 measured variables) — one-dialog workflow with no coding
- The integrated package for non-programmers: property databases + simultaneous solver + unit checking + GUI, usable without writing code
- Diagram Window with live-variable schematics, slider/radio-button inputs, child windows, animation, and Make Distributable — no coding required to build interactive teaching tools
- Make Distributable: standalone executable packaging of up to 6 problems with equation-hiding capability, deliverable to students without an EES license
- Serial port macro commands for laboratory instrument communication — integrated within the same scripting environment as the solver and property functions
- AI Librarian trained on EES-specific documentation (Professional v12.x)
- EESyGrader automated grading tool for EES homework (Academic license)
- Auto Complete for EES function names, fluid names, and keywords in the Equations window (Professional)

17. Areas Where Python Substantially Exceeds EES

- Symbolic mathematics: SymPy provides full computer algebra (symbolic differentiation, integration, equation solving, simplification). EES has no symbolic capability.
- Version control: Python source files are plain text; Git/GitHub workflows are natural. EES .ees files are binary/proprietary and are not diff-able or mergeable.
- Machine learning and data science: scikit-learn, TensorFlow, PyTorch, Keras, XGBoost — no EES equivalent.
- Data pipelines and automation: pandas, Airflow, scripted batch processing — EES has no comparable automation ecosystem.
- Web deployment: Flask, Django, Dash, Streamlit — EES cannot deploy calculations to the web.
- Large-scale numerical computation: sparse solvers (scipy.sparse), GPU computing (CuPy, JAX), parallel computing (multiprocessing, Dask) — EES is limited to its Newton solver on a single CPU.
- Community and ecosystem: Python is among the most widely used programming languages worldwide with hundreds of thousands of packages. EES has a small, single-vendor community.
- Long-term availability and sustainability: all core Python scientific libraries are permissively licensed and community-maintained, independent of any single vendor. EES depends entirely on F-Chart Software and now requires ongoing AUS payments.
- Cross-platform operation: Python runs natively on Windows, macOS, and Linux. EES is officially Windows-only.
- Reproducibility and open science: Python code with pinned dependencies and Jupyter/Quarto notebooks are the standard for reproducible computational research. EES .ees files are not reproducible in this sense.

18. Master Feature Comparison Table

Dimension	EES Professional Academic	Python Scientific Stack
Computational paradigm	Declarative, equation-oriented; order-independent	Imperative, library-based; procedural
Simultaneous solver	Built-in Newton-type; up to 12,000 variables (32-bit Pro)	scipy.optimize (fsolve, brentq, minimize); numpy.linalg.solve
Automatic unit checking	Yes — system-wide, automatic	No — manual via pint only
Real-fluid properties	Built-in; hundreds of substances (IAPWS-95, refrigerants, cryogens, hydrocarbons, blends, brines)	CoolProp (122 components, free); thermo; REFPROP ($325)
Ideal gas / combustion	JANAF + NASA (1,262 + 762 substances); $Reference RXN	Cantera (free, BSD); thermo; chemicals
Psychrometrics	AirH2O built-in; ASHRAE add-in	psychrolib (free, MIT); CoolProp HAPropsSI
Heat-transfer library	Nellis & Klein (bundled); view factors, shape factors, fins, boiling/condensation	ht library (MIT); view factors not confirmed in ht
Fluid-flow library	PipeFlow, Moody, minor losses, drag coefficients	fluids library (MIT); friction_factor, Moody, K-values
Component library	Heat exchangers, combustors, evaporators, condensors (Professional)	ht.hx (NTU/LMTD); DWSim for full process simulation
Symbolic math	None	SymPy (full CAS; free)
Uncertainty propagation	Built-in GUI dialog (up to 50×200); .UNC save files	uncertainties package (free); no GUI
Parametric studies	Parametric Table GUI (unlimited rows, Professional)	NumPy vectorization; pandas; scriptable
Array variables	DUPLICATE loops; Arrays window	NumPy arrays; Python lists; pandas
Lookup tables	Tabular data store; 1D/2D interpolation; FFT (Professional)	pandas; scipy.interpolate; numpy.fft
ODE integration	Integral function; Integral Table	scipy.integrate.solve_ivp (dense output, error estimates)
Optimization	5 methods (Professional): Variable Metric, Direct Search, Genetic, Nelder-Mead, DIRECT	scipy.optimize (15+ methods); global: differential_evolution, basinhopping
Curve fitting	Built-in; Curve Fit Table (Professional) with many pre-built forms	scipy.optimize.curve_fit; lmfit; sklearn

Diagram Window / GUI	Integrated: schematics, live variables, sliders, radio buttons, child windows, animation, Make Distributable	Custom: ipywidgets, Dash, Tkinter, PyQt; high development effort
Complex numbers	Native; i or j; automatic real/imaginary decomposition; phasor display; unit checking	Python native (j only); cmath; no auto-decomposition; no phasor display
2D/3D vector algebra	$Vector/$Vector2D (v11.834+); VectorDot, VectorCross, VectorMag, etc.; unit-aware	NumPy np.dot, np.cross, np.linalg.norm; more powerful for arrays and vector calculus (SymPy)
2D plots	X-Y, bar, area, contour, property, vector, symbol-color; GUI-based; auto-update	Matplotlib, Plotly: all types + more; scriptable; version-controlled
3D plots	3D Surface (rotatable), 3D Polygon/Points, 3D Vector; GLScene; GUI-based	Matplotlib, Plotly: equal or superior; fully interactive in Plotly; scriptable
Property diagrams (T-s, P-h, P-v)	Built-in Property Plot linked to EES fluid databases	CoolProp + Matplotlib; requires code but fully reproducible
Printing / formatted output	Formatted Equations window (auto-typeset); LaTeX export; MathType export; LaTeX/PDF full-file report; Report Window (Professional)	Matplotlib savefig; Jupyter nbconvert; Quarto; python-docx; no auto-typeset from solver equations
External interfaces	Excel, MATLAB, LabView, Python, serial port (macro commands; SendMessage v12.131+)	openpyxl/xlwings (Excel); MATLAB Engine; pyserial; REST APIs
Version control	Poor (binary .ees files)	Excellent (Git; plain-text source)
Machine learning / AI	None	scikit-learn, TensorFlow, PyTorch, etc.
Platform	Windows (Wine/Citrix for macOS/Linux)	Cross-platform (Windows, macOS, Linux)
License cost	$4,750 + annual AUS (dept 32-bit Pro Academic)	Free (open source)
Student license	$160/year	Free
Learning curve (beginner)	Low — minutes to first physics result	High — weeks of programming prerequisite
Career transferability	Limited	High
Long-term sustainability	Vendor-dependent; AUS renewal required	Community/open-source; permissive licenses
AI coding assistant	EES AI Librarian (Professional v12)	GitHub Copilot, Cursor, Claude, etc.

Textbook integration	Bundled libraries + Load Textbook (Nellis/Klein, Duffie/Beckman, etc.)	No bundled equivalent; open GitHub repos
Automated grading	EESyGrader (Academic)	No direct equivalent; nbgrader for Jupyter

19. Recommendations for USMMA

Stage 1 — EES for Introductory Thermal-Science Courses

Use EES Professional Academic for Thermodynamics, Heat Transfer, and Fluid Mechanics courses taught to students without prior programming experience. EES minimizes time-to-first-result, keeps cognitive load on physics rather than algorithms, and integrates with the Nellis & Klein materials. Budget the $4,750 32-bit Academic Professional departmental license. Critically: verify the AUS annual renewal requirement is funded each year (a lapse suspends the license under the post-June-2025 terms).

Stage 2 — Introduce Python in Parallel or Immediately After

Teach CoolProp (properties), NumPy/SciPy (solving and integration), Matplotlib (plots), and Quarto or Jupyter (reports) as early as the second thermal-science course or the first computational methods course. Use pint deliberately to teach unit discipline, acknowledging it does not replicate EES's automatic checking. Validate Python results against known EES answers to build student confidence.

Stage 3 — Python Only for Advanced and Research Courses

For combustion and kinetics (Cantera), data science, machine learning, large-scale simulation, optimization research, and any work requiring version control or web deployment, use Python exclusively.

Decision Trigger Criteria

- If F-Chart's AUS renewal costs or licensing terms tighten further, accelerate the transition to Python-only
- If a robust Python package providing automatic system-wide unit checking and declarative equation entry emerges, the primary pedagogical case for EES weakens substantially
- If entering students already have Python programming fluency, weight the curriculum toward Python from the start
- If specific accreditation requirements or courses depend on EES-integrated textbook libraries or the Make Distributable feature, retain EES for those courses regardless of other factors
- For experimental courses with laboratory hardware communicating via serial port, EES's serial port macro interface may offer a simpler integration path than Python + pyserial for students without programming backgrounds

Migration Validation Note

Where exact equivalence matters, note that both EES and CoolProp can call the same NIST REFPROP database via their respective REFPROP interfaces. This enables direct apples-to-apples validation of Python workflows against EES property calculations.

20. Caveats and Data Availability Notes

- Pricing and version facts for EES, REFPROP, and open-source libraries were confirmed from vendor and official project sources as of June 2026. Software pricing and licensing terms change without notice and should be reverified at time of purchase.
- EES built-in seawater properties: The Sea Water Property Library is listed in the EES help table of contents; the formulation used was not confirmed in this review.
- EES Mechanical Design Library: exists as an auto-loadable library; the exact function inventory (stress, fatigue, gear calculations) was not fully enumerated from available sources.
- EES electrical/semiconductor material-properties library: not confirmed in sources reviewed.
- Radiation view factors in Python ht library: not confirmed; EES's Heat Transfer Library does provide view factors.
- EES_REFPROP interface pricing and 64-bit Professional upgrade pricing: not confirmed from publicly available sources.
- Learning-curve time estimates are reasoned approximations based on tool design; they are not from controlled time studies.
- Commercial Python library options (PPDS/DIPPR, Multiflash, DWSim commercial tier, xlwings Pro): pricing was not confirmed in this review; these should be verified with vendors if needed.
- This report treats the USMMA Department of Mathematics and Computational Sciences worked-problems document (EES vs. Python, 10 problems) as background context only; all factual claims derive from independent vendor and project documentation.