PyRegen v1.0 User's Manual

Table of Contents

Preview

Quick Start

For Windows OS make sure you have Windows 8.1+, and then either:

- Download and run the executable from the website.
- Download the source code from Github, make sure you have all the requirements installed and run the ”RUN.py” file.

For Linux OS make sure you have Ubuntu 20.04+ LTS, Debian 11+ and then:

- Download the source code from Github, make sure you have all the reuirements installed and run ”python3 RUN.py” from the project folder opened in terminal.

The dependencies and requirements, as well as instructions on how to install them can be found on the Github repository, in the requirements.md file.

Licensing

PyRegen v1.0 is distributed under the GNU Affero General Public License v3.0 (AGPL 3.0). You can download and modify the software for free, under the condition that any derivative works are shared under the same license. You can find additional information regarding the license on the GNU website or on the PyRegen v1.0 Github repository.

Links

– Github repository: github.com/creatorandrew86/PyRegen-v1.0

– GNU AGPL v3.0: gnu.org/licenses/agpl-3.0.html

– CEA Documentation: readthedocs.io/rocketcea

– CoolProp Documentation: CoolProp.org/documentation.html

Any time a link or a website is referenceb below, you can access it directly from here.

1 Introduction

This document is the user's manual for the PyRegen software. As PyRegen v1.0 is the only version released at the moment, the user's manual will feature images and references to that version only. Future versions will come with an improved or rewritten user's manual, which you will be able to find on the website after any new release.

All versions of PyRegen are GUI based programs, making for very easy user input. The program output is also controlled from the GUI; however, it contains console output and files written by the program.

From a technical standpoint, PyRegen uses a quasi-1D flow assumption for the coolant and a corrected 1D heat equation for the wall heat transfer, with the NASA CEA software handling the gas-side analysis. PyRegen v1.0 only supports rectangular channels with constant or variable sizes. A more detailed technical explanation is provided in section 3.

Thus, PyRegen acts as a middle ground between purely empirical, simple analysis and CFD. Although less accurate than a 3D CFD solution, the very fast nature of the program, in both user input and execution time, makes iterating through possible designs much faster and more convenient.

PyRegen is written in Python 3.12, using additional libraries. If you download the source code from Github, you may have to install or upgrade the libraries mentioned in the requirements file. The program requires a minimum of Windows 8.1, or Ubuntu 20.04 LTS and Debian 11. If you download the executable file, the same operating system requirements apply, but the required libraries are contained within the file. The source code is 130 KB in size, and the executable file is around 76 MB.

This document will further guide the user on the program input and output, possible errors, and a case run.

2 User Input

The user input for PyRegen is divided into sections, both conceptually and physically. The GUI accepts inputs in 3 different tabs, with the following logic:

- Tab 1 - CEA Inputs / Chamber and Nozzle Geometry
- Tab 2 - Coolant Initial Properties
- Tab 3 - Cooling Channels Size

Section 2 is also divided following the same logic as the tabs, on individual groups of inputs, that are further detailed and clarified.

2.1 CEA Inputs

The **Fuel and Oxidizer** inputs have two variants, normal and custom input.

In the normal input, highlighted with red on the image below, the user selects a fuel/oxidizer from the given variants, and then, optionally, an entry temperature for the propellant. If there is no propellant temperature given, the program will assume the CEA default temperature.

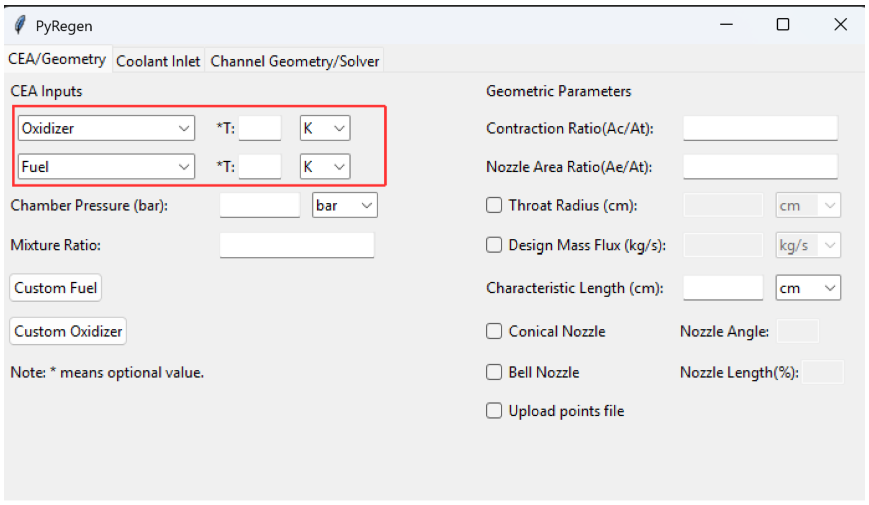


Figure 1: Propellant Selection

If custom fuel/oxidizer is chosen, the user will have to manually fill a text input, containing the definition of the propellant according to the CEA required input. An example is given for both the oxidizer and fuel custom inputs, but the full, detailed version of the input can be found on the NASA CEA documentation page (readthedocs.io) or the User's Manual for the software. The custom fuel input is presented in the images below.

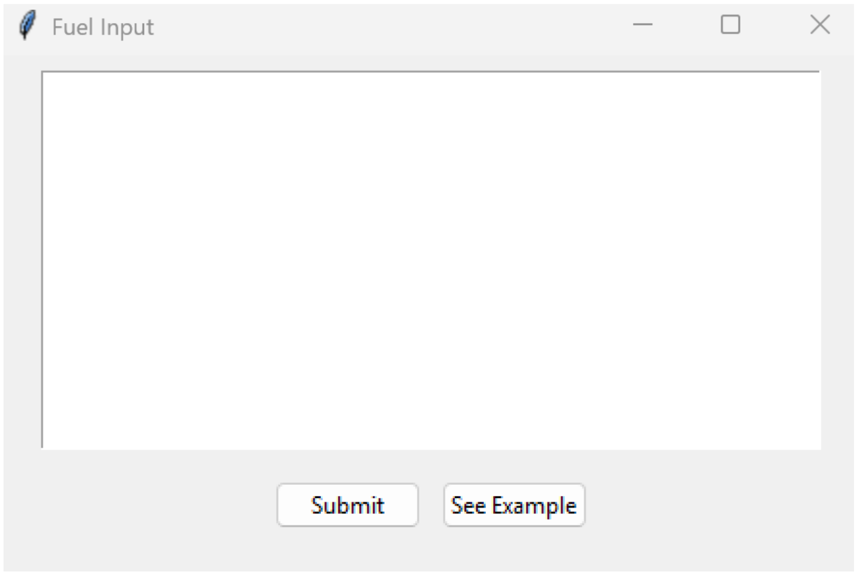
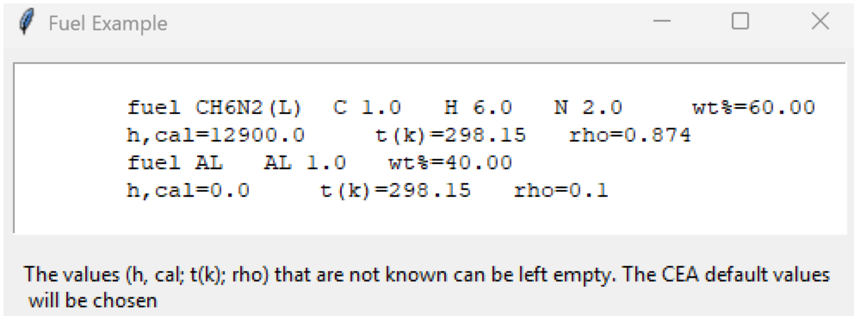


Figure 2: Custom Fuel Input



```
fuel CH6N2(L)  C 1.0   H 6.0   N 2.0     wt%=60.00
h,cal=12900.0      t(k)=298.15   rho=0.874
fuel AL   AL 1.0   wt%=40.00
h,cal=0.0      t(k)=298.15   rho=0.1
```

The values (h, cal; t(k); rho) that are not known can be left empty. The CEA default values will be chosen

Figure 3: Custom Fuel Example

Note: Custom fuel/oxidizer also support mixtures of propellants and user given enthalpy and density values.

The **Mixture Ratio** is a dimensionless value equivalent to the mass ratio of oxidizer to fuel (O/F) inside the chamber.

2.2 Chamber/Nozzle Geometry

The nozzle geometry is outlined in the interface by the title "Geometric Parameters" and is highlighted with blue on the image below.

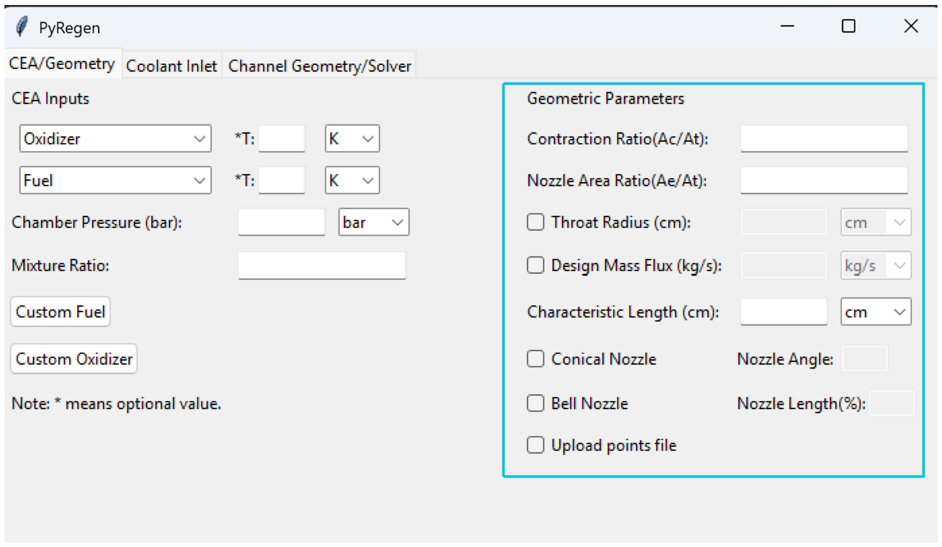


Figure 4: Chamber/Nozzle Geometry Inputs

The **Contraction Ratio** represents the ratio of the chamber area to the throat area.

The **Nozzle Area Ratio** represents the ratio of the nozzle exit area to the throat area, also known as expansion ratio.

PyRegen can calculate the throat radius using one of two input given by the user. If the throat radius is given, there is no other calculation needed. If the design mass flow is given, the throat density and velocity are calculated from the CEA output, and then the throat radius. One of the options must be selected.

The **Characteristic Length** is a property of the propellant mixture that reflects how fast the mixture can burn within the combustion chamber. Mathematically, it is equal to the ratio of the chamber volume to the throat area. You can easily find typical values for the characteristic length of many fuel combinations.

The nozzle geometry can be selected by the user from 3 options.

- Conical Nozzle: Most basic form of nozzle, used in many small, amateur level rocket engines. The nozzle angle is the angle the nozzle wall makes with the throat symmetry line

- Bell Nozzle: The most common type of nozzle among modern rocket engines, the Nozzle Length input represents the desired length of the nozzle as a percentage of the conical nozzle with the same expansion ratio.

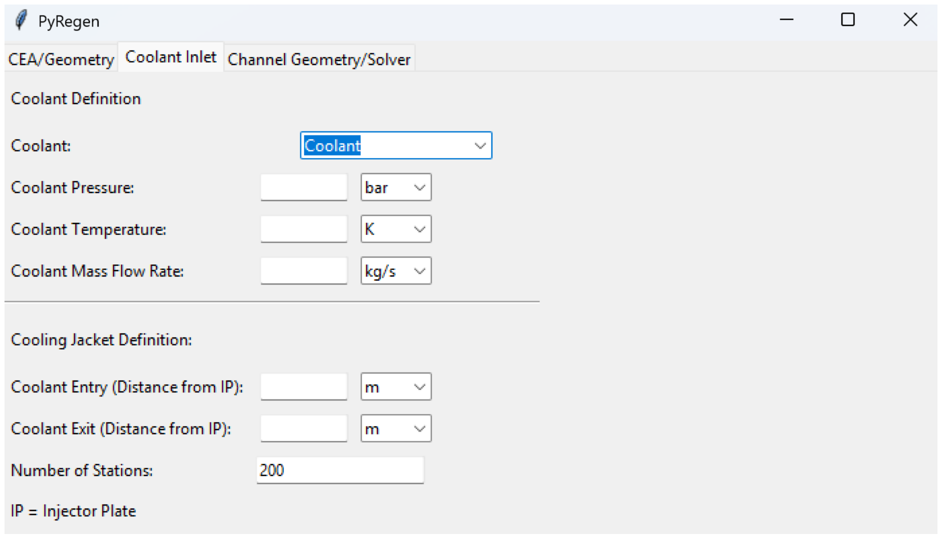
- The upload points file lets the user input a nozzle profile through the use of a text file. The format of the file is further detailed below.

PyRegen custom nozzle accepts a file in text or DAT format, with no header or title. The file should have as many lines as points on the nozzle, and 3 columns: x coordinate; y coordinate; area ratio at point x.

Note: As of version 1.0 the custom nozzle is not yet fully refined, so it is recommended to use with caution. Possible errors may arise because of an undetected fault in the code, it will be revised as of later versions.

2.3 Coolant Inlet Properties

Coolant initial properties as well as the range for the regenerative cooling analysis are described in Tab 2.


PyRegen

CEA/Geometry | Coolant Inlet | Channel Geometry/Solver

Coolant Definition

Coolant:	Coolant	
Coolant Pressure:		bar
Coolant Temperature:		K
Coolant Mass Flow Rate:		kg/s

Cooling Jacket Definition:

Coolant Entry (Distance from IP):		m
Coolant Exit (Distance from IP):		m
Number of Stations:	200	

IP = Injector Plate

Figure 5: Tab 2

The entries under Coolant Definition are all at the entry of the coolant.

The coolant entry and exit represent the boundaries of the cooling jacket, represented as the distance from the Injector Plate (IP) of the entry/exit boundaries.

The number of stations represents the number of points of analysis across the whole engine. A fraction of these points will be analyzed for regenerative cooling, depending on the length of cooling jacket specified by the user.

Note: If ”Upload Points” is used, the number of stations must be equal to the number of points in the file.

The specified number of stations are divided as follows:

- Chamber Section - 35%
- Convergent Section - 6%
- Throat Region - 15%
- Divergent Section - 44%

The percentages of the number given are rounded to the nearest integer, and the divergent section number is given by the remaining number of points.

2.4 Cooling Channel Geometry

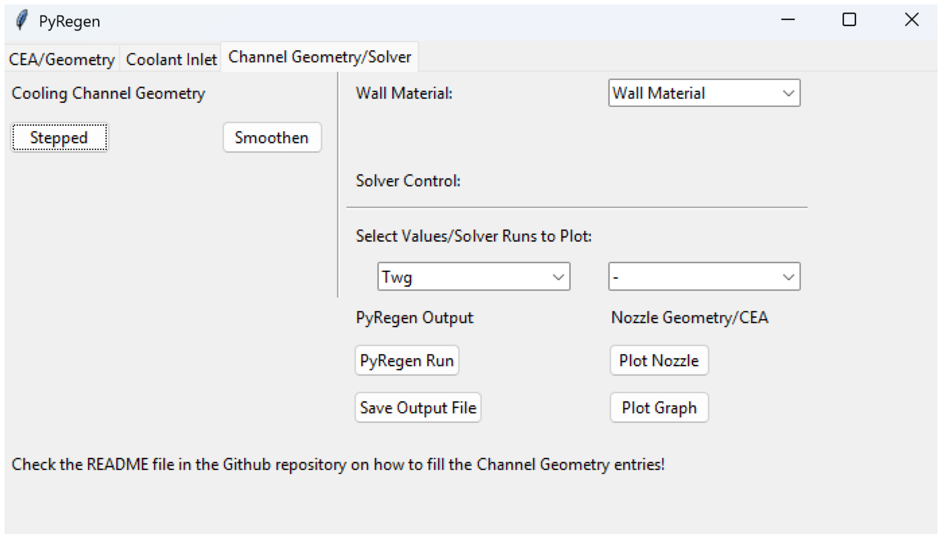
PyRegen offers two variants for cooling channel geometry: step geometry and smooth segment geometry, each being rectangular channels.

Step Geometry refers to a ”step” of constant width and height.

Smooth Segment Geometry refers to a ”smooth segment” with linearly/cubically varying width and height.

The cooling jacket can be divided in any number of steps or segments.

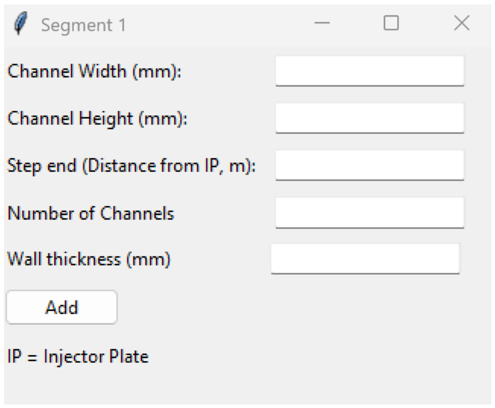
The geometry input is situated in Tab 3.



Check the README file in the Github repository on how to fill the Channel Geometry entries!

Figure 6: Tab 3

After selecting the channel geometry type, one of two windows will pop up.



Segment 1

Channel Width (mm):

Channel Height (mm):

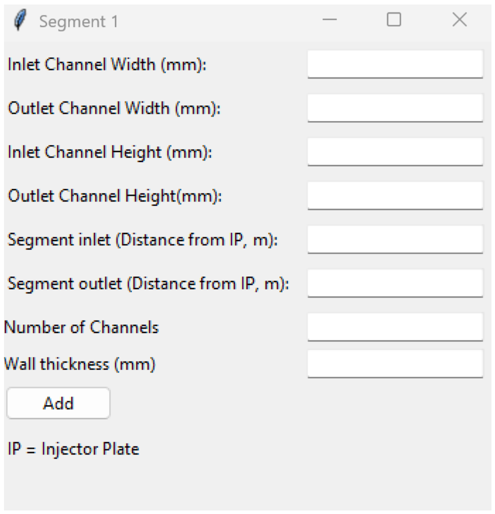
Step end (Distance from IP, m):

Number of Channels

Wall thickness (mm)

IP = Injector Plate

Figure 7: Stepped Channel Geometry



Segment 1

Inlet Channel Width (mm):

Outlet Channel Width (mm):

Inlet Channel Height (mm):

Outlet Channel Height(mm):

Segment inlet (Distance from IP, m):

Segment outlet (Distance from IP, m):

Number of Channels

Wall thickness (mm)

IP = Injector Plate

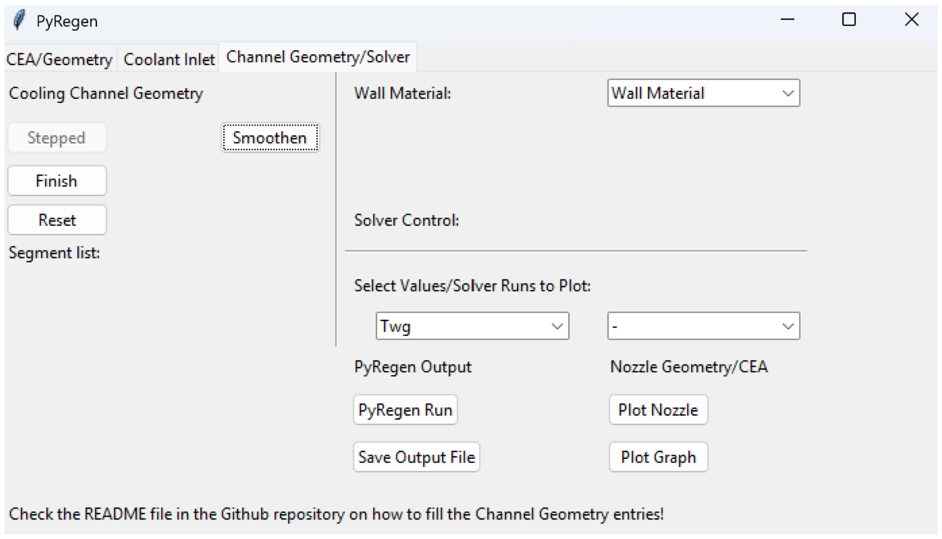
Figure 8: Smooth Segment Channel Geometry

The **Add** button in the windows hides the current step/segment window and creates a new step/segment. You can come back to each step and change values, before submitting from the **Finish** button in the tab (Figure 9).

In Figure 7, the **Step End** entry represents the outlet of the current step, as the distance from the Injector Plate, in meters. The "Step End" of the last step must coincide with the **Coolant Exit** from the Cooling Jacket Definition.

In Figure 8, the **Segment Inlet** and **Segment Outlet** entries represent the inlet and outlet of the current segment, as distances from the Injector Plate, in meters. The "Segment Inlet" of the first segment must coincide with the **Coolant Entry**, and the "Segment Outlet" of the last segment must coincide with the **Coolant Exit** from the Cooling Jacket Definition.

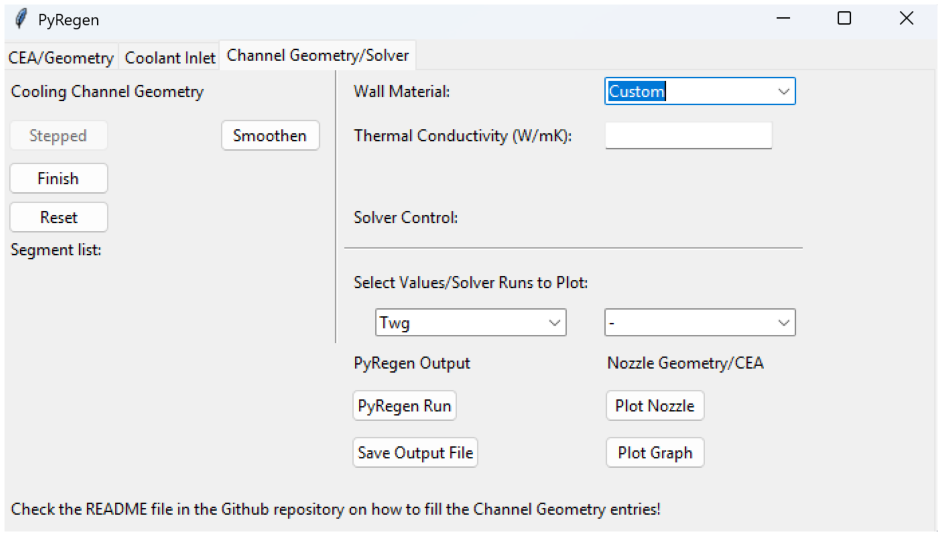
The potentially confusing part of this input is further explained through graphics on the Github repository of the software.



Check the README file in the Github repository on how to fill the Channel Geometry entries!

Figure 9: Tab 3

The wall material may be chosen from the given options, or a specific, constant value thermal conductivity (in SI units) may be given.



Wall Material: Custom

Thermal Conductivity (W/mK):

Twg

-

Check the README file in the Github repository on how to fill the Channel Geometry entries!

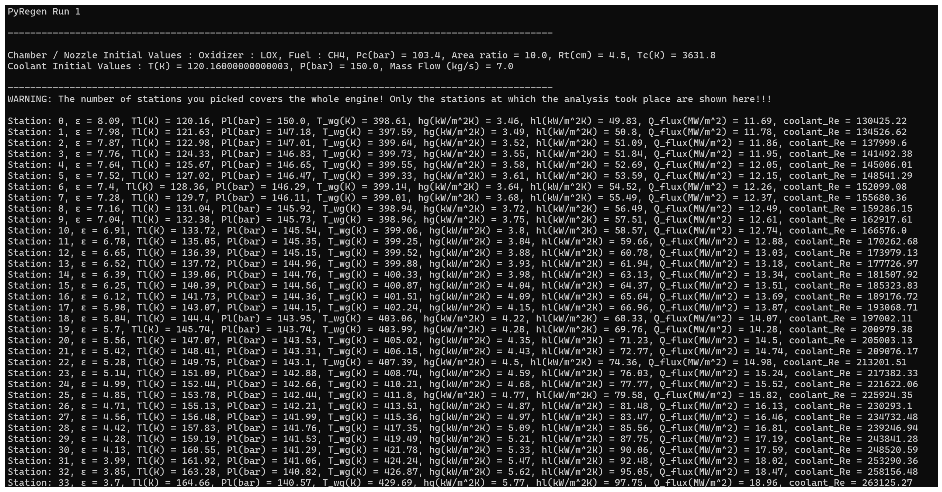
Figure 10: Tab 3

3 Program Output

There are several ways in which PyRegen outputs and displays the results. They are explained here in a chronological order from the moment the program runs.

3.1 Console Output

During the main function execution, it prints to console the main engine parameters and the results of the analysis at every point, as observed in Figure 11.



```
PyRegen Run 1

------------------------------------------------------------------------------------------------

Chamber / Nozzle Initial Values : Oxidizer : LOX, Fuel : CH4, Pc(bar) = 103.4, Area ratio = 10.0, Rt(cm) = 4.5, Tc(K) = 3631.8
Coolant Initial Values : T(K) = 120.16000000000003, P(bar) = 150.0, Mass Flow (kg/s) = 7.0

------------------------------------------------------------------------------------------------
WARNING: The number of stations you picked covers the whole engine! Only the stations at which the analysis took place are shown here!!!

Station: 0, ε = 8.09, Tl(K) = 120.16, Pl(bar) = 150.0, T_wg(K) = 398.61, hg(kW/m^2K) = 3.46, hl(kW/m^2K) = 49.83, Q_flux(MW/m^2) = 11.69, coolant_Re = 130425.22
Station: 1, ε = 7.98, Tl(K) = 121.63, Pl(bar) = 147.18, T_wg(K) = 397.59, hg(kW/m^2K) = 3.49, hl(kW/m^2K) = 50.8, Q_flux(MW/m^2) = 11.78, coolant_Re = 134526.62
Station: 2, ε = 7.87, Tl(K) = 122.98, Pl(bar) = 147.01, T_wg(K) = 399.64, hg(kW/m^2K) = 3.52, hl(kW/m^2K) = 51.09, Q_flux(MW/m^2) = 11.86, coolant_Re = 137999.6
Station: 3, ε = 7.76, Tl(K) = 124.33, Pl(bar) = 146.83, T_wg(K) = 399.73, hg(kW/m^2K) = 3.55, hl(kW/m^2K) = 51.84, Q_flux(MW/m^2) = 11.95, coolant_Re = 141492.38
Station: 4, ε = 7.64, Tl(K) = 125.67, Pl(bar) = 146.65, T_wg(K) = 399.55, hg(kW/m^2K) = 3.58, hl(kW/m^2K) = 52.69, Q_flux(MW/m^2) = 12.05, coolant_Re = 145006.01
Station: 5, ε = 7.52, Tl(K) = 127.02, Pl(bar) = 146.47, T_wg(K) = 399.33, hg(kW/m^2K) = 3.61, hl(kW/m^2K) = 53.59, Q_flux(MW/m^2) = 12.15, coolant_Re = 148541.29
Station: 6, ε = 7.4, Tl(K) = 128.36, Pl(bar) = 146.29, T_wg(K) = 399.14, hg(kW/m^2K) = 3.64, hl(kW/m^2K) = 54.52, Q_flux(MW/m^2) = 12.26, coolant_Re = 152099.08
Station: 7, ε = 7.28, Tl(K) = 129.7, Pl(bar) = 146.11, T_wg(K) = 399.01, hg(kW/m^2K) = 3.68, hl(kW/m^2K) = 55.49, Q_flux(MW/m^2) = 12.37, coolant_Re = 155680.36
Station: 8, ε = 7.16, Tl(K) = 131.04, Pl(bar) = 145.92, T_wg(K) = 398.94, hg(kW/m^2K) = 3.72, hl(kW/m^2K) = 56.49, Q_flux(MW/m^2) = 12.49, coolant_Re = 159286.15
Station: 9, ε = 7.04, Tl(K) = 132.38, Pl(bar) = 145.73, T_wg(K) = 398.96, hg(kW/m^2K) = 3.75, hl(kW/m^2K) = 57.51, Q_flux(MW/m^2) = 12.61, coolant_Re = 162917.61
Station: 10, ε = 6.91, Tl(K) = 133.72, Pl(bar) = 145.54, T_wg(K) = 399.06, hg(kW/m^2K) = 3.8, hl(kW/m^2K) = 58.57, Q_flux(MW/m^2) = 12.74, coolant_Re = 166576.0
Station: 11, ε = 6.78, Tl(K) = 135.05, Pl(bar) = 145.35, T_wg(K) = 399.25, hg(kW/m^2K) = 3.84, hl(kW/m^2K) = 59.66, Q_flux(MW/m^2) = 12.88, coolant_Re = 170262.68
Station: 12, ε = 6.65, Tl(K) = 136.39, Pl(bar) = 145.15, T_wg(K) = 399.52, hg(kW/m^2K) = 3.88, hl(kW/m^2K) = 60.78, Q_flux(MW/m^2) = 13.03, coolant_Re = 173979.13
Station: 13, ε = 6.52, Tl(K) = 137.72, Pl(bar) = 144.96, T_wg(K) = 399.88, hg(kW/m^2K) = 3.93, hl(kW/m^2K) = 61.94, Q_flux(MW/m^2) = 13.18, coolant_Re = 177726.97
Station: 14, ε = 6.39, Tl(K) = 139.06, Pl(bar) = 144.76, T_wg(K) = 400.33, hg(kW/m^2K) = 3.98, hl(kW/m^2K) = 63.13, Q_flux(MW/m^2) = 13.34, coolant_Re = 181507.92
Station: 15, ε = 6.25, Tl(K) = 140.39, Pl(bar) = 144.56, T_wg(K) = 400.87, hg(kW/m^2K) = 4.04, hl(kW/m^2K) = 64.37, Q_flux(MW/m^2) = 13.51, coolant_Re = 185323.83
Station: 16, ε = 6.12, Tl(K) = 141.73, Pl(bar) = 144.36, T_wg(K) = 401.51, hg(kW/m^2K) = 4.09, hl(kW/m^2K) = 65.64, Q_flux(MW/m^2) = 13.69, coolant_Re = 189176.72
Station: 17, ε = 5.98, Tl(K) = 143.07, Pl(bar) = 144.15, T_wg(K) = 402.24, hg(kW/m^2K) = 4.15, hl(kW/m^2K) = 66.96, Q_flux(MW/m^2) = 13.87, coolant_Re = 193068.71
Station: 18, ε = 5.84, Tl(K) = 144.4, Pl(bar) = 143.95, T_wg(K) = 403.06, hg(kW/m^2K) = 4.22, hl(kW/m^2K) = 68.33, Q_flux(MW/m^2) = 14.07, coolant_Re = 197002.11
Station: 19, ε = 5.7, Tl(K) = 145.74, Pl(bar) = 143.74, T_wg(K) = 403.99, hg(kW/m^2K) = 4.28, hl(kW/m^2K) = 69.76, Q_flux(MW/m^2) = 14.28, coolant_Re = 200979.38
Station: 20, ε = 5.56, Tl(K) = 147.07, Pl(bar) = 143.53, T_wg(K) = 405.02, hg(kW/m^2K) = 4.35, hl(kW/m^2K) = 71.23, Q_flux(MW/m^2) = 14.5, coolant_Re = 205003.13
Station: 21, ε = 5.42, Tl(K) = 148.41, Pl(bar) = 143.31, T_wg(K) = 406.15, hg(kW/m^2K) = 4.43, hl(kW/m^2K) = 72.77, Q_flux(MW/m^2) = 14.74, coolant_Re = 209076.17
Station: 22, ε = 5.28, Tl(K) = 149.75, Pl(bar) = 143.1, T_wg(K) = 407.39, hg(kW/m^2K) = 4.5, hl(kW/m^2K) = 74.36, Q_flux(MW/m^2) = 14.98, coolant_Re = 213201.51
Station: 23, ε = 5.14, Tl(K) = 151.09, Pl(bar) = 142.88, T_wg(K) = 408.74, hg(kW/m^2K) = 4.59, hl(kW/m^2K) = 76.03, Q_flux(MW/m^2) = 15.24, coolant_Re = 217382.33
Station: 24, ε = 4.99, Tl(K) = 152.44, Pl(bar) = 142.66, T_wg(K) = 410.21, hg(kW/m^2K) = 4.68, hl(kW/m^2K) = 77.77, Q_flux(MW/m^2) = 15.52, coolant_Re = 221622.06
Station: 25, ε = 4.85, Tl(K) = 153.78, Pl(bar) = 142.44, T_wg(K) = 411.8, hg(kW/m^2K) = 4.77, hl(kW/m^2K) = 79.58, Q_flux(MW/m^2) = 15.82, coolant_Re = 225924.35
Station: 26, ε = 4.71, Tl(K) = 155.13, Pl(bar) = 142.21, T_wg(K) = 413.51, hg(kW/m^2K) = 4.87, hl(kW/m^2K) = 81.48, Q_flux(MW/m^2) = 16.13, coolant_Re = 230293.1
Station: 27, ε = 4.56, Tl(K) = 156.48, Pl(bar) = 141.99, T_wg(K) = 415.36, hg(kW/m^2K) = 4.97, hl(kW/m^2K) = 83.47, Q_flux(MW/m^2) = 16.46, coolant_Re = 234732.48
Station: 28, ε = 4.42, Tl(K) = 157.83, Pl(bar) = 141.76, T_wg(K) = 417.35, hg(kW/m^2K) = 5.09, hl(kW/m^2K) = 85.56, Q_flux(MW/m^2) = 16.81, coolant_Re = 239246.94
Station: 29, ε = 4.28, Tl(K) = 159.19, Pl(bar) = 141.53, T_wg(K) = 419.49, hg(kW/m^2K) = 5.21, hl(kW/m^2K) = 87.75, Q_flux(MW/m^2) = 17.19, coolant_Re = 243841.28
Station: 30, ε = 4.13, Tl(K) = 160.55, Pl(bar) = 141.29, T_wg(K) = 421.78, hg(kW/m^2K) = 5.33, hl(kW/m^2K) = 90.06, Q_flux(MW/m^2) = 17.59, coolant_Re = 248520.59
Station: 31, ε = 3.99, Tl(K) = 161.92, Pl(bar) = 141.06, T_wg(K) = 424.24, hg(kW/m^2K) = 5.47, hl(kW/m^2K) = 92.48, Q_flux(MW/m^2) = 18.02, coolant_Re = 253290.36
Station: 32, ε = 3.85, Tl(K) = 163.28, Pl(bar) = 140.82, T_wg(K) = 426.87, hg(kW/m^2K) = 5.62, hl(kW/m^2K) = 95.05, Q_flux(MW/m^2) = 18.47, coolant_Re = 258156.48
Station: 33, ε = 3.7, Tl(K) = 164.66, Pl(bar) = 140.57, T_wg(K) = 429.69, hg(kW/m^2K) = 5.77, hl(kW/m^2K) = 97.75, Q_flux(MW/m^2) = 18.96, coolant_Re = 263125.27
```

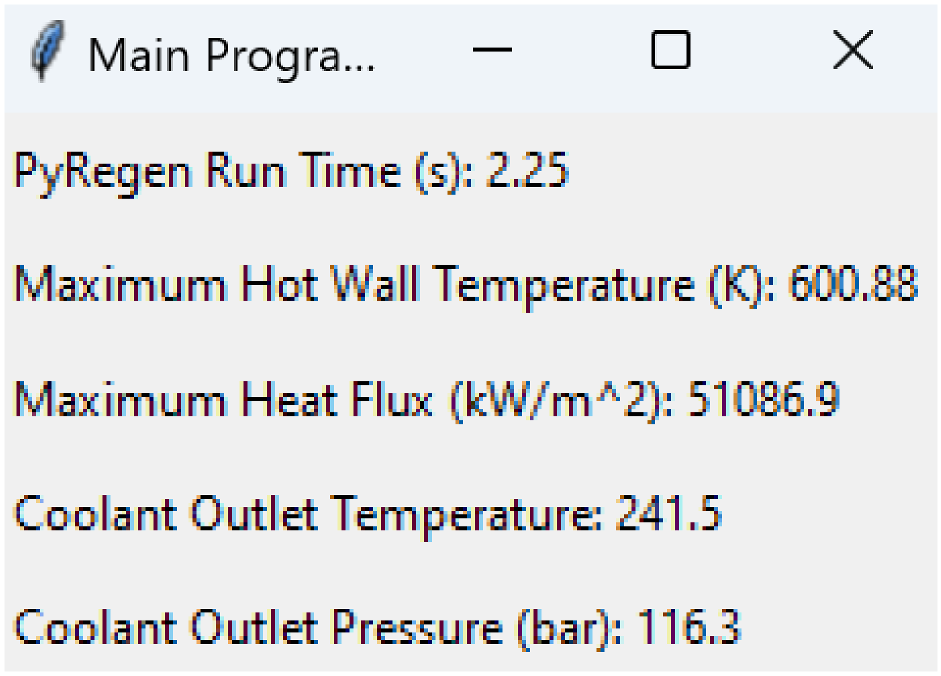
Figure 11: Console Output

Note: All of the output results are generated by the Case Run presented in Section 4.

Note: The console output presented in Figure 11 continues up to Station 110, but is only shown up to Station 33.

3.2 Main Output

The main output is generated right after the main function finishes running, and contains the general values of interest for the user.



```
PyRegen Run Time (s): 2.25

Maximum Hot Wall Temperature (K): 600.88

Maximum Heat Flux (kW/m^2): 51086.9

Coolant Outlet Temperature: 241.5

Coolant Outlet Pressure (bar): 116.3
```

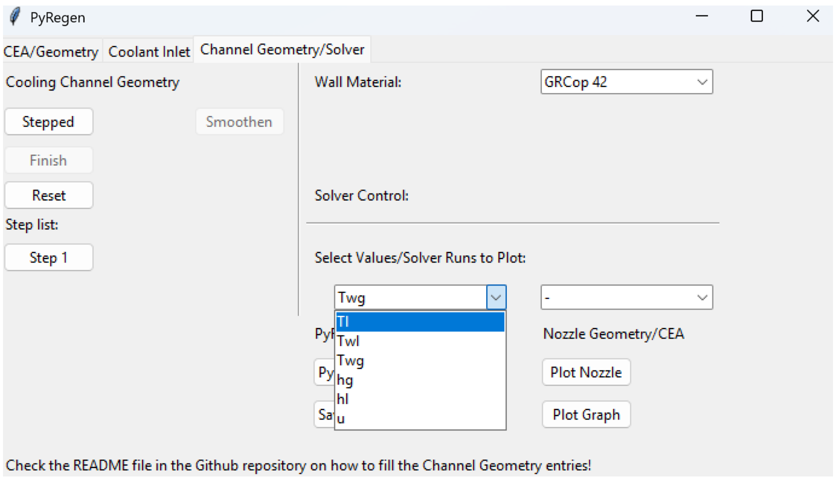
Figure 12: Main Output

3.3 Graphs

PyRegen offers a highly customizable graphing output, with either plotting results or the nozzle wall contour, generated by the software or uploaded by the user.

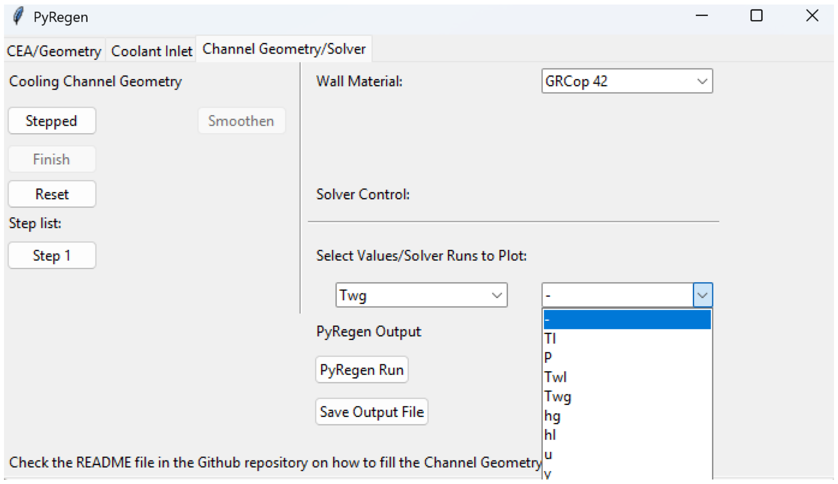
The **Plot Nozzle** button in Figure 10 simply plots the x and y coordinates of the nozzle wall contour on a separate window. The relevant documentation for the plot window (done with matplotlib) can be found on their website.

For the **Plot Graph** option in Tab 3, the user is given the following options in the drop down menu ”Select Values to Plot”:



Check the README file in the Github repository on how to fill the Channel Geometry entries!

Figure 13: Tab 3



Check the README file in the Github repository on how to fill the Channel Geometry

Figure 14: Tab 3

Figure 13 shows the Option 1 for plotting open in Tab 3. This lets the user select one value to plot against the x coordinate of the nozzle in the cooling jacket section.

Figure 14 shows the Option 2 for plotting open in Tab 3. If the user picks "-" as the selection (also the defalut) the plotting will not include any second value, just the first. If the user selects any other value, the plot will contain 2 y-axes with the first being option 1 and the second option 2, being plotted against the x coordinate of the nozzle.

If the solver ran more than once, the user can select "Run 1" (or any other run, depending on how many runs the solver performed) and the plot will contain the same value (option 1) for the current run and the specified run.

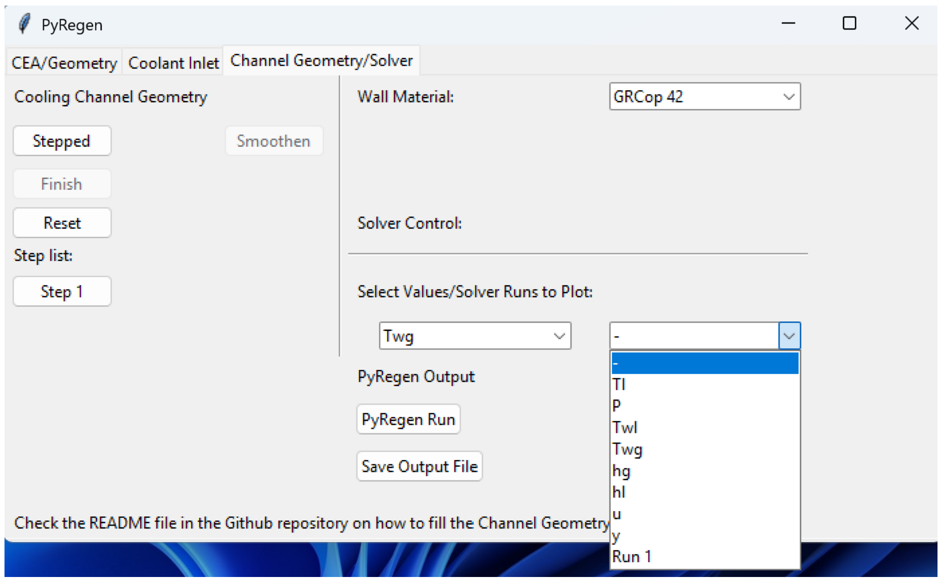


Figure 15: Tab 3

Below in Figure 16 you can see the resulting plot generated by choosing Twg (hot-wall temperature) as option 1, and P (coolant pressure) as option 2.

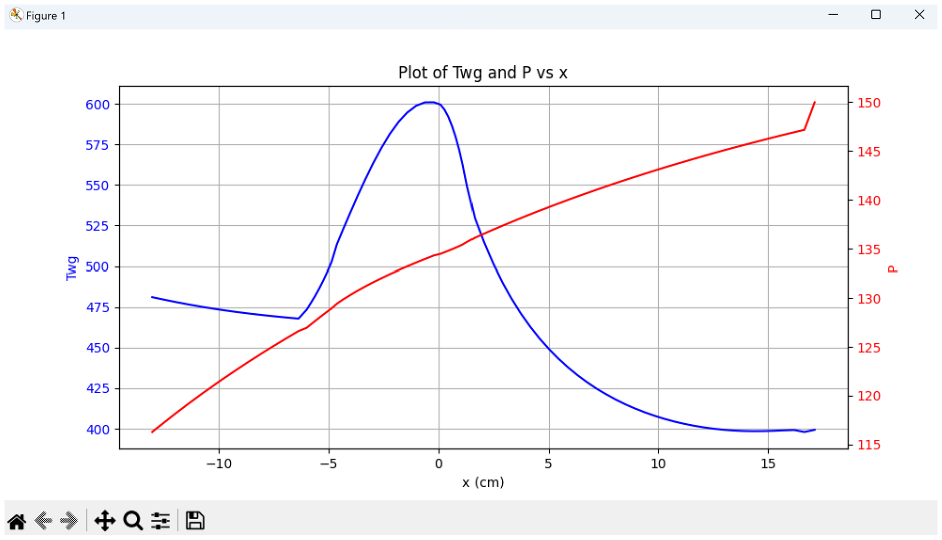


Figure 16: Plot Visualization

3.4 File Output

The full text file output contains a more detailed engine description, containing the user input and some CEA results, along with the regenerative cooling analysis results (from the Console Output).

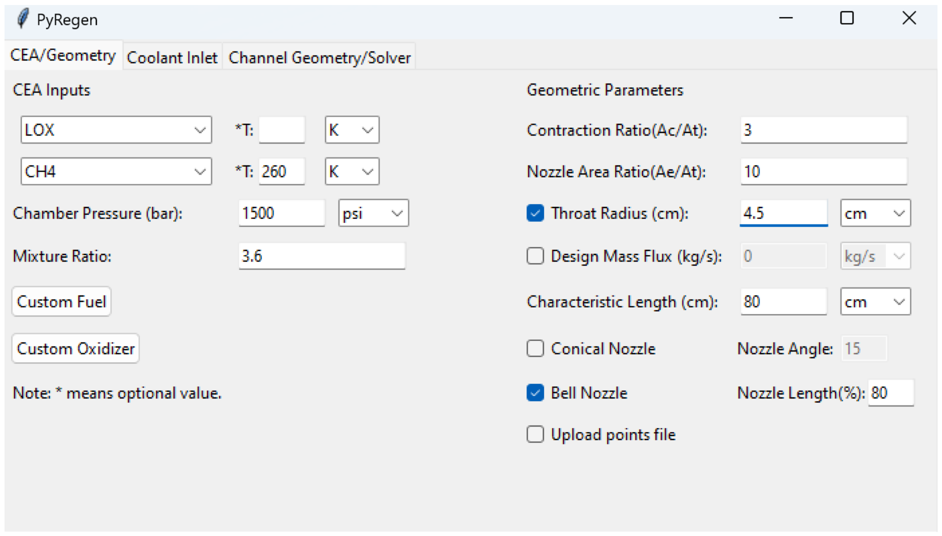
This type of output only contains one file. The file format is easily readable and understandable, so no further description is given here. You can find the example file in the Github repository, CASE RUN folder.

4 Case Run

This is an example run of the PyRegen v1.0 software. For a more detailed explanation, as well as the output file, you can check the Github repository of PyRegen v1.0, in the folder named "CASE RUN".

The figures in this file containing the output variants of the program are provided by this case run, with the only difference being the number of stations raised from 140 here to 200.

Note: The software accepts some inputs in different units (SI, Imperial..) even though a specific unit is written in the parantheses next to the name. The unit written there is a mistake, and is only related to version 1.0.



CEA Inputs

LOX *T: K

CH4 *T: 260 K

Chamber Pressure (bar): 1500 psi

Mixture Ratio: 3.6

Note: * means optional value.

Geometric Parameters

Contraction Ratio(Ac/At): 3

Nozzle Area Ratio(Ae/At): 10

[✓ Throat Radius (cm):] 4.5 cm

☐ Design Mass Flux (kg/s): 0 kg/s

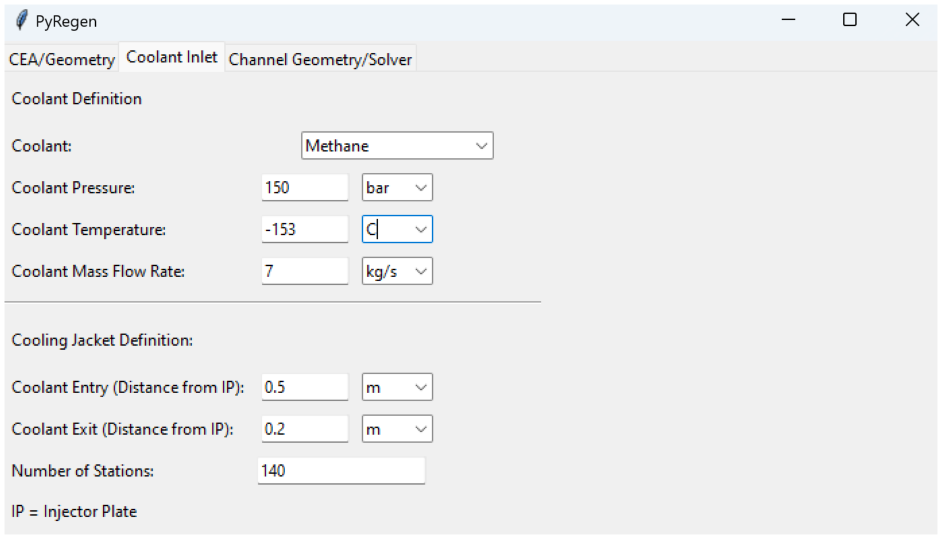
Characteristic Length (cm): 80 cm

☐ Conical Nozzle Nozzle Angle: 15

[✓ Bell Nozzle] Nozzle Length(%): 80

☐ Upload points file

Figure 17: Case Run Tab 1


PyRegen

CEA/Geometry | Coolant Inlet | Channel Geometry/Solver

Coolant Definition

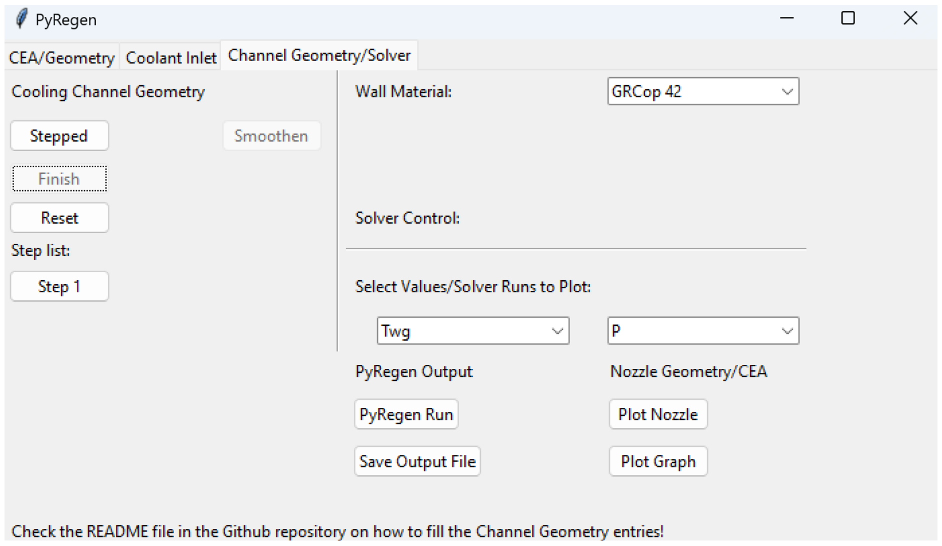
Coolant:	Methane	
Coolant Pressure:	150	bar
Coolant Temperature:	-153	C
Coolant Mass Flow Rate:	7	kg/s

Cooling Jacket Definition:

Coolant Entry (Distance from IP):	0.5	m
Coolant Exit (Distance from IP):	0.2	m
Number of Stations:	140	

IP = Injector Plate

Figure 18: Case Run Tab 2


PyRegen

CEA/Geometry | Coolant Inlet | Channel Geometry/Solver

Cooling Channel Geometry

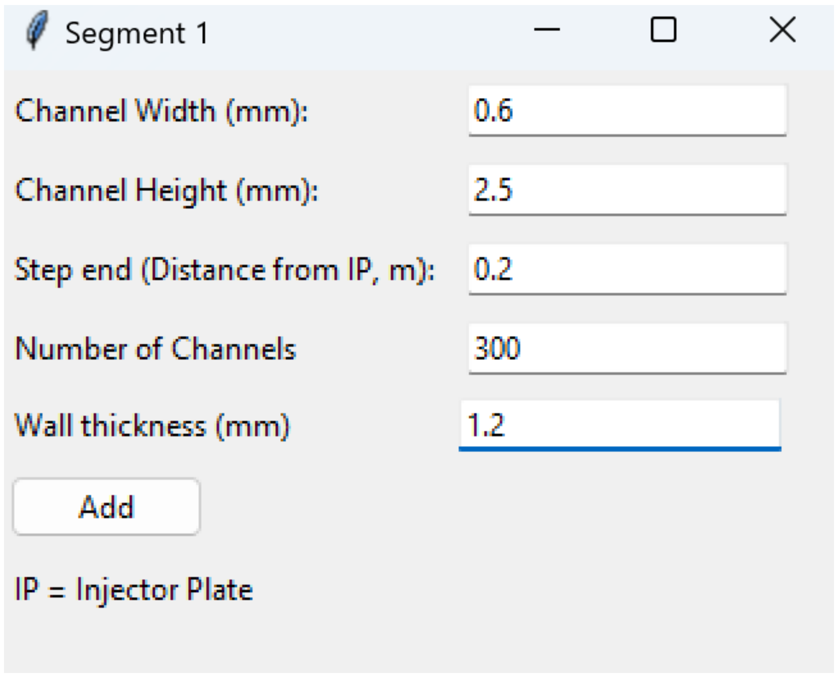
Wall Material: GRCop 42

Solver Control:

Select Values/Solver Runs to Plot: Twg | P

Check the README file in the Github repository on how to fill the Channel Geometry entries!

Figure 19: Case Run Tab 3

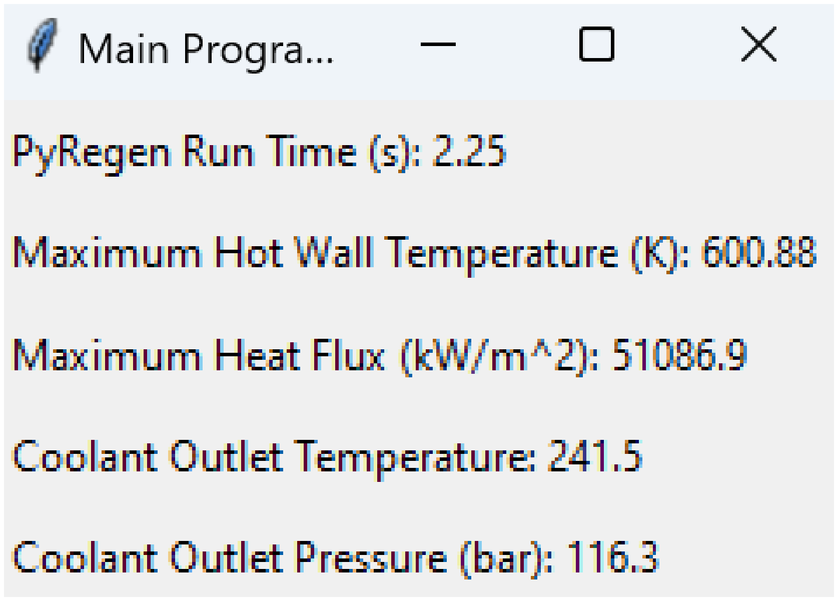


Segment 1

Channel Width (mm):	0.6
Channel Height (mm):	2.5
Step end (Distance from IP, m):	0.2
Number of Channels	300
Wall thickness (mm)	1.2

IP = Injector Plate

Figure 20: Case Run Channel Geometry



Main Progra...

```
PyRegen Run Time (s): 2.25

Maximum Hot Wall Temperature (K): 600.88

Maximum Heat Flux (kW/m^2): 51086.9

Coolant Outlet Temperature: 241.5

Coolant Outlet Pressure (bar): 116.3
```

Figure 21: Case Run Main Output

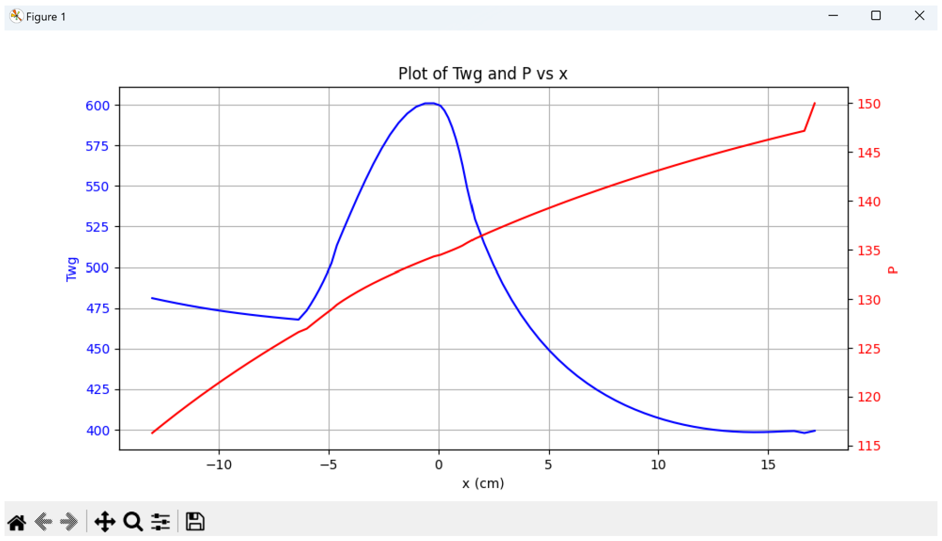


Figure 22: Case Run Plotted Values

5 Software Errors

PyRegen can send out errors in two forms:

- Console Errors
- Interface Errors

The Interface Errors originate only from an impossible input on some entries, or the lack of a required entry. The program returns if it encounters an error, and the user can continue after fixing the problem in the input section.

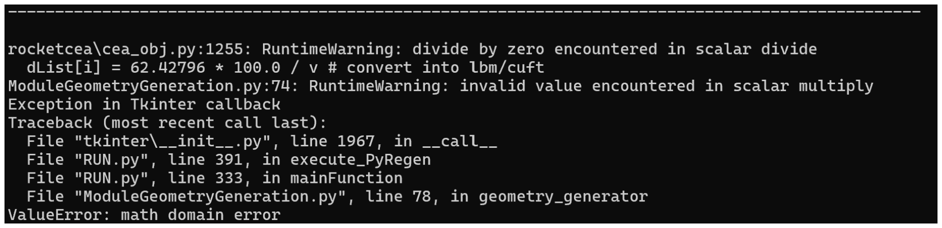
Console Errors can originate from:

- Imported Libraries(mainly CEA)
- Input Errors (detected by PyRegen)
- Python Errors

All of the aforementioned errors appear because of misinput, but they differ in the way they are sent out and the effects they have. The variants are going to be explained in further detail below.

5.1 Errors from Libraries

As the NASA CEA import is one of the most important outside parts of PyRegen, it often represents the source of input errors. They generally appear because a specified propellant temperature/enthalpy/density is out of range (for the given state) or any contradictions between the propellant definitions. A CEA return error generally looks like this:

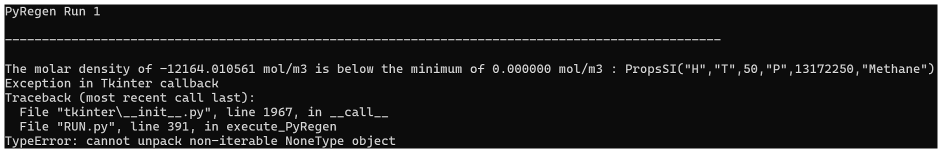


```
-----------------------------------------------------------------------------------------------

rocketcea\cea_obj.py:1255: RuntimeWarning: divide by zero encountered in scalar divide
  dList[i] = 62.42796 * 100.0 / v # convert into lbm/cuft
ModuleGeometryGeneration.py:74: RuntimeWarning: invalid value encountered in scalar multiply
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1967, in __call__
  File "RUN.py", line 391, in execute_PyRegen
  File "RUN.py", line 333, in mainFunction
  File "ModuleGeometryGeneration.py", line 78, in geometry_generator
ValueError: math domain error
```

Figure 23: CEA Console Error

To fix any error like this, verify your propellant inputs against the CEA documentation, where you will also find the input ranges for any propellant combination.

CoolProp is another important import in the functioning of the software, and it requires a very precise input to work properly, thus, a misinput here can result in a fatal error. Any ”out of range” inputs in the coolant temperature or pressure (See Tab 2) will result in an error looking like:



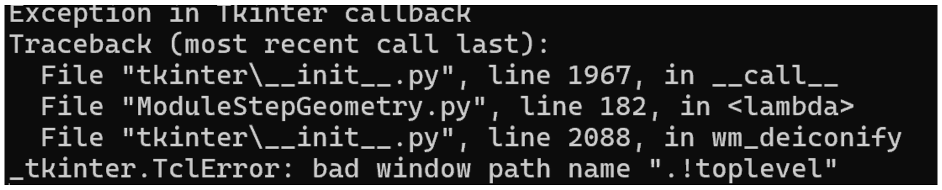
```
PyRegen Run 1

-----------------------------------------------------------------------------------------------

The molar density of -12164.010561 mol/m3 is below the minimum of 0.000000 mol/m3 : PropsSI("H","T",50,"P",13172250,"Methane")
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1967, in __call__
  File "RUN.py", line 391, in execute_PyRegen
TypeError: cannot unpack non-iterable NoneType object
```

Figure 24: CoolProp Console Error

You can find the input ranges for any coolant supported by CoolProp in the documentation on their website.

Speaking of the user interface, there are not many errors that can typically appear. The most common one is a bad window error when you try to reopen a step/segment window (SEE Chapter 2.4) after terminally closing it. The console will show the message from Figure 25, and you will not be able to reopen the window. However, this does not affect the running of the program.

If you want to change any variables in the affected step/segment, you will have to reset the channel geometry. This is a mistake on the side of PyRegen that will be revised in further releases.



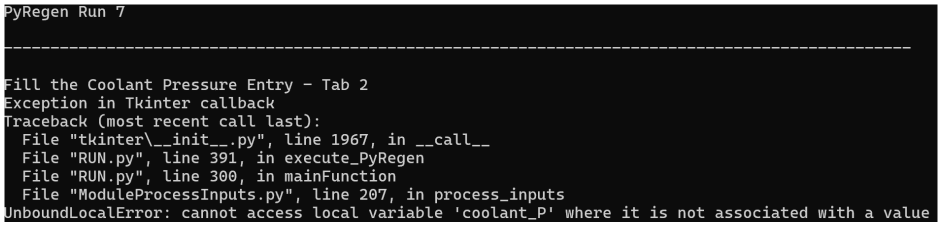
```
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1967, in __call__
  File "ModuleStepGeometry.py", line 182, in <lambda>
  File "tkinter\__init__.py", line 2088, in wm_deiconify
_tkinter.TclError: bad window path name ".!toplevel"
```

Figure 25: Tkinter Console Error

Other modules such as numpy and scipy may return ValueErrors, but these originate entirely from bad user input. If one of them appears, check your input makes sense, then retry. If the problem persists, feel free to contact us.

5.2 Input Errors

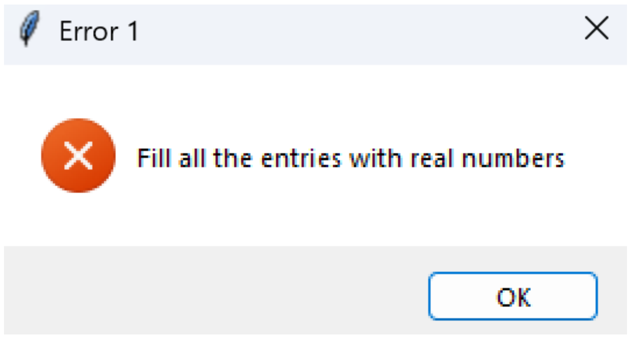
Besides the bad input errors, that are generally signaled by an error in the running of the software, PyRegen is equipped to catch missing input errors in the inputs processing phase. An input error originating from an empty coolant pressure entry can look like:



```
PyRegen Run 7

-----------------------------------------------------------------------------------------------

Fill the Coolant Pressure Entry - Tab 2
Exception in Tkinter callback
Traceback (most recent call last):
  File "tkinter\__init__.py", line 1967, in __call__
  File "RUN.py", line 391, in execute_PyRegen
  File "RUN.py", line 300, in mainFunction
  File "ModuleProcessInputs.py", line 207, in process_inputs
UnboundLocalError: cannot access local variable 'coolant_P' where it is not associated with a value
```

Figure 26: Empty Input Error

For the cooling channel geometry inputs, Tkinter messageboxes are used for displaying errors and returning, in case of empty or bad inputs. Errors may look like:


Error 1

Fill all the entries with real numbers

Figure 27: Tkinter Messagebox Error

For the moment, version 1.0 is not really good at catching errors in the inputs processing phase. Improved capabilities and more protection will be added in future releases.

5.3 Python Errors

This section is dedicated to the users who downloaded the source code. Python errors may arise because of:

- Outdated version
- Missing imports
- Wrong file names

For the first two causes, make sure you have all the requirements mentioned in the Github repository installed and up to date. Some of them may require addition to path, depending on your system.

If you rename the .py modules, or you do not put them in the same folder, as specified in the README file, in the Github repository, the program will not work due to a missing import error. You have to verify the naming scheme makes sense, or run the code from the executable file.

Other python-related errors are generally related to bad input, detailed in chapter 5.1.

6 Contact

Thank you for downloading and using PyRegen! If you have any questions regarding the software, or any message for us, feel free to contact us at pyRegen@gmail.com, or on the website, in the ”Support” menu.

We are still in the development process, so any feedback is highly appreciated.