Water-rocket-simulator code documentation

Manfred Gawlas

20.12.2023

Contents

1 Introduction **2**
1.1 Code Overview . . . 2
1.2 Graphical user interface . . . 2

2 How to use program **3**
2.1 Simulate . . . 3
2.2 Optimalization . . . 4

3 Variables **5**
3.1 Constants Declaration . . . 5
3.2 Simulation Output Data . . . 5
3.3 Optimization Output Data . . . 5
3.4 Export Data . . . 6

4 Functions **7**
4.1 Optimization Function . . . 7
4.2 Simulation for optimalization . . . 8
4.3 Simulation Function . . . 8
4.4 Plotting Functions . . . 8
4.5 Save to File Function . . . 9
4.6 Load from File Function . . . 9
4.7 Export to File Function . . . 9
4.8 Save and Close Function . . . 9
4.9 Save to Export Function . . . 10

5 Thinker GUI **11**

6 Simulation algorithm **13**
6.1 Input variables . . . 13
6.2 Structure . . . 13
6.3 Lauch of rod phase . . . 13
6.4 Liquid expulsion phase . . . 14
6.5 Mach 1 gas expulsion phase . . . 15
6.6 Sub Mach 1 gas expulsioin phase . . . 16

Chapter 1

Introduction

This document serves as simple documentation for the Water Rocket Simulation program, detailing the underlying algorithms of the simulation. The program is implemented in Python, utilizing the Tkinter and Matplotlib libraries. Its primary purpose is to offer a easy-to-use and efficient tool for the design of rocket engines powered by pressurized gas and liquid. Additionally, the optimization component of the program may contributes to a deeper understanding of such propulsion systems for user.

Big thanks to Sebastian Król for helping with the whole project. Without him Kononowicz's prediction would come true, there would be nothing(nie było by niczego).

1.1 Code Overview

The code consists of several parts, including:

- Variable declaration
- Main simulation function
- Optimalize function and simulation for optimalization loops
- Plot functions
- Save, load and export functions
- Main thinker body

1.2 Graphical user interface

Program has a graphical interface which consits of 4 main regions.

- Left plot - plot for data from single simulation
- Left input table - list of entries for data needed to preform single simulation
- Right plot - plot for data from optimalization
- Right input table - list of entries for data to optimalize around

Chapter 2

How to use program

In this part I will focus on explaining what certain inputs and outputs mean and how they change the way program works. I will not cover how to install aplication since it's already explained in README.md.

Output short names

- Ic - Total impuls
- tc - total time
- Ist - Specific impuls
- delta_v - delta of velocity

2.1 Simulate

Efficiency

Efficiency button allows you to input coefficients that are used when calculating thrust.

$$\text{Output\_Thrust} = \text{Efficiency} \cdot \text{Thrust}$$

Program allows you to choose custom efficiency for each seperate phase, that is for lauch phase, liquid expulsion phase and gas expulsion phase.

Custom gas

For program to use your custom gas, firstly you need to choose Custom in dropbox. Then insert information about your custom gas in "Specific heat ratio" and "Spec. gas constant" input fields.

Launch rod

By defeault program skips this stage. This happens when "Launch rod lenght" input is equal to 0. For program to include launch phase you simply have to change value of this input to value higher to 0.

Rod inside diamater input should be used when your launch rod is empty inside. When that's the case, insert internal diameter of the rod.

Pressure

Inserted value of pressure should be equal to absolute pressure inside the tank.

Dry mass

Dry mass is mass of structure of engine, that means mass of engine without the mass of propelant.

2.2 Optimalization

Choose variable to optimalize

In this dropbox you should choose name of value that you want your program to optimalize around. Importantly, when you choose to optimalize around something, you may leave coresponding input field at 0 or whatever value you like. It will not impact optimalization.

Optimalization range

Here input the range for values to optimalize around. These values NEED to be in same units as in input fields for corresponding variables.

Itterations

Number of simulations that you want the optimalization function to run throught. The higher the better output, however for big numbers it may take a while. Personally I recommend around 100-1000. If you choose 1000 it make take few secounds depending on your computer.

Weird plots

If you think that your plot is weird, it is most likely because of the fact that this simulations are done numerically. So for example total inpuls vs throat diameter can take many shapes, however for 1000 itterations it should be readable and give you general idea on what's happening.

Chapter 3

Variables

3.1 Constants Declaration

```
delta_t = 0.0001
P_atm = float(1 * 101325)
```

The `delta_t` variable represents the time step for the simulation. `P_atm` is the atmospheric pressure.

3.2 Simulation Output Data

The code declares several arrays to store simulation output data such as time, height, thrust, mass, temperature, pressure, and volumes of water and air.

```
array_time = []
array_h = []
array_Ft = []
array_mass = []
array_temperature = []
array_pressure = []
array_V_water = []
array_V_air = []
```

3.3 Optimization Output Data

Similar to simulation output data, there are arrays to store optimization output data.

```
array_opt_x = []
array_opt_Ic = []
array_opt_tc = []
array_opt_delta_v = []
array_opt_Ist = []
opt_variable_name = ""
```

3.4 Export Data

Variables to store information about the rocket engine for export.

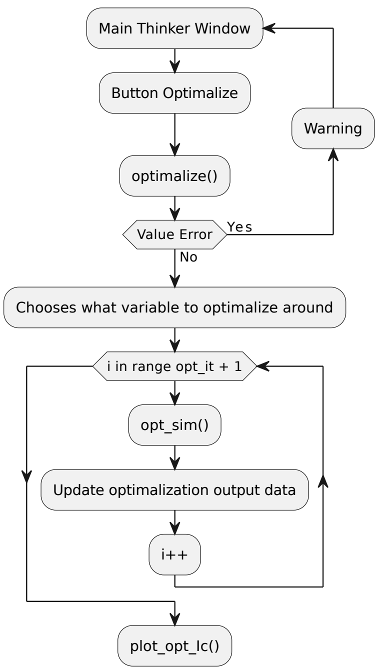
```
engine_name = ""
engine_diameter = 0
engine_length = 0
engine_dry_mass = 0
engine_full_mass = 0
engine_manufacturer = ""
```

Chapter 4

Functions

4.1 Optimization Function

```
def optimize():
    # Function to optimize water rocket design
    ...
```



The `optimize` function sets up and executes the optimization process based on user inputs and choosen variable.

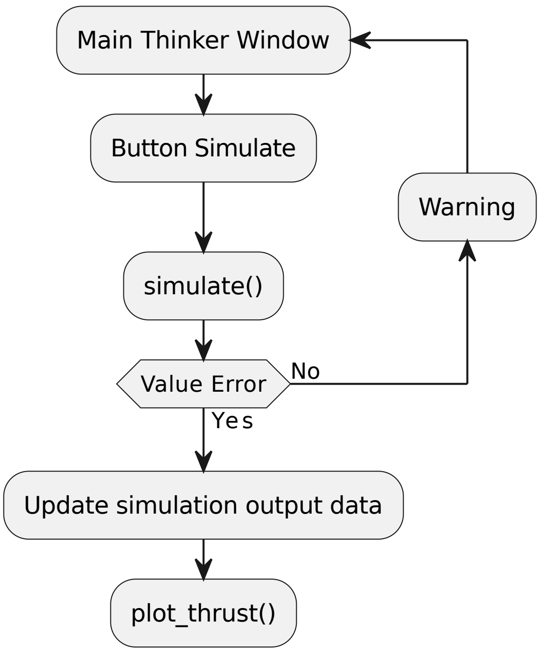
4.2 Simulation for optimalization

```
def opt_sim(k_const, Rs, water_density, P_ins, At, V_air, V_water, Roc_mass, T,
rod_length, rod_inside_diameter):
    # Function to simulate water rocket flight for optimization
    ...
    return [Ic, total_time, Ic / (mass_propellant * 9.81), delta_v]
```

This function performs the numerical simulation of the water rocket's flight for optimization.

4.3 Simulation Function

```
def simulate():
    # Function to simulate certain rocket motor
    ...
```



The `simulate` function executes simulation for a single data setup. It's responsible for left plot.

4.4 Plotting Functions

Functions to plot optimization results.

```
def plot_opt_Ic():
    ...
```

```
def plot_opt_tc():
    ...

def plot_opt_Ist():
    ...

def plot_opt_delta_v():
    ...
```

These functions use the Matplotlib library to plot various optimization results. Optimalize and simulate functions have their coresponding plot functions.

4.5 Save to File Function

```
def save_to_file(entry_text):
    # Function to save simulation parameters to a file
    ...
```

4.6 Load from File Function

```
def load_from_file(entry_text):
    # Function to load simulation parameters from a file
    ...
```

4.7 Export to File Function

```
def export_to_file():
    # Function to export simulation data to a .eng format
    ...
```

This function consits mainly of code for making additional window for export data that user needs to input.

4.8 Save and Close Function

```
def save_and_close(entry_export_name, entry_export_diameter, entry_export_lenght, entry
    # Function to save export parameters and close the export window
    ...
```

4.9 Save to Export Function

```python
def save_to_export():
    # Function to save exported data to a file
    ...
```

Chapter 5

Thinker GUI

This part of code creates a Tkinter-based GUI application for a Water Rocket Simulator. Here's a summarised version of code:

```
# Create the main Tkinter window
root = tk.Tk()
root.title("Water Rocket Simulator")

# Menu Bar
menubar = tk.Menu(root)
file_menu = tk.Menu(menubar, tearoff=0)
file_menu.add_command(label="Save", command=lambda: save_to_file(entry_0))
file_menu.add_command(label="Load", command=lambda: load_from_file(entry_0))
file_menu.add_command(label="Export as .eng", command=lambda: export_to_file())
menubar.add_cascade(label="File", menu=file_menu)
root.config(menu=menubar)

# Set the window size to full screen
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
root.geometry(f"{screen_width}x{screen_height}")

# ... (Other setup code)

# LEFT
frame_simulate = ttk.Frame(root, padding="10")
# ... (Left frame setup)

# LEFT Plot
frame_plot_left = ttk.Frame(root, padding="10")
# ... (Matplotlib setup for left plot)

# RIGHT
frame_simulate_r = ttk.Frame(root, padding="10")
# ... (Right frame setup)
```

```python
# RIGHT Plot
frame_plot_right = ttk.Frame(root, padding="10")
# ... (Matplotlib setup for right plot)

# Run the Tkinter event loop
root.mainloop()
```

Note: The code involves extensive GUI setup, including input fields, buttons, and plots using Tkinter and Matplotlib. It also incorporates error handling and dynamic updates based on user inputs.

Chapter 6

Simulation algorithm

6.1 Input variables

Although the set of inputs for the user is easy to understand, this function takes another set of values as input, which can be calculated from user input. These are the same as below:

```
def opt_sim(k_const, Rs, water_density, P_ins, At, V_air, V_water, Roc_mass, T,
rod_length, rod_inside_diameter):
```

6.2 Structure

The simulation consists of 5 while loops. The first two are responsible for the launch of the rod period, the third concerns the liquid expulsion phase, and the last two are responsible for the gas period.

6.3 Lauch of rod phase

If rod inside diameter is equal to 0 then we treat it as foregin volume inside chamber. Model is basicly an adibatic expansion of gas into volume left by pushed out rod. We calculate thrust simply as:

$$F = P_{ins} \cdot A_t$$

Rest is basicly numerical analysis for Δt.

$$v_e + = a \cdot \Delta t \qquad a = \frac{F}{m_r oc}$$

$$\Delta V = A_t v_e \Delta t$$

Now we can calculate new volume of gas:

$$V_{new} = Vold + \Delta V$$

Using equation for adiabatic expansion:

$$pV^k = const$$

$$T = \frac{pV}{mRs}$$

This concludes calculations for single time frame. All of this is in while loop, which can be expressed as:

```
while( s < rod_lenght)
        s += s + ve*delta_t + a*delta_t*delta_t*0.5
        ...
```

If rod inside diameter is higher then 0 the computational model is even simple, we assume that

$$p = const, V = const, T = const$$

$$F = P_{ins} \cdot A_t$$

6.4 Liquid expulsion phase

During this phase we will again assume that thrust of motor is caused by adiabatic expansion of gas. For thrust we will use following equation:

$$F = \dot{m} v_e$$

Since velocities are relaltivly low we will use Bernoulies law which can be writen as:

$$p_1 + \frac{\rho v_1^2}{2} = p_2 + \frac{\rho v_2^2}{2}$$

Since area of liquid facing gas inside is much greater then A_t we assume that $v_1 = 0$. That gives negligible error. From that we have:

$$p_{ins} = p_{atm} + \frac{\rho v_e^2}{2}$$

$$v_e = \sqrt{2\frac{\Delta p}{\rho}}$$

$$\dot{m} = \rho \cdot A_t \cdot v_e$$

That gives:

$$F = 2A_t \Delta p$$

Now we calculate new values of V, P, T of gas.

$$\Delta V = A_t v_e \Delta t$$

$$pV^k = const$$

$$T = \frac{pV}{mRs}$$

Algorithm in steps

1. $F_t = ...$
2. $v_e = ...$
3. $\Delta V = ...$
4. $T = ...$
5. $m_{roc}- = \Delta V \cdot \rho_{wat}$
6. $V_{air} = ... \qquad V_{wat} = ...$
7. $P_{ins} = ...$

6.5 Mach 1 gas expulsion phase

We will treat this phase as a series of adiabatic expansions of gas. To calculate thurust we will use well known formula which I won't explain here.

$$F = \dot{m} v_e + A_t(p_{atm} - p_{ins})$$

Since this phase is for Mach 1 we can calculate v_e in following manner:

$$v_e = M_e \cdot a_e = a_e = \sqrt{kRsT_t}$$

Now we use very cool formulas for Mach 1

$$T_t = \frac{2}{k+1} T^*$$

$$\rho_t = \left(\frac{2}{k+1}\right)^{\frac{1}{k-1}} \rho^*$$

$$\dot{m} = p^* A_t \left(\frac{2}{k+1}\right)^{\frac{k+1}{k-1}} \left(\frac{k}{RsT^*}\right)$$

Now for numerical solution:

$$\Delta m = \dot{m} \cdot \Delta t$$

$$\rho^*_{new} = \frac{m - \Delta m}{V}$$

$$p = \frac{mRsT}{V}$$

Now we rearange the adiabatic equation:

$$p_1 v_1^k = p_2 v_2^k$$

By substituting ideal gas equation and we get following equation for T:

$$T = \frac{p}{Rs} \left(\frac{V}{m_1}\right)^k \left(\frac{V}{m_1 - \Delta m}\right)^{1-k}$$

Now we compute it numericaly till we have Mach 1 on exit.

Algorithm in steps

1. $T_t = ...$
2. $\dot{m} = ...$
3. $v_e = ...$
4. $P_t = ...$ $\quad F_t = ...$
5. $\Delta m = \dot{m} \cdot \Delta t$ $\quad m_{roc} = ...$
6. $T = ...$ $\quad P_{ins} = ...$

6.6 Sub Mach 1 gas expulsioin phase

Calculations are similar but this time we have to calculate M_e. We assume $p_t = p_{atm}$.

$$M_e = \sqrt{\frac{2}{k-1}\left(\left(\frac{p_{ins}}{p_t}\right)^{\frac{k-1}{k}} - 1\right)}$$

$$v_e = M_e \cdot a_e$$

Rest are just analogical equations for $M_e < 1$.

Algorithm in steps

1. $M_e = ...$
2. $T_t = ...$
3. $v_e = ...$
4. $\rho^* = ...$ $\quad \rho_t = ...$
5. $\dot{m} = ...$
6. $F_t = ...$
7. $\Delta m = \dot{m} \cdot \Delta t$ $\quad m_{roc} = ...$
8. $T = ...$ $\quad P_{ins} = ...$

Now with the understanding of underlaying physics reader should have no problem understanding numerical computations behind simulation loops.

Bibliography

[1] Michael de Podesta (2006) A guide to building and understanding the physics of Water Rockets, NPL

[2] Sebastian Król (2023) Rocketry formulas and derivations, PWr In Space