Carleton Hybrid Rocket Motor Simulator

Adrian Comisso

March 29, 2025

1 Abstract

This document outlines the usage and underlying theory of the Carleton Hybrid Rocket Motor Simulator (CHRMS). Developed as part of the CU InSpace rocketry team's hybrid rocket motor project, CHRMS models the performance of liquid-oxidizer hybrid engines. The simulator incorporates a variety of models to represent critical phases of the engine's operation, including the tank, combustion, regression, chamber, and nozzle phases. It offers the ability to simulate both cold flow and hot fire scenarios and provides new simulation models to deal with two-phase injector flow. Inspired by Robert A. Nickel's Hybrid Rocket Analysis Program (HRAP) from the University of Tennessee [1], CHRMS diverges with its unique tank and injector models. A cloned version of HRAP has been integrated to validate and refine simulations. The simulator currently only features configurations for CO2 and N2O.

For readers seeking only the instructions on how to use the simulator, see section 7

Contents

2 Nomenclature

Report Variable	Sim Varible	Name
t_{max}	t_max	Maximum Simulation Time
Δt	dt	Time Step Length
Q	Q	Heat
U	U	Internal Energy
Z	Z	Compressibility Factor
h	h	Specific Enthalpy
γ	gamma	Specific Heat Ratio
P_{atm}	P_atm	Ambient Pressure
T_{atm}	T_atm	Ambient Temperature
ρ_{atm}	rho_atm	Ambient Air Density
R	R	Ideal Gas Constant
a_g	a_g	Acceleration due to Gravity
V_{tank}	V_tank	Oxidizer Tank Volume
T_{tank}	T_tank	Initial Oxidizer Tank Temperature
$m_{ox,total}$	m_ox_total	Oxidizer Mass
$m_{ox,liquid}$	m_ox_l	Liquid Oxidizer Mass
$m_{ox,vapor}$	m_ox_v	Vapor Oxidizer Mass
d_{vent}	d_vent	Vent Diameter
$C_{d,vent}$	Cd_vent	Vent Coefficient of Discharge
N_{inj}	N_inj	Number of Injector Holes
d_{inj}	d_inj	Injector Hole Diameter
$C_{d,inj}$	Cd_inj	Injector Coefficient of Discharge
ΔP_{inj}	delta_P_inj	Pressure Drop Across Injector
$V_{chamber,empty}$	V_cmbr_empty	Empty Combustion Chamber Volume
$V_{chamber,grn}$	V_cmbr_empty	Combustion Chamber Volume With Grain
$P_{chamber}$	P_cmbr	Initial Combustion Chamber Pressure
$d_{ID,grain}$	id_grn	Grain Inner Diameter
$d_{ID,out}$	od_grn	Grain Outer Diameter
l_{grain}	l_grn	Grain Length
C^*_{eff}	c_star_eff	Combustion Efficiency
ρ_{fuel}	rho_fuel	Fuel Grain Density
OF_{const}	const_OF	Constant Oxidizer-Fuel Ratio
OF	OF	Current Oxidizer-Fuel Ratio
a	a	Regression Coefficient (Shifting OF)
n	n	Regression Coefficient (Shifting OF)
m	m	Regression Coefficient (Shifting OF)
α_{nozzle}	alpha_noz	Nozzle Half-Angle
$d_{nozzle,throat}$	d_noz_throat	Nozzle Throat Diameter
ER	exp_ratio_noz	Nozzle Expansion Ratio
η_{nozzle}	eff_noz	Nozzle Efficiency
$\lambda_{divergence}$	lambda	Nozzle Divergence Factor
$C_{d,nozzle}$	Cd_noz	Nozzle Coefficient of Discharge
$m_{dry,motor}$	m_dry_motor	Motor Dry Mass (without Fuel/Oxidizer)

$CG_{dry,motor}$	`cg_dry_motor`	**Dry Motor Center of Gravity from Nozzle**
$l_{tank,bottom}$	`l_tank_bottom`	**Distance from Nozzle to Tank Bottom**
d_{tank}	`d_tank`	**Tank Inner Diameter**
$l_{grain,bottom}$	`l_grn_bottom`	**Distance from Nozzle to Bottom of Fuel Grain**
LC		**Launch Canada Competition**
$IREC$		**International Rocket engineering Competition**
$HRAP$		**Hybrid Rocket Analysis Program**
$COTS$		**Commercial off the Shelf**
$SRAD$		**Student Researched and Designed**

3 Introduction

The CU InSpace Rocketry Design Team is a student-led group within Carleton University's Faculty of Engineering and Design, based in Ottawa, Canada. The team comprises undergraduate students who design, build, test, and launch high-powered rockets annually. These launches take place at the International Rocket Engineering Competition (IREC) and the Launch Canada Competition (LC). Historically, the team has utilized commercial off-the-shelf (COTS) motors, sourced from suppliers like Cesaroni Technologies Inc. and AeroTech Rocketry [2][3].

In recent years, the team has focused on developing an in-house Hybrid Rocket Motor (HRM). Unlike conventional solid or liquid rocket motors, a hybrid rocket motor uses a combination of liquid and solid propellants, with the oxidizer typically in liquid form and the fuel as a solid grain. The process of modeling the motor from the oxidizer tank to the nozzle can be quite complex. Several 3rd party hybrid rocket simulators exist. The most popular is the open source HRAP project originally created by Robert Nickel of the University of Tennessee Rocket Team [1]. This simulator lacks a couple of key features the team is looking for, such as a two-phase injector flow model.

The Carleton Hybrid Rocket Motor Simulator (CHRMS) addresses these limitations by implementing advanced physical models for two-phase flow, improved tank thermodynamics, and enhanced nozzle calculations. The simulator provides multiple options for each component, allowing users to select appropriate models based on their specific motor configuration and available empirical data. This document details the theoretical foundations, computational implementation, and practical usage of the CHRMS tool.

4 Support Material

The following key sources were instrumental in developing this simulator. While this document aims to summarize the most relevant information, reading the original papers in full

is highly recommended. For convenience, all referenced papers are included in ..../Reference Papers/PDF Paper Catalogue.txt within the installation directory.

Although the simulator was developed from scratch, it draws significant structural inspiration from Robert A. Nickel's Hybrid Rocket Analysis Program (HRAP) [1]. Nickel's *HRAP Theory of Operation* [4] provides an in-depth overview of HRAP's functionality and valuable insights into nitrous oxide behavior simulation.

HRAP itself is based on three foundational papers by Rick Newlands of Aspire Space, essential for understanding hybrid rocket motor modeling:

- *The Physics of Nitrous Oxide* [5]– Explains the self-pressurizing behavior of nitrous oxide.
- *Modeling the Nitrous Run Tank Emptying* [6]– Details the mathematical model used to simulate oxidizer tank dynamics.
- *Introduction to Hybrid Design* [7] – Covers combustion chamber modeling and fuel regression dynamics.

The new simulator model is based on the equilibrium section of *Review and Evaluation of Models for Self-Pressurizing Propellant Tank Dynamics* [8]. This model will be referred to as the *Internal Energy Model* going forward.

A primary motivation for developing this simulator was to assess whether the injector would encounter two-phase flow, which can cause mass flow rate fluctuations. *An Investigation of Injectors for Use With High Vapor Pressure Propellants With Applications to Hybrid Rockets* [9] provides a comprehensive analysis of injector behavior, including two-phase flow, the Vena Contracta effect, inefficiencies, and relevant modeling techniques.

5 Theory

5.1 Hybrid Rocket Design

The simulator models the operation of a hybrid rocket motor. While hybrid rocket motors can vary in design, this model simplifies the motor into the configuration shown in Figure 1

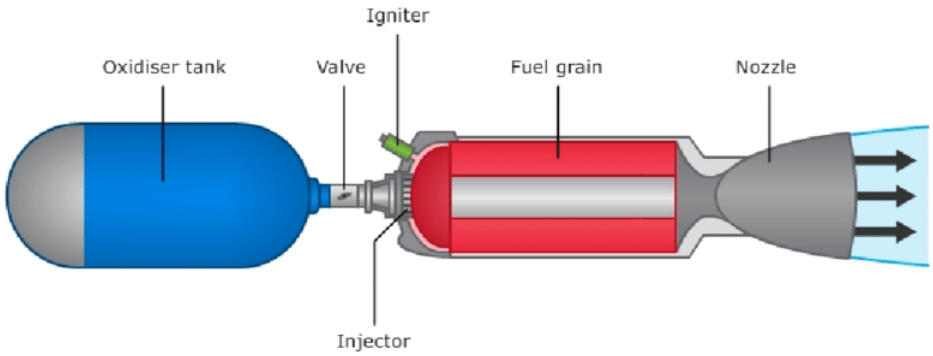

Figure 1: Hybrid Motor Layout

The simulator consists of seven main modules: Tank, Feed System/Injector, Regression, Combustion, Chamber, Nozzle, and Center of Mass (COM).

The tank is modeled as a thermodynamic system containing a self-pressurizing, saturated liquid such as nitrous oxide or carbon dioxide. Upon rocket launch, this liquid flows through the feed system to the injector (or through a vent, if present). As the liquid exits, the tank pressure drops, pushing the fluid past the saturation line. To restore equilibrium, a portion of the liquid rapidly boils off, a process assumed to occur almost instantaneously due to its speed. This vaporization cools the remaining liquid, as the energy required for the phase change is drawn from the fluid itself.

The flow rate from the tank depends on the pressure differential between the tank and the combustion chamber, as well as the geometry of the injector and feed system. The combustion chamber pressure, in turn, is influenced by the fuel's regression rate and the injector's mass flow rate. The exhaust flow through the nozzle, which is driven by the chamber pressure, ultimately determines the rocket's thrust.

5.2 Valve Coefficients

Several coefficients are used to quantify the efficiency of flow through a valve or orifice: the Discharge Coefficient (C_d), Loss Coefficient (k), Flow Coefficient (C_v), and Flow Factor (k_v). Each represents a distinct measurement, so care must be taken to avoid confusing them.

5.2.1 Discharge Coefficient

The discharge coefficient (C_d), also known as the efflux coefficient, is a dimensionless parameter that relates the actual flow rate to the ideal flow rate:

$$C_d = \frac{\dot{m}_{real}}{\dot{m}_{ideal}} \tag{1}$$

The ideal flow rate can be derived by simplifying Bernoulli's equation:

$$P_1 + \rho g h_1 + \frac{1}{2}\rho v_1^2 = P_2 + \rho g h_2 + \frac{1}{2}\rho v_2^2 \tag{2}$$

For a typical rocket feed system, the height difference is negligible. Additionally, the volume upstream of the valve or injector orifice is assumed to be large enough that the fluid velocity at point 1 is approximately zero compared to the velocity at point 2. Simplifying under these assumptions yields:

$$P_1 = P_2 + \frac{1}{2}\rho v_2^2 \tag{3}$$

We know that $v = \frac{\dot{m}}{\rho A}$. Simplifying this relation gives us the equation for ideal orifice mass flow:

$$\Delta P_{inj} = \frac{1}{2}\rho v_2^2 \tag{4}$$

$$\Delta P_{inj} = \frac{\dot{m}^2}{2\rho A^2} \tag{5}$$

$$\dot{m} = A\sqrt{2\rho\Delta P_{inj}} \tag{6}$$

With the original equation for C_d we can get an equation for $\dot{m}_{real}$

$$\dot{m}_{real} = C_d \cdot A\sqrt{2\rho\Delta P_{inj}} \tag{7}$$

The discharge coefficient is a straightforward efficiency factor that accounts for real-world deviations from ideal flow. However, it has some limitations. The value of C_d is not generalizable and must be experimentally determined for each specific orifice size and shape. Additionally, discharge coefficients from multiple valves or fittings cannot be summed directly.

5.2.2 Loss Coefficient

The loss coefficient (K), also known as the minor loss coefficient or head loss coefficient, is a dimensionless property that quantifies the pressure loss across an orifice.

$$h_L = K\frac{v^2}{2g} \tag{8}$$

$$\Delta P = K\frac{v^2\rho}{2} \tag{9}$$

By applying this to the discharging scenario described in section 5.2.1 it can be shown that

$$\dot{m}_{real} = A\sqrt{\frac{2\rho\Delta P_{inj}}{k}} \tag{10}$$

In the case of injector discharge the the loss coefficient (K) can be expressed as

$$C_d = \frac{1}{\sqrt{k}} \tag{11}$$

The loss coefficient is useful because they can be added to get a $K_{total} = K_1 + K_2$..... for valves, orifices, pies etc placed in series. There also exists tools to estimate their value for various basic valve, fitting, and pipe shapes that can be found online or in fluid dynamics textbooks.

5.2.3 Flow Factor / Flow Coefficient

The Flow Factor (k_v metric units) and Flow Coefficient (C_v imperial units) quantify the flow capacity of a valve. It represents the flow rate (m^3/h) of water through an orifice at a standard pressure drop. **Note: The units on this are very unusual and are different for the imperial C_v refer to Wikipedia for more information**.

$$k_v = \sqrt{\frac{SG}{\Delta P}} \tag{12}$$

Specific gravity (SG) is also referred to as relative density. It is the ratio of density to that of water. Once again converting and applying the Flow Factor to the injector situation in 5.2.1, a relationship between C_d and k_v can be made.

$$k_v = \frac{C_d A \Delta P\sqrt{2\rho_{H_2O}}}{N_{hs}\rho_{fluid}} \tag{13}$$

N_{hs} Represents a factor of 1/3600 used to convert the flow rate from units of m^3/h to m^3/s.

The Flow Factor and Flow Coefficient are useful when trying to control flow rates rather than pressure drops. To convert from metric to imperial $C_v = 1.156 \cdot k_v$. The Flow Factor for subsequent valves, fittings and orifices in series can be added using the inverse squares

$$\frac{1}{k^2_{v,total}} = \frac{1}{k^2_{v1}} + \frac{1}{k^2_{v2}} + \dots \tag{14}$$

It should be noted that the loss coefficient, flow factor and flow coefficient are designed for use with single phase incompressible fluids. For additional reading on Valve Coefficients, see [10]

5.3 Tank

Nitrous oxide's unique saturated properties create a self-pressurizing mechanism that distinguishes it from traditional rocket oxidizers. As nitrous oxide flows from the tank, the liquid level drops, causing an expansion of the vapor space. This expansion triggers the vapor pressure to drop, prompting liquid to vaporize and attempt to restore pressure. Critically, this vaporization process extracts thermal energy from the remaining liquid, progressively cooling the tank contents.

The tank model uses an insulated control volume with a saturated liquid vapor mixture inside. The volume of vapor at the top of the tank is referred to as 'ullage'. Both models neglect the heat transfer from the tank wall and only consider energy leaving the tank and converting from liquid to vapor. The liquid/vapor can leave the tank through the injector and optionally through the vent if enabled. The injector is assumed to be at the bottom of the tank where only liquid can exit initially. The vent is assumed to take from the vapor section of the tank and can be configured to vent into the chamber or the atmosphere.

There are three tank models available in the simulator. The first model is the method used by the original HRAP simulator [1]. The HRAP model is originally based on two technical papers published by Rick Newlands of Aspire Space [6][7]. Both of these papers are **MUST READS** for understanding how self-pressurizing hybrid rocket motors work. This HRAP method relies on an integration routine that converges on the amount of liquid that vaporizes each time step.

The second method is based on [8] and uses the internal energy of the tank to rebalance the liquid vapor mixture after every time step. This method produces slightly different results and is considered to be more accurate because it relies on fundamental thermodynamic equations rather than waiting for the vaporized liquid value to converge over several time steps.

Notes:

- The tank acts as a two-phase closed-loop thermosiphon. This is a type of heat transfer device used in geothermal energy to transmit small temperature differences very large distances. A combination of the heat pump effect and convection means that the tank can transmit heat very fast (200-500 times faster than copper by some sources[11]). This means that the oxidizer itself is almost incapable of maintaining a thermal gradient and can be treated as one uniform temperature.

5.3.1 HRAP Liquid Method

The vaporization process is estimated iteratively. An initial guess is made for the amount of oxidizer vaporized (m_v) to restore the tank pressure. This value is then refined at each time step and quickly converges to the actual vaporized mass. Once it has converged, the value sticks with the actual value until the tank is nearly all out of liquid. The energy loss due to vaporization is calculated as:

$$\Delta Q = m_v H_v \tag{15}$$

where H_v is the latent heat of vaporization. The temperature drop in the remaining liquid nitrous is given by:

$$\Delta T_{liquid} = -\frac{Q}{m_{liquid} C_{liquid}} \tag{16}$$

Using the temperature change (ΔT_{liquid}), the temperature of the liquid oxidizer is updated. The equations of state are then recalculated to obtain the new liquid and vapor densities, as well as the vapor pressure. Based on the mass flow calculations discussed in section 5.2.1, the mass flow rate is computed. The amount of mass that exits the tank during this time step is $\dot{m} \cdot \Delta T$. The resulting mass of liquid oxidizer is the amount of liquid that would remain if no reaction had occurred. This is referred to as $m_{liquid,unvaporized}$. Since the mixture is saturated, the total density remains unchanged. With the known tank volume and oxidizer mass, the new mass of liquid can be calculated as:

$$m_{liquid} = \frac{V_{tank} - \frac{m_{total}}{\rho_{vapor}}}{\frac{1}{\rho_{liquid}} - \frac{1}{\rho_{vapor}}} \tag{17}$$

The vaporized mass is then updated as:

$$m_v = m_{liquid,unvaporized} - m_{liquid} \tag{18}$$

This iterative process is repeated for each time step.

As the tank nears depletion, the model begins to show that vapor is condensing within the tank instead of vaporizing. This behavior is not physically accurate. To address this, the model switches to a mode where a constant ΔT is assumed based on the average temperature drop observed so far. To avoid repeatedly calling the equation of state function, the model instead uses an average pressure drop, simplifying the calculations.

5.3.2 HRAP Remastered Liquid Model

This model is a refined and updated version of the HRAP simulator. A detailed discussion of the differences between the original HRAP model and the current version can be found in section 7.2.

5.3.3 Internal Energy Liquid Model

The internal energy method is considered more accurate because it relies on fundamental thermodynamic equations, as opposed to waiting for the iterative vaporized mass to converge.

The model begins by calculating the mass flow and the resulting mass that has exited the tank during the time step. Next, the remaining mass in the tank and the remaining internal energy are determined.

$$m_{total} = m_{total} - (\dot{m}_{inj} \cdot \Delta t) - (\dot{m}_{vent} \cdot \Delta t) \tag{19}$$

$$U_{total} = U_{total} - (\dot{m}_{inj} \cdot \Delta t \cdot h_{liquid}) - (\dot{m}_{vent} \cdot \Delta t \cdot h_{vapor}) \tag{20}$$

The enthalpy is used here because the total energy leaving the system includes both internal energy and pressure energy. Using the equations of state, the internal energy of the liquid and vapor components is calculated; however, this is done at the previous tank temperature.

To determine the new tank temperature, an iterative loop is set up, which converges on the correct temperature by recalculating the individual internal energies, mixture quality, and tank volume until the calculated volume matches the actual tank volume.

Once the new temperature is determined, the equations of state are used to calculate the vapor pressure, enthalpies, and mass components for the next time step.

5.3.4 Vapor Phase

Once the liquid oxidizer has been depleted, the models switch to a common vapor phase. As with the liquid models, mass flow out of the injector (and vent) is calculated, but this time using a vapor mass flow model. The vapor phase is solved through an iterative loop.

Initially, the temperature and pressure are stored, along with the values for C_p, C_v, γ, and the compressibility factor Z_1. A new temperature and pressure are then calculated using the following relations for a real gas near the saturation line:

$$T_2 = T_{tank} \cdot \left(\frac{Z_2 \cdot m_{ox}, 2}{Z_1 \cdot m_{ox,1}} \right)^{\gamma - 1} \tag{21}$$

$$P_2 = P_{tank} \cdot \left(\frac{T_2}{T_{tank}} \right)^{\frac{\gamma}{\gamma - 1}} \tag{22}$$

Where $Z_{2,initial} = Z_1$. Using the new pressure and temperature, the compressibility factor Z_2 is recalculated using the equations of state. The average of the old and new compressibility factors is assigned to Z_2, and the loop repeats until the change in Z_2 becomes negligible. Typically, this process takes between 4 to 9 iterations. Since the vapor phase is relatively short compared to the liquid phase, this iterative process is computationally feasible within the larger simulation loop.

The HRAP tank uses the custom equation of state function `Nitrous Properties`, which uses curve fit equations. In contrast, the Energy tank model uses exact equations of state from CoolProp. The HRAP version of the vapor phase tends to end at higher remaining oxidizer levels because the curve fits become less accurate near the extremes (end of the burn).

5.4 Injector

Injectors are a complicated subject and could involve hours of discussion, but I'll try to keep it brief. For the purposes of this simulator, the entire feed system is represented by the injector. If it helps, you can think of the bottom of the tank as the injector plate, and the real-world feed system is accounted for within the efficiency factor (i.e., the discharge coefficient C_d, inj). This simplification makes the modeling process much easier and allows the model to be more adaptable to feed system alterations. However, this simplification comes at the cost of some accuracy.

The simulator can handle basic shower-head injector designs (a plate with multiple holes). The complexity of injectors arises from modeling saturated liquid being forced through a small orifice. When liquid passes through a small orifice, it can be largely treated as an incompressible fluid, so a simple bell inlet equation can be used to model the mass flow. Traditional liquid injectors also have to account for effects such as vena contracta and boundary layers, which are typically captured in the discharge coefficient as discussed in section 5.2.1.

For hybrid motors, an additional factor must be considered. The liquid leaving the tank is in a saturated state, meaning it is ready to vaporize at even the smallest pressure drop. As the liquid flows through the orifice, the pressure drops due to the venturi effect, potentially causing the liquid to vaporize within the injector. If enough of the liquid vaporizes, the compressibility effects of the vapor can take over and lead to choked flow. This mixture of vapor and liquid is broadly referred to as two-phase flow and is the topic of extensive research.

Depending on the injector and feed system design, the transient vaporization process may or may not have time to occur within the injector. For most thin plate shower-head injectors, the transient vaporization process occurs more slowly than the orifice residence time, so it is acceptable to model the fluid as a single-phase incompressible liquid. This is why the original HRAP simulator uses the single-phase incompressible method for the injector. However, for injectors with an L/D ratio greater than 3, vaporization may start to influence the mass flow rate. Several models have been developed to address this, as discussed in [9]. The simulator incorporates three of these models: Single Phase Incompressible, Homogeneous Equilibrium, and Dyer. Figure 2 shows how the mass flow evolves differently for each model

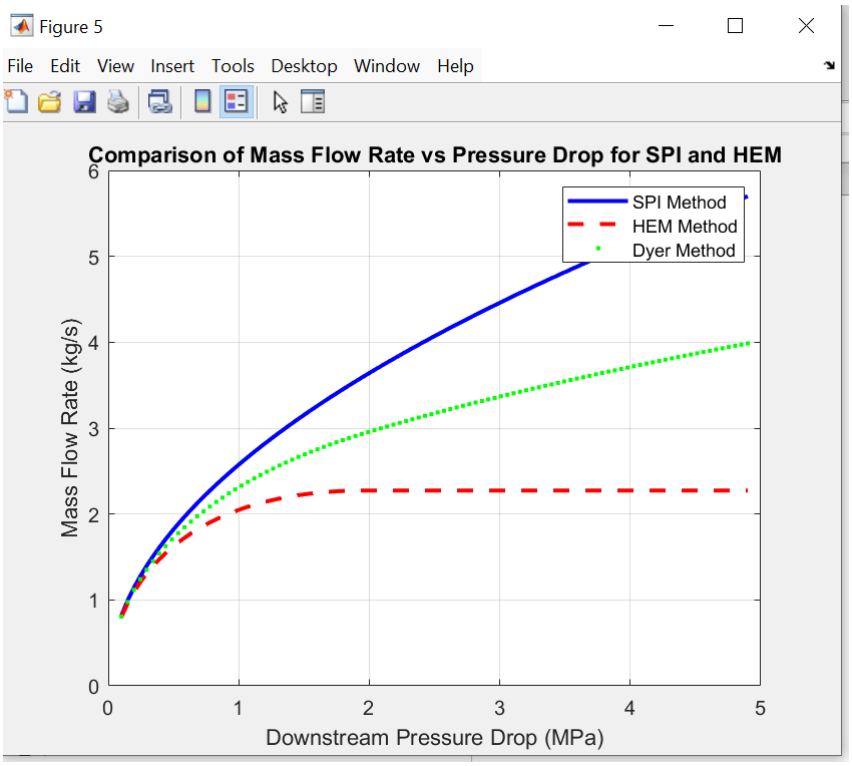

Figure 2: Comparison of Mass Flow Rates

5.4.1 Single Phase Incompressible (SPI)

This mass flow model assumes the fluid leaving the tank to be a single phase incompressible liquid. This allows us to use equation (7) derived in section 5.2.1 using Bernoulli's Equation.

$$\dot{m}_{real} = C_d \cdot A\sqrt{2\rho\Delta P_{inj}} \tag{23}$$

The original HRAP program used this model universally, including on the vent.

5.4.2 Homogeneous Equilibrium (HEM)

The HEM method uses internal energy combined with the continuity equation to produce the following relationship [9].

$$\dot{m}_{crit} = C_d A \rho_2 \sqrt{2\left(h_{ox,l} - h_2\right)} \tag{24}$$

This relationship exhibits a maximum as the downstream pressure drops. Beyond this maximum (critical pressure P_{crit}), the downstream pressure becomes low enough to cause

vaporization and choking within the injector, limiting the mass flow rate. Consequently, the mass flow rate will not increase past this critical point.

The model brute forces the first P_{crit} value. For subsequent P_{crit} values, the model only searches for pressures below the initial one to save compute time.

5.4.3 Dyer (Non Homogeneous Non Equilibrium)

This method uses a weighted mixture of the previous two models. The weighting factor k is meant to be adaptable to supercharged hybrid motors where an additional nitrogen pressure tank is added. In the case of a normal self-pressurizing nitrous rocket P_v is the same as P_{tank} and thus $k = 1$.

$$k = \sqrt{\frac{P_{tank} - P_2}{P_v - P_2}} \tag{25}$$

The other two models are weighted as follows:

$$\dot{m} = \left(\frac{k}{1+k}\right)\dot{m}_{SPI} + \left(\frac{1}{1+k}\right)\dot{m}_{HEM} \tag{26}$$

5.5 Regression

The regression phase calculates the current OF ratio and the amount of fuel that will vaporize from the grain during the time step. There are two primary methods for modeling fuel regression. The first method is based on calculating the mass flux G_{port} from the grain, using the previous OF ratio as a starting point. This mass flux is then fed into a modified form of St. Robert's Law, which allows the determination of the new regression rate.

$$\dot{r}_{grn} = aG_{port}^{n}L_{grn}^{m} \tag{27}$$

The coefficients a, n, and m are not always available before the engine has been hot-fired, as they are highly dependent on specific motor characteristics and grain formulations. In the absence of this experimental data, the second method becomes useful.

The second regression model uses a static OF ratio that is set based on historical data from similar engines or fuel grain types.

5.6 Combustion

The combustion phase uses data compiled in the selected `fuel.m` configuration. This data is derived from a NASA CEA simulation, and new fuel files can be created for a specific fuel by copying the template. This phase of the simulation is a direct clone of the HRAP simulator. More information on fuel configurations can be found in the HRAP *Theory of Operation* [4].

The function calculates C^* by interpolating the specific heat ratio, molar mass, and temperature from tables in the fuel configuration. Once the necessary properties are determined, C^* can be computed using the following relation:

$$C^* = C^*_{eff} \sqrt{\frac{R_{mixture} T}{\gamma \left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}}} \tag{28}$$

5.7 Chamber

This phase first calculates the new volume of the chamber and the change in chamber volume based on the regression rate.

$$\dot{V}_{cmbr,grn} = \frac{0.25\pi \left(ID^2_{grn} - (ID_{grn} - 2\dot{r}_{grn}\Delta t)^2\right) l_{grn}}{\Delta t} \tag{29}$$

Next, the mass of gas in the chamber is calculated

$$\dot{m}_{nozzle} = P_{cmbr} \cdot C_{d,nozzle} \cdot \frac{(0.25\pi \cdot D^2_{throat})}{C^*} \tag{30}$$

$$\dot{m}_{gas} = \dot{m}_{fuel} + \dot{m}_{inj} - \dot{m}_{nozzle} \tag{31}$$

$$m_{gas} = m_{gas} + \dot{m}_{gas} \cdot \Delta t \tag{32}$$

With the mass of gas in the chamber, it becomes possible to find the new chamber pressure.

$$\dot{P}_{cmbr} = P_{cmbr} \cdot \left(\frac{\dot{m}_{gas}}{m_{gas}} - \frac{\dot{V}_{cmbr,grain}}{V_{cmbr,grain}}\right) \tag{33}$$

$$P_{cmbr} = P_{cmbr} + \dot{P}_{cmbr} \cdot \Delta t \tag{34}$$

5.8 Nozzle

Finally, using basic adiabatic compressible flow equations, it is possible to determine the thrust of the rocket. First, the exit Mach number can be found by using the following nozzle relationship:

$$ER = \left(\left(\frac{2}{\gamma+1}\right) \cdot \left(1 + \frac{\gamma-1}{2} M^2\right)\right)^{\frac{\gamma+1}{2(\gamma-1)}} \cdot \frac{1}{M_{exit}} \tag{35}$$

Next, the exit pressure and thrust coefficient C_T can be found [12].

$$P_{exit} = P_{cmbr} \left(1 + 0.5 \cdot (\gamma-1) M^2_{exit}\right)^{-\frac{\gamma}{\gamma-1}} \tag{36}$$

$$C_T = \sqrt{\left(\frac{2\gamma^2}{\gamma-1}\right)\left(\frac{2}{\gamma+1}\right)^{\frac{\gamma+1}{\gamma-1}}\left[1-\left(\frac{P_{exit}}{P_{cmbr}}\right)^{\frac{\gamma-1}{\gamma}}\right]} + \frac{(P_{exit}-P_{atm})\,(0.25\pi d_{throat}^2\,ER)}{P_{cmbr}\cdot 0.25\pi d_{throat}^2} \quad (37)$$

One of two models can be selected to calculate the final rocket thrust. The first and simpler model assumes that all the mass leaving the nozzle is traveling in a one-dimensional flow straight backward from the nozzle. This "1D" model can be computed by applying the efficiency factors to the classical rocket force equation:

$$F_{rocket} = \eta_{noz}\cdot C_{d,noz}\cdot C_T\cdot 0.25\pi d_{throat}^2\cdot P_{cmbr} \quad (38)$$

For wide conical nozzles, there are losses due to the divergence of the nozzle. The ideal rocket force is calculated first, then the pressure contribution is subtracted to isolate the mass flow component. The divergence loss only affects the mass flow contribution of the rocket force, where gas that has a radial velocity component takes away from the axial momentum imparted on the vehicle.

$$F_{rocket,ideal} = C_T\cdot 0.25\pi d_{throat}^2\cdot P_{cmbr} \quad (39)$$

$$\lambda_{divergence} = \frac{1}{2}\left(1+\cos(\alpha_{noz})\right) \quad (40)$$

$$F_{rocket,vel} = F_{rocket,ideal} - (0.25\pi d_{throat}^2\cdot ER)\cdot(P_{exit}-P_{atm}) \quad (41)$$

$$F_{rocket,vel} = F_{rocket,vel}\cdot\lambda_{divergence} \quad (42)$$

Finally, the pressure component is returned, and the efficiency factors are applied.

$$F_{rocket} = \eta_{noz}\cdot C_{d,noz}\cdot\left(F_{rocket,vel} + \left(0.25\pi d_{throat}^2\cdot ER\right)\cdot(P_{exit}-P_{atm})\right) \quad (43)$$

5.9 COM

On each time step, the mass of liquid, vapor, fuel and combustion chamber gas is calculated. These are then combined with a simple center of mass calculation. The center of mass of the fuel and gas are assumed to be at the center of the fuel grain.

6 Simulator Design

The simulator discussed is designed to be user-friendly but, more importantly, to be approachable and understandable for an aerospace engineering student. This means that when given a choice between an intuitive structure and an efficient one, the intuitive option was chosen. This approach was taken with the goal of helping future students modify and fix the program more easily. It should also be noted that I (Adrian Comisso) am not a software engineer, and may not be fully aware of all the best practices in programming.

The CHRMS simulator is written in MATLAB, a versatile programming and simulation application with many useful math and system simulation toolboxes. The simulator can be broadly divided into four main parts: the GUI, the main script, the simulation loop, and the calculation functions.

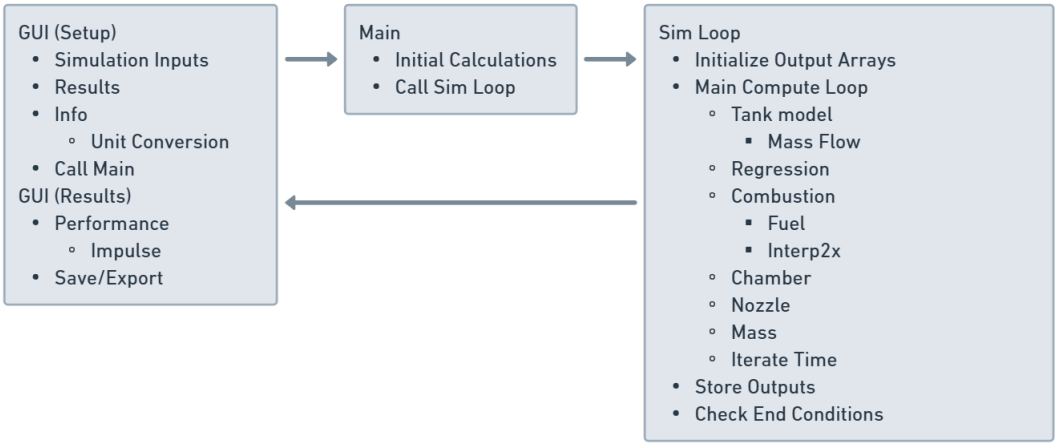

Figure 3: Simulator Flow Diagram

The simulator works by passing what MATLAB calls 'structures' through each function. A structure is a variable that holds multiple related values. For example, the structure `state` may have multiple properties, like `velocity` or `position` that can be accessed and modified (e.g., `state.velocity = 1` or `state.position = 2`). This method makes it easier to store all the variables in one place and pass them between functions, without bloating the code with numerous function inputs and outputs.

In the CHRMS simulator, there are three main structures: state, fuel, and output. The state structure carries all the current variables of the system while the loop is running. The fuel structure contains a loaded configuration variable that stores combustion and other fuel properties. The output structure stores the state at each time step in a large array, which is used for analysis at the end of the simulation.

The simulator uses a numerical approach because the equations of state rely on interpolated tables, which introduce nonlinearities.

For the tank models, the equations of state are necessary to calculate various properties of the oxidizer. The HRAP models use data from the National Physics Laboratory (NPL). NPL is the UK equivalent of the National Institute of Standards and Technology (NIST) in the USA. The data from NPL has been converted into curve-fit functions to allow the `Nitrous Properties` function to be called more quickly. These curve fits, however, break at the edge cases.

The Internal Energy Tank model uses a thermophysical library called CoolProp [13]. This library was chosen because the NIST REFPROP and the original NPL libraries require a subscription. CoolProp is a library that contains 122 different fluids and uses equations of state from various literature sources, verified against the original NIST data. CoolProp is a C++ library, but a Python wrapper is available. While MATLAB has its own wrapper for CoolProp, this is being discontinued in favor of using Python directly with MATLAB. MATLAB can interface with Python, and this is how it calls upon the CoolProp library. However, it's important to note that each MATLAB release only supports certain versions of Python, usually not the latest one. Make sure you have the correct version of Python for your MATLAB version. Instructions on how to ensure MATLAB is properly communicating with Python can be found here [14].

For the combustion function in the simulator, the program relies on interpolated combustion tables generated using NASA's CEA program. These tables are specific to each fuel type. The simulator comes with several fuel configurations borrowed from the original HRAP program, and there is also a template for creating a new configuration. For more information on how to create a fuel file, see [4].

6.1 GUI

The GUI is designed using MATLAB's App Designer, which provides an intuitive and easy-to-use environment for UI development. To open an app in the App Designer, simply open MATLAB and double-click on the app in the file navigation bar on the side. The App Designer allows you to seamlessly switch between the UI design and the callback functions. It conveniently locks users out of editing the app structure code in the callback function view, favoring a graphical approach to editing in the design view. It is not recommended to try making changes to the app without using the App Designer, as doing so would make navigating the 2200 lines of code extremely difficult.

Upon pressing the "Run" command, the GUI performs all necessary unit conversions and assigns the values to the state structure, which will carry the current `state` of the motor throughout the entire simulation.

6.2 Main Script

The main program clears previous data from MATLAB and performs initial calculations such as the total injector area. The simulation loop function is then called upon.

6.3 Simulation Loop

The simulation loop contains the main iterative calculation process that updates the state using the functions discussed in the theory section. It also handles any logic, such as determining whether the burn is a cold flow or if the mass properties should be recalculated. Finally, the loop generates the required output arrays and checks if any of the end conditions have been met. The end conditions are as follows: depleted oxidizer, maximum simulation time reached, tank simulation enters the solid-gas phase, chamber pressure exceeds tank pressure (indicating the end of the cold flow), fuel grain depletion, or chamber pressure dropping below atmospheric pressure.

6.4 Functions

There are numerous functions designed to calculate different sections of the motor. The `state` variable is passed along through them during each time step. Each function is discussed in section 5.

7 How to use

To run this program, you will need:

- MATLAB 2024b or newer.
- A compatible version of python for your MATLAB edition see [14]

To launch the app, run the `app.mlapp` file in your installation folder. This will open the GUI, where you will find 3 tabs. The first tab contains all the simulation setup inputs. The second tab contains the results analysis page, and the last page provides a link to this document and other useful info/sources.

7.1 Simulation Setup

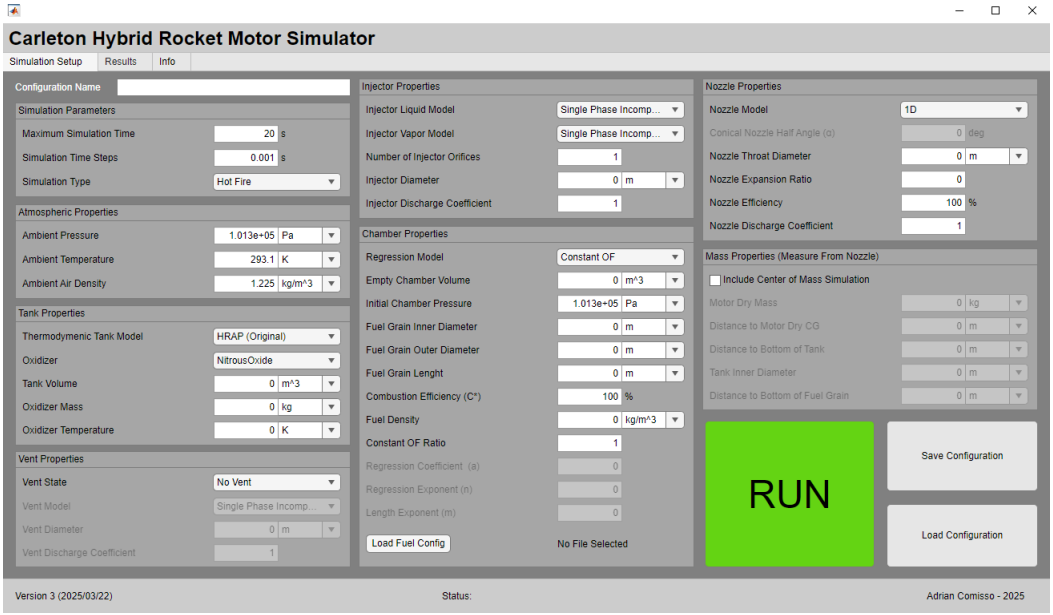

Figure 4: Simulation Setup GUI Page

In the simulation setup tab, you can name your motor configuration in the top-left corner for easy saving later. The simulation type option allows you to choose between Cold Flow, where the injector vents into the air, and Hot Fire, where full combustion and nozzle simulation are active.

For atmospheric properties, you may need to update these settings if the motor is tested at a high-elevation site or if the temperature deviates from the standard 20°C.

Under Tank Properties, there is a model selection drop-down that controls the tank simulation model. More details on each model can be found in Section 5.3. The next drop-down allows you to select the oxidizer. For most hybrid engines, this is typically nitrous, but for cold flow simulations, CO2 is used. Please note, CO2 only works if you select the Energy Tank model.

In Vent Properties, you can choose whether to vent to the atmosphere or into the combustion chamber. If all vents are closed during launch, select "No Vent." The vent model controls the fluid model used to determine the mass flow through the vent at each time step. Two vent models are available; more information on these is provided in Section 5.4. For HRAP-equivalent results, use the single-phase incompressible model. If your motor has multiple vents, the total vent area and discharge coefficient should be combined to create an imaginary 'total' vent. In the future, it may be possible to define multiple vents. See section 8 for more information.

For Injector Properties, select Single Phase Incompressible for HRAP-equivalent results. If the injector orifice Length/Diameter ratio exceeds 3, it may be useful to switch to a different model, although this is not a hard rule. It's best to compare simulation results with test data for all three models and see which applies best to your injector.

In Chamber Properties, the regression model determines whether a constant or shifting oxidizer-to-fuel (OF) ratio is used. Typically, a constant OF ratio is used until the regression coefficients are known. The Empty Chamber Volume refers to the chamber volume with everything (mixing plate, fuel liner) inside except for the fuel grain. At the bottom of the chamber properties a fuel config must be selected. Several fuel configs, such as paraffin and ABS have been copied from the original HRAP simulator. To make new fuel configs, copy the template and follow the instructions in [4].

In Nozzle Properties, you can switch between 1D and 2D nozzle models. The 2D option accounts for slight inefficiencies caused by diverging conical nozzles, as discussed in Section 5.8.

Lastly, in Mass Properties, you can simulate the center of mass if the corresponding box is checked.

To save/load a motor configuration use the save/load buttons in the bottom right corner.

Finally, to run the simulation, press the RUN button. This will convert all the inputs into base SI units and then begin the simulation. Keep in mind the energy tank model and the homogenous equilibrium injector model both add considerable compute time to the result. **Keep an eye on the MATLAB command line. If an error occurs it may show up there depending on what the error is.**

7.2 HRAP vs HRAP Remastered

The program allows users to switch between the new Energy Tank Model and a clone of the original HRAP Model. If the tank model is set to "HRAP", the injector/valve models are set to "SPI" and the nozzle model is set to 1D flow, the simulator will perform EXACTLY the same as the original HRAP program. This feature was added so that users could quickly verify the new model if necessary.

You'll also notice a model called HRAP Remastered. This is a refined version of the original HRAP model. While it uses the same iterative evaporation technique, it corrects several bugs and calculation errors found in the original HRAP program. These issues were relatively minor but caused slight deviations in results.

The following is a list of known bugs identified in the original HRAP program:

- Grain Length Units Bug: The original HRAP GUI doesn't have an "if" condition that checks for units of meters in the grain length input. As a result, the program throws an error if meters are used, preventing the simulation from running. **This issue has**

been fixed in the new CHRMS GUI, and you can now use meters without any problems.

- Tank Pressure Calculation Bug: During the second part of the liquid phase in the HRAP tank model (see section 5.3.1), the model switches from iteratively calculating the new tank pressure to assuming an average pressure drop. In the first iteration, the calculation works fine, but at the end of the iteration, the program calculates the change in tank pressure (ΔP_{tank}) before calling the equation of state function. This means that the vapor pressure in the equation of state is not updated, and the calculated ΔP_{tank} ends up being zero. In subsequent iterations, this zero result affects the average pressure drop, leading to skewed results. This is clearly a mistake, as the order of operations in the first phase is reversed, and the Theory of Operation document [4] does not suggest this behavior. The HRAP Remastered model fixes this issue by adjusting the order of operations.

- Gamma Value Bug in Vapor Phase: During the vapor phase of the HRAP model, the program uses a short loop to iteratively find the correct compressibility factor. The loop uses a fixed value of $\gamma = 1.3$, even though γ changes with pressure and temperature. The equation of state function already calculates a curve fit for γ, making the fixed value unnecessary. The HRAP Remastered model uses the calculated γ value, correcting this issue.

- Mass Flow Calculation Bug: In the original HRAP model, the mass flow is calculated at the beginning of the loop rather than within the conditional statements. If the remaining liquid mass in the tank goes slightly negative when transitioning to the vapor phase, the model continues using the liquid mass flow equation for one time step, even though the liquid mass should have been corrected to zero. HRAP Remastered resolves this issue by ensuring that mass flow is calculated only after the liquid mass has been corrected.

7.3 Results

The Results section functions similarly to the HRAP program. Once a simulation has been run, a summary of the motor's performance will be displayed in the bottom right corner. On the top right, users can configure various graph options, which will then be plotted on the left side of the interface.

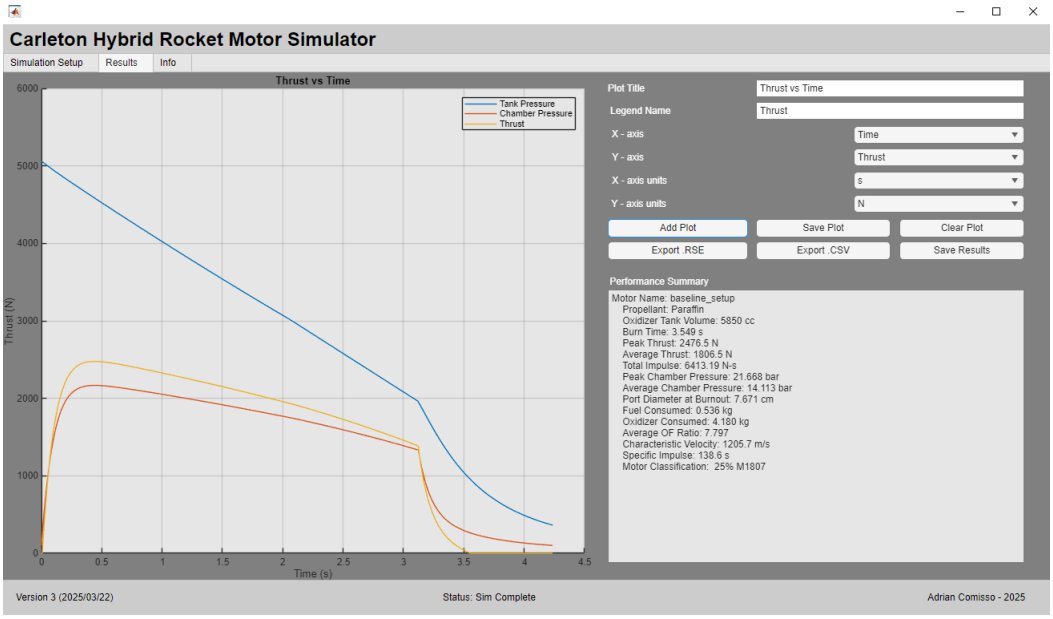

Figure 5: Results GUI Page

8 Future Work

In the future, a Simulink model should be developed for this simulator. By using Simulink it would be possible to make all the motor components modular. This would allow users to add differently configured vents and feed system components. A model like this may even be capable of modeling transient behavior, such as the time it takes for the feed system to fill with nitrous and the time it takes for the nitrous liquid to boil off.

References

[1] "Hybrid rocket analysis program (hrap)." https://github.com/rnickel1/HRAP$_source.Accessed$: $2025-03-26$.

[2] "Cesaroni technologies incorperated." http://cesaroni.net/. Accessed: 2025-03-26.

[3] "Aerotech consumer aerospace." https://aerotech-rocketry.com/. Accessed: 2025-03-26.

[4] R. A. Nickel, "Hrap theory of operation," tech. rep., University of Tennese, 2020.

[5] R. Newlands, "The physiscs of nitrous oxide," tech. rep., Aspire Space, 2004.

[6] R. Newlands, "Modeling nitrous tank emptying," tech. rep., Aspire Space, 2012.

[7] R. Newlands, "Introduction to hybrid design," tech. rep., Aspire Space, 2017.

[8] J. E. Zimmerman, "Review and evaluation of models for self-pressurizing propellant tank dynamics," tech. rep., Stanford University, 2013.

[9] B. S. Waxman, "An investigation of injectors for use with high vapor pressure propellants with applications to hybrid rockets," tech. rep., Stanford University, 2014.

[10] "Valve sizing technical bulletin," *Swagelok Company*, 2007.

[11] S. Noie, "Heat transfer characteristics of a two-phase closed thermosyphon," *Applied Thermal Engineering*, vol. 25, no. 4, pp. 495–506, 2005.

[12] J. Etele, *Fundementals of Transatmospheric Spacecraft Propulsion.* Aldanox Group, 2022.

[13] "Cool prop." http://www.coolprop.org/. Accessed: 2025-03-26.

[14] "Cesaroni technologies incorperated." https://www.mathworks.com/help/matlab/matlab$_e$*xternal*/*i*
supported − *python* − *implementation.html*. *Accessed* : 2025 − 03 − 26.