Simulation of a Rocket Engine

Steinn Hauser, steinnhm

November 2017

Institute of theoretical astrophysics, Blindern University, Oslo
steinnhauser@mac.com

Abstract

A simplified model of a rocket taking off a home planet in a randomly generated solar system. The solar systems characteristics such as the masses, positions and velocities followed with the seed (52772). The goal of the model is to have a rocket take off from it's home planet and reach escape velocity within a time frame of 20 minutes. The model is based on filling the rocket with extremely hot hydrogen-gas which will provide the thrust to accelerate it out of the home planets gravitational pull. To calculate the logistics of this, the engine will be divided up into many small boxes which will be filled with the gas. To simulate this, the number of H_2 particles in each subsection of the engine is needed. This number can further be used to count how many particles would escape a square hole at the bottom of the box to calculate the total momentum gain of the rocket.

Python version 2.7.12 was an ideal programming language and update to simulate the system through various methods using Newton's Laws and numerical integration techniques.

Introduction

In order to one day colonize other planets we must first get there by means of fast rockets with vast engines that generate thrust exceeding what was thought possible a hundred years ago. Before the engineers and mechanics are sent to do their jobs we must first make sure we have a consistently safe model for human spaceflight, meaning we must first simulate the feat thousands of times over to make sure all variables are calibrated to perfection.

The main challenge is getting a 1,000 kg satellite in addition to its rocket fuel to escape it's home planet. This is very difficult to calculate in the real world but in this case the model will be simplified to neglect factors like friction and air resistance. The total mass of the rocket will decline as hydrogen gas is discarded, so the logistics of calculating the acceleration analytically are far inferior to numerical calculations.

Simulating an entire rocket engine in a 20 minute time period requires far more computing power than simply simulating a single box of hydrogen gas in a smaller time period. The model will therefore assume that it can multiply the single box thrust (increasing the number of boxes) to scale the thrust up to one powerful enough to accelerate our rocket to escape velocity.

Method

The simulation will involve modelling a small 3 dimensional cube with length $L = 10^{-6}m$ which will be filled up with 100,000 hydrogen gas particles and raised to a temperature of 10,000 Kelvin (this model will not take into consideration collisions between particles). This means giving each particle a position coordinate $\vec{r} = (x, y, z)$ and a velocity vector $\vec{v} = (v_x, v_y, v_z)$ which will dictate where it is and where it will be at a later point. These are the values that must first be randomly generated to begin the simulation.

The gas particles will be uniformly distributed throughout the box. However, its velocity components must have a Gaussian distribution centered at $\vec{v} = 0$ (Since the velocities can be positive and negative and both must be equally probable). The standard deviation of the distribution can be found by the formula:

$$\sigma = \sqrt{\frac{kT}{m}}$$

Where T is the gas' temperature in Kelvin, k is Boltzmann's constant and m is a single particles mass.

The box will have a small square hole centered at its bottom with an area of $(\frac{L}{2})^2$ square

meters where the particles will be able to escape the box. A particle escaping results in the rocket receiving the momentum the particle had (in the vertical direction only) since the total momentum of the system must be held constant. Each time a particle escapes, it will be replaced by a new particle with the same kinetic energy, so as to make sure that the box does not empty or cool down. From this the thrust force generated by a single box can be found in some time interval Δt given the equation $\vec{F} = \frac{\Delta \vec{p}}{\Delta t}$. This approximation is most realistic given a small time interval, so $\Delta t = 10^{-9}s$ is chosen as simulation time. The particles will be set in motion at a start time $t_0 = 0$ and chaotically bounce around the box, having their velocity components flipped accordingly (the angle of entrance staying the same) when hitting the walled off areas. To make sure that the gas particles don't pass through a wall (through imprecise computing) it is important to take many time steps in the interval from t_0 to Δt. Having 1,000 time steps in the interval should make sure no imprecision affects the calculations, as we will be updating the position- and velocity vectors every $\delta t = 10^{-12}$ seconds. The principle is to have thousands of these boxes simulated over a longer time period for our rocket engine to successfully accelerate a satellite out of the home planets gravitational pull.

The simulation will be initiated with three randomly generated (uniform distribution) start coordinates for each particle, saved in the form of a 3x100,000x1000 matrix (three "x-y-z" coordinates times 100,000 particles times 1000 time steps), in addition to three randomly generated (Gaussian distribution) start velocities for each particle (saved in the same matrix format).

The simulation has the origin in one of the boxes 8 corners, placed such that $0 \leq x, y, z \leq L$. The hole of the box is placed on the side $z = 0$, such that $\frac{L}{4} \leq x, y, \leq \frac{3L}{4}$ is where the particles can escape.

For each time step δt, the program will check for particle positions to make sure that, if they are outside (or equal to) the confines of the 6 walls, their velocity components will be flipped accordingly. However, this bounce off the wall will be overwritten if they happen to hit the wall $z = 0$ in the area where they can escape. In this case the particle escaping will be noted (saving the particles velocity in z-direction), then discarded. Each time this happens, a new particle will be released from a hole at the top of the box (position coordinates set to $\vec{r} = (\frac{L}{2}, \frac{L}{2}, L)$, and velocity vector equal to that of the particle that escaped) to make sure the number of particles in the box (as well as the temperature of the box) is the same at all times.

Checking the positions of 100,000 particles 1,000 times each adds up to 100,000,000 calculations and checks to compute. Designing the program with for-, while-, and if-loops to check each particle takes far too much time. Vectorizing the code is therefore a great way to cut down on calculation time, so functions like numpy's *logical_and*() and *logical_or*() are very useful here since they take two conditions or statements and assesses their validity. Thereafter, a function like numpy's *where*() (which takes in a condition and returns it coordinates) can be used to know which velocities should be flipped. Flipping the velocities is simply multiplying the velocity array with another array of ones where the ones are negative when the wall condition is met.

After simulating, the program will be met with some numbers such as how many particles escaped the box and how fast each of their z velocity components were. This data is essential as it can be used to confirm that our simulation is running properly and that all numerical values match with any analytic calculation there might be. Examples of analytic calculations the numerical values can compare with are:

Equation of state: $P = nkT$
Mean kinetic energy: $< E_k >= \frac{3}{2}kT$
Mean particle velocity: $< v >= \int_0^\infty vP(v)dv$

To calculate the mean particle velocity the probability for speed (absolute value of the

velocity $P(v) = P(|\vec{v}|))$ for a given particle is used:

$$P(v) = (\frac{m}{2\pi kT})^{\frac{3}{2}} e^{-\frac{mv^2}{2kT}} 4\pi v^2$$

$$\Rightarrow < v >= \int_0^\infty v(\frac{m}{2\pi kT})^{\frac{3}{2}} e^{-\frac{mv^2}{2kT}} 4\pi v^2 dv$$

$$\Rightarrow < v >= 4\pi(\frac{m}{2\pi kT})^{\frac{3}{2}} \int_0^\infty v^3 e^{-\frac{mv^2}{2kT}} dv$$

Use substitution method; setting

$$u = \frac{mv^2}{2kT} \Rightarrow du = \frac{mv}{kT}dv \Rightarrow dv = \frac{kT}{mv}du$$

$$\Rightarrow < v >= 4\pi(\frac{m}{2\pi kT})^{\frac{3}{2}} \int_0^\infty v^3 e^{-u} \frac{kT}{mv} du$$

Inserting $v^2 = \frac{2kTu}{m}$ gives us:

$$< v >= 4\pi(\frac{m}{2\pi kT})^{\frac{3}{2}} \frac{kT}{m} \int_0^\infty \frac{2kTu}{m} e^{-u} du$$

Using the rule $\int_0^\infty xe^{-x}dx = 1$ returns:

$$< v >= \sqrt{\frac{8kT}{\pi m}}$$

Which can be compared to the numeric mean particle velocity to check if the simulation is working properly. The results of these comparisons are under the section *Results*.

To check if these values agree with the numeric calculations we must find a method to calculate the values numerically. First, calculating the pressure numerically involves summing up all the particle bounces off a certain wall (choose here wall y=0) to find the force:

$$F = 2\frac{dp}{dt} = 2m\frac{dv}{dt} = 2m\sum_i \frac{v_{y,i}}{\Delta t}$$

Where $\sum_i v_{y,i}$ is the sum of all the y-component speeds of the particles which hit the wall. Multiplying the speed with two takes into consideration that the particle goes from $(-p_y\hat{j})$ to $(p_y\hat{j})$ when bouncing off the wall at y=0. Using $P = F/A$ gives us the pressure expression:

$$P = \frac{2m}{L^2\Delta t}\sum_i v_{y,i}$$

Where the area of the side of the box is L^2. This can be used to calculate and compare to the analytic results. Additionally, the numeric calculations for kinetic energy velocities can be found:

$$< v >= \frac{1}{N}\sum_{i=0}^{N} \sqrt{v_{x,i}^2 + v_{y,i}^2 + v_{z,i}^2}$$

Where N is the number of particles in the box. From this we can get the mean kinetic energy:

$$< E_k >= \frac{1}{2}m < v >^2$$

This data can also be used to calculate the momentum the rocket will gain from each box, and from that calculate how many boxes will be needed to accelerate to escape velocity within some time Δt (assuming the rocket has a constant mass) using the formulas:

$$F = \frac{dp_{total}}{dt} = \frac{dp_{box}}{dt} \cdot nrofboxes \tag{1}$$

$$F = ma = m\frac{\Delta v}{\Delta t} \tag{2}$$

These equations can be combined to find the number of boxes:

$$\frac{dp_{box}}{dt} \cdot nrofboxes = m\frac{\Delta v}{\Delta t} \Rightarrow nrofboxes = m\frac{\Delta v}{\Delta t}\frac{dt}{dp}$$

Where Δv is the escape velocity, found by using energy conservation $E_0 = E_1$:

$$E_0 = \frac{1}{2}mv_{esc}^2 - G\frac{Mm}{r_0^2}$$

$$E_1 = \frac{1}{2}mv_1^2 - G\frac{Mm}{r_1^2} = 0$$

$$\Rightarrow v_{esc} = \sqrt{\frac{2GM}{r_0}}$$

Where r_0 is the home planets radius, $r_1 = \infty$, M is the mass of the home planet, and m is the mass of the rocket. G is the gravitational constant, v_{esc} is the escape velocity and v_1 is the final velocity, which should be zero. Furthermore, Δt is the new simulation time (which will be set to 20 minutes), dp is the momentum gain for one box in some time interval dt, and m is the total mass of the rocket.

Since this model will take into consideration a mass that varies in time (due to fuel loss) as well as the force of gravity, an equation for the acceleration at any given time must be found. For this, the *equation of state* can be used to find the pressure, and use $P = \frac{F}{A}$ and $F = ma$ to find the acceleration:

$$P = nkT = \frac{F}{A}$$

$$\frac{N}{L^3}kT = \frac{ma}{(\frac{L}{2})^2} \Rightarrow a = \frac{NkT}{4mL}$$

Where N is the number of particles in the box, k is Boltzmann's constant, T is the temperature of the gas in the box (in Kelvin), m is the current mass of the rocket and L is the length of each box. Now the only variable of the equation is the mass, which can be integrated into the Euler-Chromer loop.

Now there is a gravitational force pulling the rocket down (as well as the rockets mass being larger than before), so it will most likely not reach the escape velocity within the same time frame. Following are the results of the program calculations:

Results

———————	Analytic Calculated	Numeric Calculated
Pressure	$1.381 \cdot 10^4 Pa$	$1.468 \cdot 10^4 Pa$
Kinetic Energy	$2.071 \cdot 10^{-19} J$	$2.067 \cdot 10^{-19} J$
Mean Velocity	$10248 \frac{m}{s}$	$10229 \frac{m}{s}$

Escape Velocity	$16285.4 \ \frac{m}{s}$
Nr. of boxes	$1.335 \cdot 10^{10} boxes$
Amount of fuel	$2017.7 kg$
Velocity after 20min	$10248 \frac{m}{s}$

Fine-tuning the variables to reach escape velocity after 20 minutes involved scaling the number of boxes needed by 3 and the amount of fuel was scaled by a factor 2.5. This resulted in a velocity of $16287.9 \frac{m}{s}$.

Comments

The numeric calculations match quite well with the analytic ones and we can argue that the simulation of the gas particles in the box are quite consistent with reality. The velocity of the rocket at the end of the simulation was not enough to get to the escape velocity needed for our planet, so the variables needed some changing to have the rocket exit the planets gravitational pull. The final product was found through a lot of trial and error where the variables were scaled up and down according to the velocity at the end of the simulation. The trick here is to have the rocket be as efficient as possible; were the rocket to have a final velocity higher than the escape velocity, the number of boxes would be decreased. Similarly, the number of boxes had to be increased when the final velocity was not enough. The goal was to reach escape velocity but not overshoot it either, which was accomplished.

Conclusion

The rocket engine has many complications and simplifications, but managed to reach escape velocity in under twenty minutes at the end of the simulation. The rocket ended up weighing $7051.7 kg$ (satellite weight included), having $3.34 \cdot 10^{10}$ boxes, and provided enough thrust to accelerate the rocket to a speed of $16287.9 \frac{m}{s}$ upward in a time interval of 20 minutes. Although simplifications made throughout the simulation were quite drastic, the results are not too far off how a true rocket engine behaves. The largest error margins likely come from not taking atmospheric drag force into account, as well as the many simplifications of an engine composition.

Summary

To summarize, the numeric simulation of a gas chamber had close ties to reality, as seen by the comparison with analytic calculations. The error-margin for the program results is directly connected with the simulation time interval δt and the inaccuracies that arise when this interval is larger than normal. The closer δt is to zero, the more accurate the gas chamber model becomes. Therefore, given enough computing power, this model can be used to predict how a true rocket engine behaves.

References

Frode K. H. Astronomy Lecture notes

http://www.uio.no/studier/emner/matnat/astro/AST2000/h17/undervisningsmateriale-2017/forelesningsnotater/part1a_v2.pdf

AST2000SolarSystemViewer Module for planet information (seed=52772)

http://www.uio.no/studier/emner/matnat/astro/AST2000/h17/undervisningsmateriale-2017/filer-til-oppgaver/ast2000solarsystemviewer_v2.pyc