AST 2000 - Part 1
Modelling A Rocket Engine

Welcome to Part 1 of the AST 2000 Satellite Project. To begin the journey we will begin with some elementary statistical physics and rocket science. Using our findings we will simulate a rocket engine using a particle description of our fuel mass; next, we will use data from the engine simulation in order to simulate a satellite launch.

GOALS

- ∇ Derive three fundamental results in statistical physics.
- ∇ Simulate the harsh conditions inside your rocket engine.
- ∇ Model a rocket engine's output based on your simulation.
- ∇ Test your newly designed rocket engine and use it to simulate a satellite launch from your home planet.

THE ROCKET ENGINE

1. Relevant Physics

The principles behind our rocket engine model are discussed in Lecture Notes 1A. You should *absolutely* read the lecture notes before you continue.

In short: The fuel mass is modelled as a collection of point particles trapped in a chamber. We will ignore quantum mechanics (for now) and have the particles behave according to the Maxwell-Boltzmann distribution. A hole on the side of the chamber will then be opened such that our particles can escape. Finally the rocket engine will be propelled forwards by the momentum loss from the escaped particles (conservation of momentum).

2. Simplifications and Assumptions

- ⊗ Our fuel tanks are equipped with pure H_2 gas.
- ⊗ While the temperature and the density of the gas vary greatly on the microscopic level, we will assume the gas is uniform on the macroscopic scale. To accomplish this, the temperature in the Maxwell-Boltzmann distribution and the total number of particles in the chamber are both kept constant.
- ⊗ Gravitational effects are negligible during the particle simulation.
- ⊗ The particles have no spatial extension, thus we ignore the possibility of particle-particle collisions.
- ⊗ When a particle collides with a chamber wall, the collision is perfectly elastic.

CHALLENGES

The programs you write in this part will most likely be run several times. Try to write the programs with a sense of generality in mind; lots of comments and easy-to-adjust parameters. You do not want to come back to these programs in a month or two and struggle to understand your original thought process.

A. Investigating Boltzmann Statistics

1. *A First Meeting with the Gaussian Distribution*

The normal probability distribution may be written as:

$$f(\mu, \sigma; x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2 \right] \tag{1}$$

where μ and σ (the mean and standard deviation) are distribution parameters and x is independent variable.

1. Write a function that evaluates:

$$P(a \leq x \leq b) = \int_a^b f(\mu, \sigma; x)\, \mathrm{d}x \tag{2}$$

   where a, b, μ and σ are function parameters. Use your favorite integration algorithm.

2. What is $P(a \leq x \leq b)$?

3. Test your integrator by calculating the values of

$$P(-1\sigma \leq x - \mu \leq 1\sigma)$$
$$P(-2\sigma \leq x - \mu \leq 2\sigma)$$
$$P(-3\sigma \leq x - \mu \leq 3\sigma)$$

   You can find the correct values in Lecture Notes 1A.

4. Use the definition of FWHM and equation (2) in order to show that

$$\mathrm{FWHM} = 2\sqrt{2\ln 2}\sigma \tag{3}$$

2. The Maxwell-Boltzmann Distribution

In this challenge you may assume the gas in comprised of $N = 10^5$ H_2 molecules at a temperature of $T = 10^4$ K. You should write your code with some generality in mind, although this is not a requirement.

1. Plot the one-dimensional v_x velocity distribution for the range $v_x \in [-2.5, 2.5] \cdot 10^4$ m/s. Give a physical interpretation of the shape of the curve.

2. Integrate the velocity distribution on the range $v_x \in [5, 30] \cdot 10^3$ m/s using your favorite integration algorithm, what have you found? If you multiply your integral with N, what is the resulting number an estimate of?

3. Plot the absolute velocity distribution for the range $v \in [0, 3] \cdot 10^4$ m/s. Explain why this plot is not in conflict with your plot from question 1.

3. Elementary Statistical Physics

You *"may"* find these integrals helpful:

$$\int_0^\infty x e^{-x} \, \mathrm{d}x = 1 \tag{4}$$

$$\int_0^\infty x^{3/2} e^{-x} \, \mathrm{d}x = \frac{3}{4}\sqrt{\pi} \tag{5}$$

1. You are now going to find the average speed of a molecule in a gas by solving:

$$\langle v \rangle = \int_0^\infty v P(v) \, \mathrm{d}v \tag{6}$$

   You need to choose the correct probability distribution function.

2. The ideal gas law is a fundamental result in statistical physics and thermodynamics:

$$P = nkT \tag{7}$$

   Derive it by solving

$$P = \frac{1}{3} \int_0^\infty pv \, n(p) \, \mathrm{d}p \tag{8}$$

   where $n(p) = nP(p)$ is the number density ditribution.

3. Finally, you are going to derive the average energy of molecule in an ideal gas:

$$\langle E \rangle = \frac{3}{2} kT \tag{9}$$

   Recall that ideal gases do not have intermolecular forces and that the kinetic energy of body is $\frac{1}{2}mv^2$. **Hint:** Note that $\langle v^2 \rangle \neq \langle v \rangle^2$.

B. Simulating Energetic Gas Particles

We will begin by simulating the motion of gas particles inside a locked chamber. An illustration of such a chamber is shown in figure 1. The shape of the chamber is yours to choose, but we advise using a simple cubic box (in fact the next paragraph assumes this).

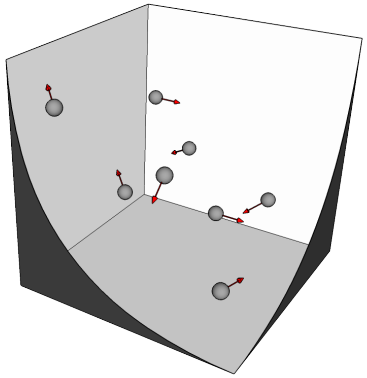

FIG. 1. An illustration of energetic particles inside a rocket engine chamber.

Let the side of the box be $L = 10^{-6}$ m and the temperature be $T = 3 \cdot 10^3$ K. Begin your simulation with $N \approx 100$ number of particles (this will be increased later). Define two arrays: one containing all N particle positions and the other containing all N particle velocities. Draw a reasonable set of initial particle positions from the assumption of uniform density.

Simulate the particles' movements using your favorite integration method. Let the simulation run for 1000 time steps with a time interval $\tau = 10^{-9}$ s. *Remember to implement particle-wall collisions!* A collision may occur at any time step, which is why you need to check for a collision at each time step.

Increase now the number of particles to $N = 10^5$, then adjust τ and the number of time steps such that the accuracy of your simulation is satisfactory. **What accuracy-criteria could you use? One of the possibilities would be one of the analytical expressions from challenge** A 3.

C. Introducing a Nozzle

Having now modelled our energetic gas, we now must turn to the nozzle. Select a side of the box and define an escape hole with an area of $0.25L^2$. How you decide to implement the escape hole is your decision, but be sure to remember that the number of particles inside the chamber must *always* remain constant. The 'refilling' from the combustion chamber can be implemented in many different ways, but after choosing a way to do it, you should discuss if you think the distribution of the gas particles will remain the same over time.

There are two important quantities that need to be accounted for: the number of escaped particles and the accumulated momentum loss from these escaped particles. Based on these quantities you will be able to derive the fuel consumption of your rocket engine (kg/s) and the thrust (propulsion) force of the rocket (N).

D. The Rocket Engine's Performance

The next step is to determine our rocket engine's performance. As explained in the lecture notes, the *actual rocket engine* is a superposition of many small rocket engines. The question now is how many small rocket engines will you need? Well, it depends... For now, make an educated guess and be prepared to change this later. *Write your code with this in mind!*

The goal of this challenge is to write a program that determines the amount of fuel your rocket will burn for an arbitrary boost Δv. The program should have a structure similar to this:

`input`	`output`
rocket thrust force (N) fuel consumption (kg/s) initial rocket mass (kg) speed boost (m/s)	fuel consumed (kg)

Remember to consider that the rocket's mass is split between its fuel and load! You do not need to take gravity into account in this challenge, you are just testing your rocket's potential. **Remember that you can extract your satellite's mass from the `AST2000SolarSystem` class (you should have looked at the documentation by now).**

Later in the project when we are going to use this program to calculate our fuel consumption during the journey. We are also going to ignore the time we spend accelerating (in this program) as it is miniscule compared to the time it takes to travel between planets.

E. Simulating a Satellite Launch

It is now time to test your rocket engine with gravity by simulating a one-dimensional launch of the rocket carrying the satellite! You may also ignore air resistance (we will tackle this one later).

Due to the difficulty of achieving a successful launch, we invite you to adjust the parameters in your engine simulation. However, there is no need to optimize the parameters - write general programs that easily allow you to come back and make simple changes.

→ You may choose any temperature up to and including $T_{\max} = 10^4$ K. Note that 3000 K is the actual burning temperature of Hydrogen: thus the closer your temperature is to 3000 K, the more realistic your simulation is.

→ You may freely adjust the size and shape of your engine chamber and nozzle.

→ You are also free to experiment with the number of particles, but note that increasing the number density is likely to increase fuel consumption.

→ You have complete control over the amount of fuel your bring with you and the number of small rocket engines that comprises the actual rocket engine.

Our definition of *being in space* is when the velocity of the rocket is equal to the escape velocity of the planet, which is when the kinetic energy equals the gravitational potential energy. Remember to account for the decreasing amount of fuel and the increasing distance between the rocket and the center of the planet.

Note that it is not sufficient to simply escape gravity whatever the cost! The launch should take between 5 and 20 minutes, and the rocket also needs to carry additional fuel needed for later parts. We obviously do not know this amount yet, so you need to make an educated guess now and write your program in a general way, so that you can come back and easily rerun the simulation later. The amount of fuel needed for later parts will generally be a lot lower than the amount needed to launch the rocket.

Once your simulated satellite hits escape velocity, you need to end the simulation and account for the following:

→ The satellite's final position.

→ The satellite's final velocity.

→ The final mass of the rocket.

→ The duration of the launch.

F. Entering the solar system

Now that you've successfully simulated the satellite launch, it is time to look at the bigger picture. Your challenge is to change your final frame of reference from the planet frame to the solar system frame. Details on your solar system can be found in the `AST2000SolarSystem` module. Be careful to include the position, velocity and rotation of your planet in your calculations.

Later in the project you will be given the option to launch your satellite at a later point in time, and from anywhere on the planet. For now you should assume your satellite launches at $t = 0$ so that you can use the initial position and velocity of your home planet from `AST2000SolarSystem`. Furthermore, you should also assume that the location of your rocket's launch site is as indicated in figure 2. **You're only going to use figure 2 in Part 1!**

What is meant by *"your final frame of reference"* is the time, position, and velocity when you are in space.

The solar system frame is expressed in Astronomical Units instead of SI units: These include AU for position, AU/yr for velocity and yr for time.

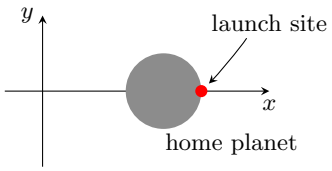

FIG. 2. The location of the rocket's launch site at $t = 0$.

VERIFICATION/CONFIRMATION

Hopefully you have now completed Part 1 of the project! As a final test of your hard work use the `engine_settings` and `mass_needed_launch` methods from the `AST2000SolarSystem` module to verify that you have found a satisfactory solution to the challenges. You can find details on how `engine_settings` and `mass_needed_launch` work in the `AST2000SolarSystem` Documentation.

FIG. 3. Motivational duck.

AST2000 - Part 1
Tips, Hints & Guiding Questions

I. IN GENERAL

It is very likely that for most of you, AST2000 is the first university level project-based course you have ever participated in. There is no denying the project is extensive and demanding; we hope you take this into consideration before you commit yourself to the project variant of this course.

Working with the project is very different from preparing for an exam. If you are struggling with Part 1, especially with respect to programming, please consider switching to the exam variant of the course. It only gets more difficult from here.

For you to get started with the project we have written the challenges in Part 1 in a slightly more detailed fashion. Our goal is for you to approach each challenge with your own perspective so as to find your own solution. There is no "one correct" way to solve a challenge. Be sure to familiarize yourself with how you should work with the project, especially with regards to your project's freedom.

One minor thing: the programs we have written to check your results are based on the same assumptions and simplifications you have been given. For example, air resistance is ignored in the rocket launch challenge. Say you wish to implement a more realistic model and include air resistance in your calculations; while it is great for your learning and understanding of physics, it will only result in your calculations being inconsistent with ours. Of course, if no assumption or simplification has been presented, you are free to approach the challenge however you like. [**?**]

II. STATISTICS AND PROBABILITY THEORY

For several students, the idea of integrating functions in probability theory is *probably* a new concept. Here are some highlights of the basics:

Probability distribution functions, in their most basic form, come as simple single-variable functions such as $f(x)$. It's important to realize that $f(x)$ **is not** "the probability of x". If you want to find a probability P you **need** to integrate it:

$$P(a,b) = \int_a^b f(x)\,\mathrm{d}x \tag{a}$$

Equation (a) should be read as *"the probability of finding x somewhere between a and b.* When it comes to probability distribution functions, you *cannot* ask the probability of finding $x =$ something. It simply doesn't exist. You can only ask for the probability of something like $a \leq x \leq b$, $x \leq 0$, etc.

Let's assume you have the probability distribution function $f(x)$. In the event you want to find the mean (average) value of something else, say $g(x)$, you then need to compute

$$\langle g(x) \rangle = \int_{-\infty}^{\infty} g(x) f(x)\,\mathrm{d}x \tag{b}$$

Here, $g(x)$ can be anything such as x^2, $1/x$ or e^x. Note the integration limits, those are important when you want to find the average value of something.

If you want to learn more about probability and statistics, check out STK1100. The course builds the mathematics of probability from its fundamental axioms.

III. MODELLING THE GAS PARTICLES

In FYS-MEK 1100 you learned how to find the position of a particle numerically using $\mathbf{F} = m\mathbf{a}$ and integrating with the Euler-Cromer scheme (or any other integration scheme for that matter). The only difference between this and simulating the rocket engine is that the rocket engine contains N particles. What this means is that at each integration step, you need to move N particles instead of just 1.

As the simulation is not continuous, a particle will never actually collide with a wall. You need to check whether the particles are inside or outside the chamber for each time step.

For tips and hints on programming the gas model (especially with regards to wall collisions), see the Python section of the Numerical Compendium. In particular, you should *definitely* have a grasp on the potential use of vectorization and masking.