ENGR 151 Project 3

Tuesday October 11th 2022

This assignment is due Thursday October 27th 2022 at 11:59PM

1 Overview

Coding... it's not rocket science! Actually, for this project, it is. We are going to calculate the payload that can be carried by a rocket to a low Earth orbit (LEO), for example to put a satellite into space. A low Earth orbit is 2000 km above the earth's surface. A rocket works by burning propellent that is ejected through a nozzle at the rear of the rocket to push it along. This is known as the thrust. The thrust pushes against the force of gravity and the air resistance as it pushes through the air. In such a case where we are including the air resistance, finding a closed-form solution (i.e. a mathematical equation) even for one-dimensional (1D) motion is generally challenging. When situations like this occur, engineers often rely on *numerical methods* to get approximate results. In this project, you will solve for the acceleration of the rocket numerically using *finite difference* methods.

2 Finite difference equations describing rocket motion

To solve the numerical equations describing the rocket motion we will use an *Euler* scheme (which is often pretty useless as a numerical method due to it's predilection for instability, but simple and works here). The origin of these equations is given in the appendix at the end of this document. You don't need to derive these, they are just there for reference for those who are interested in where they came from. The result is a set of iterations from an initial value to find the velocity v_n, height h_n, mass M_n etc. for the rocket at each time step labelled n.

3 Assignment

The assignment is to write a code to calculate the rocket trajectory using the finite different equation and optimize the fuel load for a given payload mass, broken down into 4 subtasks. For all problems, we are going to calculate and output to a precision of **3 significant figures**. Δt for all parts of this problem should be $\mathbf{\Delta t = 0.001}$ seconds. The following units should be used for your code: Time in seconds, distance in kilometers, mass in tonnes (metric tons, i.e. 1 t = 1000 kg), velocity in kilometers per second. Each task should be submitted as a separate file, with four files total for submission to the autograder.

- **Task 1:** Calculate the maximum *velocity* the rocket can achieve with no gravity or air resistance; solve

  $$v_{n+1} = v_n + \frac{T}{M_n}\Delta t \ ,$$

  and

  $$M_{n+1} = M_n - \frac{T\Delta t}{v_e} \ ,$$

  The initial velocity $v_0 = 0$ and the initial mass $M_0 = M_{total}$. You can test if your code is working by verifying that the maximum change in velocity obeys the rocket equation:

  $$M_{payload} = M_{total} e^{-\frac{v_{final} - v_{initial}}{v_e}} \ .$$

  Note $M_{payload}$ is $M_{total} - M_{fuel}$ and here $e = 2.718\ldots$ is Euler's number.

  Choose sensible values for this test; Note for the numerical integration to work, $T/M_n\Delta t$ and $T\Delta t/v_e$ need to be *small* compared with v_n and M_n respectively. For example, for Saturn V rocket, $T = 35\ tonne \cdot km/s^2$, $v_e = 4\ km/s$, $M_{total} = 3000\ tonnes$, $M_{payload} = 300\ tonnes$.

  Your file must be named `rocket_task1.cpp`. It should read in from a file named `init.txt`, which will be a file supplied by the autograder containing 4 numbers separated by end of line only, representing the input values T, v_e, M_{total} and $M_{payload}$ respectively. The code should execute returning 0 and print to `stdout` using `cout` ONLY a single number, which should be the value of the final speed v. Set the precision of `cout` by using the statement `cout.precision(3);`, which will format the output to contain three significant figures. Note that any trailing zeros after the decimal point will not be displayed, e.g. 4894.498749847 will display as "4890", but 1.20 will display as "1.2".

- **Task 2:** Add gravity and calculate the maximum *distance from Earth* the rocket can achieve. The new terms to add are shown in blue:

$$v_{n+1} = v_n + \Delta t \left(\frac{T}{M_n} - g_n \right) ,$$

$$M_{n+1} = M_n - \frac{T \Delta t}{v_e} ,$$

$$h_{n+1} = h_n + v_n \Delta t ,$$

where

$$g_n = \frac{3.962 \times 10^5}{(h_n + r_{Earth})^2} ,$$

and $r_{Earth} = 6356$ km. Note $3.962 \times 10^5 \equiv 396200$ and can be expressed as a floating point number in C++ as `3.962e5`, where here `e5` means "times 10 to the 5".

Since gravity falls off in strength with distance, if the speed of the rocket gets sufficiently high, gravity can never bring it to rest, and therefore the rocket can keep going to infinity. This is known as the *escape velocity* $v_{escape} = \sqrt{2g \times (h + r_{Earth})}$. Obviously, you don't want this to happen as this will result in an infinite loop. Therefore, be sure to add a condition in your code to detect if the velocity exceeds v_{escape}. Some test values for this task are given in table 1.

Table 1: Test values for tasks 2 and 3.

Test	T [t-km s^{-2}]	v_e [km s^{-1}]	M_{total} [t]	$M_{payload}$ [t]	h_{max} [km](Task 2)	h_{max} [km] (Task 3)
1	35	4	3000	300	3950	3920
2	35	4	3000	250	5430	5390
3	30	4.5	2000	200	8440	8330
4	30	4.5	2000	100	∞	∞

Your file must be named `rocket_task2.cpp`. It should read in from a file named `init.txt` containing 4 numbers only as in Task 1. The code should execute and if successful, the code should return 0, and print ONLY a single number to `stdout` using `cout`, which should be the value of the final height h. Set the precision of `cout` by using the statement `cout.precision(3);`. If the escape velocity is reached, the code should instead print to `stdout` "Escape velocity reached" and return 1.

- **Task 3:** Add air resistance to the model. (The new terms to add are shown in blue)

$$v_{n+1} = v_n + \Delta t \left(\frac{T}{M_n} - g_n - \frac{1}{2M_n} \rho_n C_D A v_n^2 \right) ,$$

$$M_{n+1} = M_n - \frac{T\Delta t}{v_e} ,$$

$$h_{n+1} = h_n + v_n \Delta t ,$$

where

$$g_n = \frac{3.962 \times 10^5}{(h_n + r_{Earth})^2}$$

and

$$\rho_n = 1.225 \times 10^6 \exp\left(-\frac{h_n}{9} \right) .$$

Calculate the maximum *distance from Earth* the rocket can achieve. You can assume that the dimensionless coefficient $C_D = 0.500$ and the cross sectional area A is circular with diameter 6.6×10^{-3} km for this rocket. Some test values for this task are given in table 1.

Your file must be named `rocket_task3.cpp`. It should read in from a file named `init.txt` containing 4 numbers only as in Task 1. The code should execute and, if successful, should return 0 and print ONLY a single number, which should be the value of the final height h, to `stdout` using `cout`. Set the precision of `cout` by using the statement `cout.precision(3);`. If the escape velocity is reached, the code should instead print to `stdout` "Escape velocity reached" and return 1.

- **Task 4:** Find the optimal fuel mass for given thrust and payload to reach a height of 2000 km. The optimum is when the rocket is brought to rest at a height of 2000 km at the point when the fuel mass tends to zero. Use **any technique** to find this optimum to 0.2% accuracy (i.e. the fuel mass that just gets the rocket to 2000 ± 5 km). However, the calculation must complete within 10 seconds to pass the test. You can assume again that the dimensionless coefficient $C_D = 0.500$ and the cross sectional area A is circular with diameter of 6.6×10^{-3} km.

You may search from a minimum fuel mass of zero up to the maximum fuel mass allowed by the equation for the velocity update:

$$v_{n+1} = v_n + \Delta t \left(\frac{T}{M_n} - g_n - \frac{1}{2M_n} \rho_n C_D A v_n^2 \right) ,$$

for which $v_{n+1} > v_n$ on the first step or the rocket won't take off.

Your file must be named `rocket_task4.cpp`. It should read in from a file named `init_opt.txt` containing 3 numbers only, representing the input values T, v_e and

$M_{payload}$. The code should execute and, if successful, it should return 0 and print ONLY a single number, which should be the value of the optimal *fuel* mass M_{fuel}, to `stdout` using `cout`. Set the precision of `cout` by using the statement `cout.precision(3);`. On error, if the total mass is too heavy, it should instead return 1 and should print to `stdout` "Rocket too heavy". (You should still test for reaching escape velocity within your search, to avoid an infinite loop, but it should not terminate the code).

4 Grading

The project should be submitted to project 3 on autograder.io. This project will be graded in two parts. First, the autograder will evaluate your submission and provide a maximum score of 100 points. Second, one of our graders will evaluate your submission for style and commenting, and will subtract 0-10 points from the score that autograder evaluated. The following is the breakdown for each part:

# pts	Description
25	Task 1: Velocity correctly calculated
25	Task 2: Height correctly calculated
25	Task 3: Height correctly calculated
25	Task 4: Mass correctly calculated

# pts	Description (pts will be subtracted if item is missing or insufficient)
2	Each file has name/section number/submitted date included in a header comment
2	Comments are used appropriately (e.g. major steps explained)
2	Indenting and whitespace are appropriate (including functions properly formatted)
2	Variables are given descriptive names
2	Overall program structure (e.g, using functions instead of repeated code blocks)

5 Appendix: Physical Equations describing Rocket Motion

We will approximate this situation using a very simple one-dimensional (1D) model; in other words, considering the rocket to be confined to travelling directly upwards. In reality a rocket would have to manoeuvre into an LEO, it may have multiple stages and all sorts of other complications that we will ignore for simplicity. The rocket motion is described by the force balance:

$$F_{rocket} = T - M_{rocket} \times g - R_{air} \ ,$$

where F_{rocket} is the resultant force on the rocket, T is the thrust generated by burning the propellant, $M_{rocket} \times g$ is the gravitational force (weight) on the rocket with mass M_{rocket}, and R_{air} is the air resistance. We will consider T to be a constant, i.e. the rocket generates constant thrust.

There are several complications to consider. The first is that due to the fact the rocket is traveling far from the Earth's surface so we can't use $g = 9.807$ ms^{-2}, but instead need to use

$$g = \frac{GM_E}{r_{rocket}^2} = \frac{3.962 \times 10^5}{\{(h_{rocket} + r_{Earth})\ [\text{km}])\}^2}\ \text{km}\,\text{s}^{-2}\ ,$$

where h_{rocket} is the height of the rocket in kilometers as measured from the Earth's surface, which is at radius $r_{Earth} = 6356$ km. (G is the gravitational constant and M_E is the Earth's mass. Note this expression is only accurate to four significant figures.)

We can calculate the height of the rocket by integrating the equation

$$\frac{dh_{rocket}}{dt} = v_{rocket}\ ,$$

where v is the rocket velocity. We can calculate the velocity of the rocket using

$$\frac{dv_{rocket}}{dt} = \frac{F_{rocket}}{M_{rocket}}\ .$$

The next complication comes from the fact that the rocket thrust is generated by burning fuel, so *the rocket mass changes as a function of time.* Hence we need to take into account the "rate of change of mass", i.e. how much fuel is expelled out of the back of the rocket per second. The rocket thrust can be defined as

$$T = -v_e \frac{dM_{rocket}}{dt}\ ,$$

where v_e is the exhaust velocity - i.e. how fast the propellant is ejected from the nozzle. We will assume that this quantity is a constant, so the rate of mass loss is also a constant. Figure 1 is a cartoon (courtesy of Wikipedia) which illustrates how the mass loss leads to thrust.

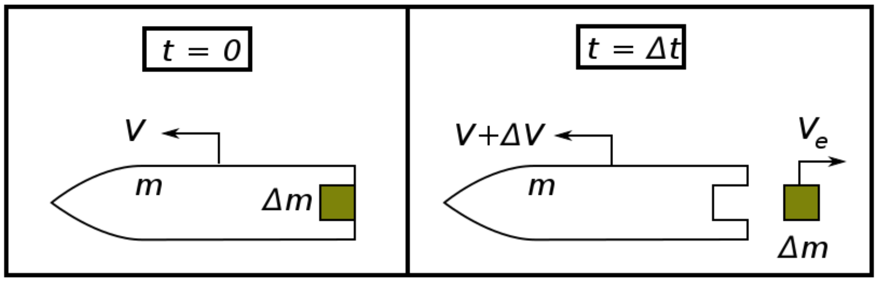

Figure 1: Cartoon (courtesy of Wikipedia) illustrating how the ejection of a small mass of propellant leads to thrust for a rocket.

Finally, we may add air resistance, which depends on the rocket velocity, through the expression $R_{air} = \frac{1}{2}\rho C_D A v_{rocket}^2$, with A the cross sectional area, ρ the air mass density and C_D the drag coefficient. In general the drag coefficient C_D depends on the shape and dimensions of the rocket and has some speed dependence at low velocity, but we will assume it is a constant.

Lastly, of course the air density varies with altitude; the higher you are, the thinner the air is. We will use a very simple model to describe the air density as a function of height:

$$\rho = 1.225 \times 10^6 \exp\left(-\frac{h_{rocket}\ [\text{km}]}{9.000}\right)\ \text{tonnes}\,\text{km}^{-3}\ .$$

Hence, all together we solve:

$$\frac{d}{dt}v_{rocket}(t) = \frac{T}{M_{rocket}(t)} - g(t) - \frac{1}{2M_{rocket}(t)}\rho(h_{rocket}(t))C_D A v_{rocket}(t)^2 \ ,$$

combined with

$$\frac{d}{dt}h_{rocket}(t) = v_{rocket}(t) \ ,$$

and

$$\frac{d}{dt}M_{rocket}(t) = -\frac{T}{v_e} \ ,$$

where (t) next to a quantity indicates that it is a function of time. We can approximate the derivatives by *finite difference* equations, e.g. using:

$$\frac{dv}{dt} \approx \frac{v(t+\Delta t) - v(t)}{\Delta t} \ .$$

Δt is a constant finite sized step in time, so that after n time steps the time is $t = n\Delta t$. Since $v(t+\Delta t) = v(n\Delta t + \Delta t) = v([n+1]\Delta t)$, we make use of the notation $v(t+\Delta t) \equiv v_{n+1}$. $v(t) \equiv v_n$ etc. Using the Euler scheme, the equations of motion in *finite difference* form (dropping the $_{rocket}$ suffix for clarity) become:

$$\frac{v_{n+1} - v_n}{\Delta t} = \frac{T}{M_n} - g_n - \frac{1}{2M_n}\rho_n C_D A v_n^2 \ ,$$

$$\frac{h_{n+1} - h_n}{\Delta t} = v_n \ ,$$

and

$$\frac{M_{n+1} - M_n}{\Delta t} = -\frac{T}{v_e} \ ,$$

with

$$g_n = \frac{3.962 \times 10^5}{(h_n + r_{Earth})^2}$$

and

$$\rho_n = 1.225 \exp\left(-\frac{h_n}{9}\right) \ .$$

or, after rearranging:

$$\boxed{\begin{gathered} v_{n+1} = v_n + \Delta t\left(\frac{T}{M_n} - g_n - \frac{1}{2M_n}\rho_n C_D A v_n^2\right) \ , \\ h_{n+1} = h_n + v_n \Delta t \ , \\ \text{and} \\ M_{n+1} = M_n - \frac{T\Delta t}{v_e} \ , \end{gathered}}$$