Team Name: SCAM

Members:

Matthew Amicaterra, Cameron Eure, Stefan Glende, Andrew Schneider

ME 1311

Dr. Ruhala

Introduction

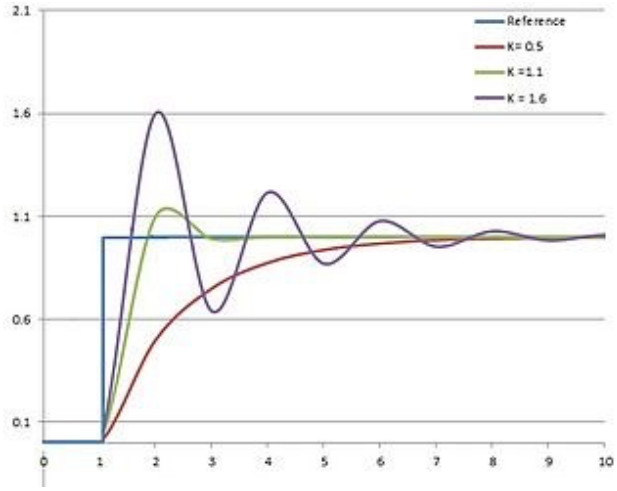
Written Report

A space program has asked us to come up with a way to ensure their new rover can land on Planet X. With math techniques and MATLAB, we can create a program that uses control theory to do this safely and reliably. While these calculations are not holistic (things such as air resistance and changing mass due to burning fuel are not considered), they provide a good estimation for what the rover will need to land safely. We also limited the planets to Mercury, Earth, Venus, and Mars. The other planets in the solar system are known as “gas giants” and do not have solid surfaces to land on. Any other planet can be added but is not an option by default. Additionally, there is an option to input data on a custom planet for user flexibility

Landing a rover requires a few things to be aware of. If it is falling too quickly when it encounters the ground, the rover will crash and not be able to complete its mission. If it takes too long to reach the ground, then the rover will burn too much fuel and not be able to finish its mission. Our group’s plan is to write a program that will determine the best constants for a PID (Proportional-Integral-Derivative) loop, which will help land the rover safely without burning too much fuel.

Optimally as the rover approaches the ground, the amount of force from thrust will increase such that it will come to a near stop before it reaches the ground. Exact values for thrust with respect to time can be found using the PID loop. While most of these are easily determinable through Riemann sums or linear approximation, depending on the gravitational force of planet X, the “correct” constants for the PID loop could vary significantly. Our program would simulate what could happen to the rover based on the

gravitational force of the planet, the properties of the rover, and various values for the constants.



Equations

1. $\mathrm{output(t)} = \mathrm{K_p e(t)} + \mathrm{K}_i \int_0^t e(t)\,dt + K_d \frac{de}{dt}$
    a. Classic independent PID equation
    b. This equation is the backbone of the m-file, as it will be doing much of the calculation. The equation is being looped as a live calculation in the sense of a simulation
    c. $\mathrm{K_p e(t)}$ is the proportional term for the PID Loop.
    d. $\mathrm{K}_i \int_0^t e(t)\,dt$ is the integral term for the PID Loop. The integral uses previously created data in an effort to help predict actions.
    e. $K_d \frac{de}{dt}$ is the Differential term for the PID Loop.
2. $\mathrm{thrust}_{max} = 5 * \mathrm{weight}$
    a. Predefined maximum thrust the boosters can exert based on a ratio of thrust to weight.
    b. This was calculated as the limit for the maximum thrust.
3. $a = \frac{thrust - weight}{mass}$
    a. The equation of acceleration given the variables in this system.
4. $v_t = v_{t-1} + at$
    a. Traditional kinematic equation
5. $\Delta h = \frac{v_t + (v_{t-1})}{2} * t$
    a. Traditional kinematic equation

Parameters

- keyset
  - Cell array value of user's planet selection
- planet
  - String conversion of keyset
- gravity
  - Either the selected planet's surface acceleration due to gravity, or, if the user selected custom, the manual input of a custom planet's surface acceleration, in m/s^2
- drop_height
  - User input of the value of the distance of the rover to the landing site, in meters
- rover_mass
  - User input of the value of the mass of the rover, in kilograms.
- mintime
  - The amount of time it took the rover to successfully land.
- bestv
  - The rover's velocity when it lands.
- bestkp
  - The value of kp for the best run
- bestki
  - The value of ki for the best run
- bestkd
  - The value of kd for the best run
- bestthrust
  - The thrust vector when the rover lands

Operations

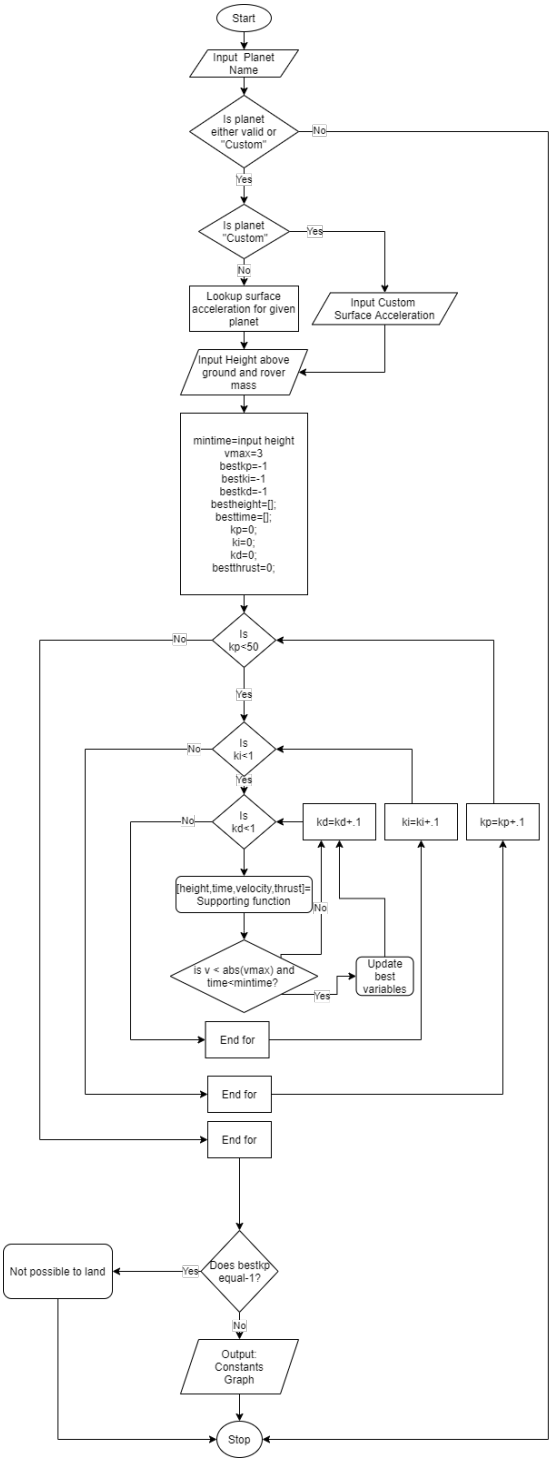
Our code for this project has two main parts: the main script that handles the inputs and creating the plot along with a supporting function that does the computational work.

The main script first asks the user what planet they are simulating around. If the planet is in the “known planet” database, then it uses the known value of gravity. If it does not know it asks the user for the gravity in meters per second. The script then asks for the initial height and the mass of the rover. It then loops through a multitude of possible combinations for kp, ki, and kd and ultimately determine which ones will result in the rover landing in the least time while being within the “safe landing” velocity. After doing this, it displays the time it will take to land, the final velocity, and a graph of plot of height over time and thrust over time.

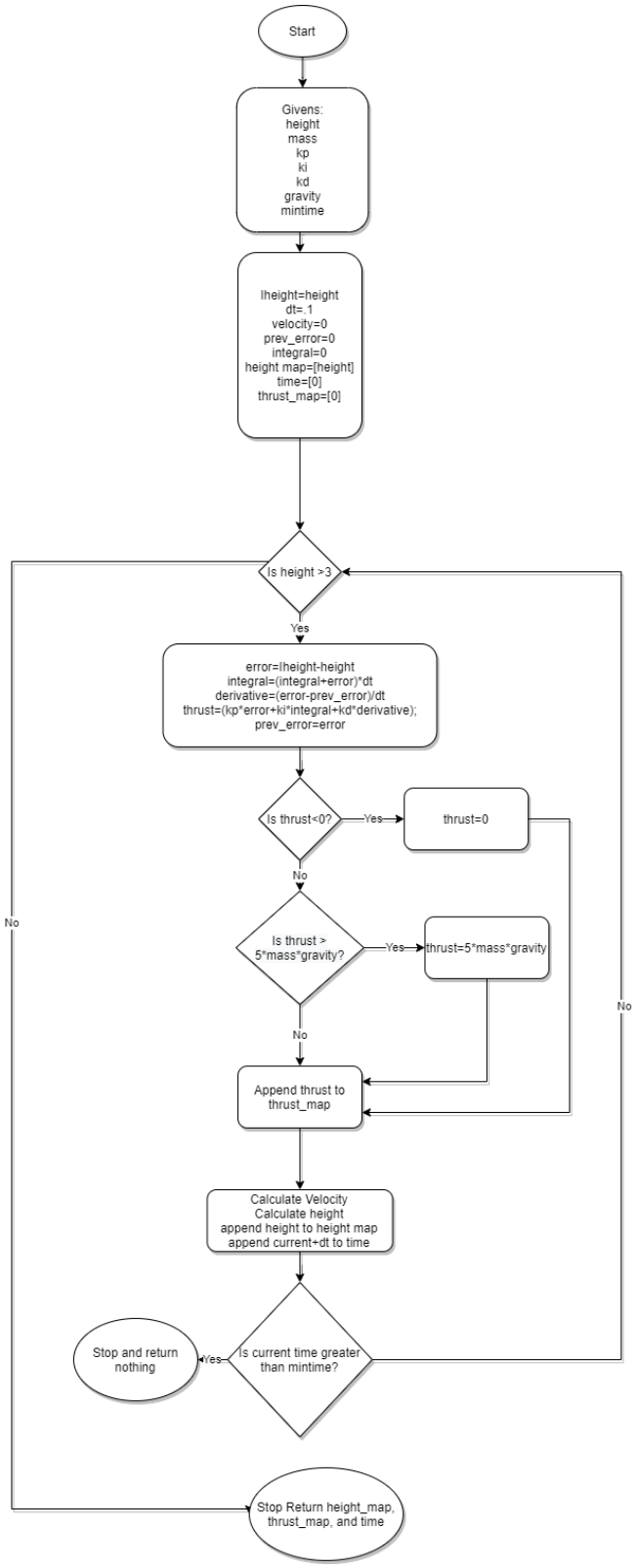
The function takes inputs of kp, ki, kd, the mass of the rover, the height of the rover, a max time, and outputs a time vector, a height vector, a thrust vector, and the final velocity. To get these values this code first does a PID calculation to determine the thrust the boosters should apply. If this number is negative, it sets thrust to zero as the thrusters cannot exert a downwards force. If the number is greater than 5 times the weight of the rover, it sets the thrust to 5 times the weight of the rover as that is the predefined “maximum” thrust the boosters can apply. After determining this number, the code changes the height of the rover based on the translation caused by the acceleration of the boosters. It repeats this process until the rover is less than or equal to 3 feet from the ground in which the program ends and returns the height, time, and velocity values.

Flowchart

Main Script:



Supporting function:



Script & functions

mfile_SCAM.m

```
% mfile_SCAM.m
% May 3, 2021
clear, clc, format compact, close all

Names = {'Mercury','Venus','Earth','Mars'}; % Creates an cell array of
terrestrial planet names
SurfAccel = [3.7 8.87 9.807 3.721]; % creates an array of those planets
furface gravitation acceleration in m/s^2
M = containers.Map(Names,SurfAccel); % creates a searchable map used to match
SurfAccel to a selected planet


disp('Please input either one of the four terrestrial planets:')
disp('''Mercury, ''Venus'', ''Earth'', or ''Mars''')
disp('Optionally, input ''Custom'' to choose your own gravitational surface
acceleration')
keyset={input('Planet: ','s')}; % User enters the planet being landed on

planet=cell2mat(keyset); % converts the entered planet to a string type

if isKey(M,keyset)==0 & strcmp(planet,'Custom')==0
   disp('Error, invalid command, run the program again and check your input')
   return
end

if strcmp(planet,'Custom')==0 % checks to see if the custom option is being
used
valueset=values(M,keyset); % since custom is not used, looks up the selected
planet's surface acceleration
gravity= cell2mat(valueset); % converts the surface acceleration from a cell
type to a number type

else
    gravity=input('Please input your custom surface acceleration: '); % User
has chosen custom and must enter their custom surface acceleration
end

drop_height=input('Please input the distance to the ground in meters: '); %
User/Rover enters height from which it is falling
rover_mass=input('Please input the mass of the rover in kg: '); % Generalized
so that this program can theoretically be used for multiple rovers


mintime=drop_height;
vmax=3; %arbitrary crashing velocity we state is percent error
bestkp=-1; %Starting value for best kp, will be changed later
bestki=-1; %Starting value for best ki, will be changed later
bestkd=-1; %Starting value for best kd, will be changed later
bestheight=[]; %Instantiates the bestheight vector
besttime=[];   %Instantiates the besttime vector
bestv=0;   %Starting value for best kp, will be changed later
bestthrust=[];
for kp = .1:.1:50  %loop through .1 to 50 values of kp
```

```
    %fprintf("Running kp=%f\n",kp)  %error handling
    for ki= .1:.1:1   %loop through ki
        for kd= .1:.1:1  %loop through kd

[height,time,velocity,thrust]=RunPIDController(drop_height,kp,ki,kd,gravity,r
over_mass,mintime); %run the PID loop
            if((abs(velocity)) < vmax) && (time(length(time)) < mintime)  %if
velocity is within margin and the time is faster
                mintime=time(end);   %update the values
                bestv=velocity;
                bestkp=kp;
                bestki=ki;
                bestkd=kd;
                bestheight=height;
                besttime=time;
                bestthrust=thrust;
            end
        end
    end
end



disp(' ') % Proper Spacing

if bestkp == -1
    disp('Rover has failed to land')
else
    disp('The final time, height, and graph are displayed.')
    fprintf('The rover has taken a time of %.f seconds to land successfully.
\n', mintime)
    fprintf('The rover is coming in at a velocity %.f m/s. \n', bestv)
    figure(1)
    plot(besttime,bestheight, 'b-',besttime,bestthrust,'red-')
    title(sprintf('Height vs Time of Rocket kp=%.1f, ki=%.1f,
kd=%.1f',bestkp,bestki,bestkd))
    xlabel('Time [s]')
    ylabel('Height[m] / Thrust [N]')
    grid on
    legend('Height','Thrust')


end % The time taken to land successfully.


disp(' ')

disp('Program has ended') %Program has ended

disp('To repeat or input a new planet, press ''Run'''')
```

runPIDcontroller.m

```
function [height_map,time,velocity,thrust_map] =
RunPIDController(height,kp,ki,kd,gravity,mass,mintime)
%Simulates a PID and has a exit case
%   Simulates what a PID loop given kp,ki,kd and the gravity on the planet
%   Assumes the following:
%       Rocket engine can only be updated 10 times per second
%       No exteral forces (wind, air resistance, etc)
%       Stops once the rocket hits the ground or the runtime is too great
%       Source http://cires1.colorado.edu/jimenez/CHEM-
5181/Lect/LV_Class_2_PID.pdf
Iheight=height;  %stores the initial height
dt=.1; %sets the "step" between thrust changes
velocity=0;  %sets inital v=0
prev_error=0;  %set prev error to zero
integral=0;  %sets integral from 0 to 0 to 0
height_map=[height]; %starting at intitial height
time=[0];  %starting at time=0
thrust_map=[0];
while(height>3)
    error=Iheight-height;  %proportional of PID
    integral=(integral+error)*dt; %integral of the PID
    derivative=(error-prev_error)/dt;  %derivative of the PID
    thrust=(kp*error+ki*integral+kd*derivative);   %does the PID math
    prev_error=error; %updates previous error
    if(thrust<0) %You can thrust negatively so this prevents that
        thrust=0;
    elseif(thrust>5*mass*gravity) %limit of maxinum thrust is 2 times the
weight of the rover
        thrust=5*mass*gravity;
    end
    thrust_map=[thrust_map,thrust];
    acceleration=(thrust-mass*gravity)/mass;  %a=(thrust-weight)/mass
    Oldv=velocity;
    velocity=velocity+acceleration*dt;  %for small changes in time,
velocity=previous velocity+ acceleration*time
    height=height+((Oldv+velocity)/2*dt);  %change the height based on v
    height_map=[height_map,height];  %update height map
    time=[time,time(end)+dt]; %update time
    if(time(end)>=mintime)  %pseudo alpha beta pruning to decrease runtime,
Basicly just saying "if this run is longer than the current best, give up
        break
    end
end
end
```

Output Results:

Mercury example:

Please input either one of the four terrestrial planets:

'Mercury, 'Venus', 'Earth', or 'Mars'

Optionally, input 'Custom' to choose your own gravitational surface acceleration

Planet: Mercury

Please input the distance to the ground in meters: 1000

Please input the mass of the rover in kg: 1100

The final time, height, and graph are displayed.

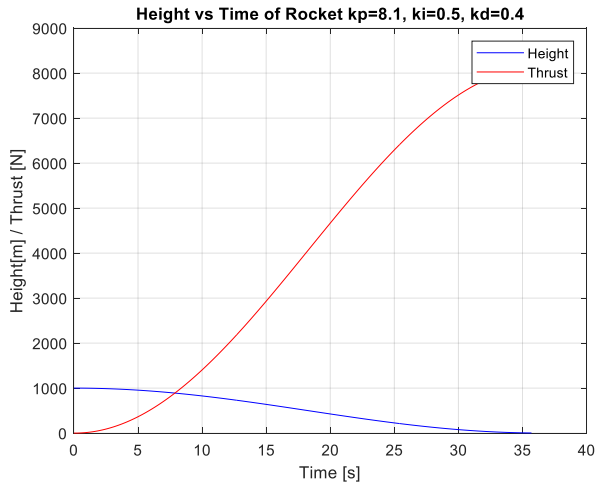
The rover has taken a time of 36 seconds to land successfully.

The rover is coming in at a velocity -3 m/s.

Program has ended

To repeat or input a new planet, press 'Run'

>>



Venus example:

Please input either one of the four terrestrial planets:

'Mercury, 'Venus', 'Earth', or 'Mars'

Optionally, input 'Custom' to choose your own gravitational surface acceleration

Planet: Venus

Please input the distance to the ground in meters: 1000

Please input the mass of the rover in kg: 1100

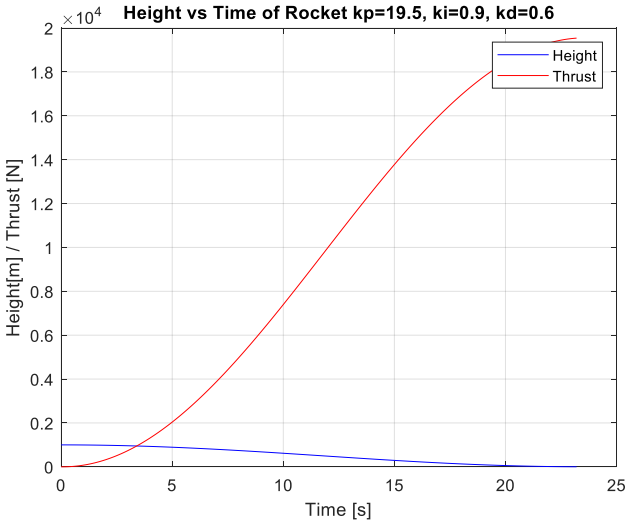
The final time, height, and graph are displayed.

The rover has taken a time of 23 seconds to land successfully.

The rover is coming in at a velocity -3 m/s.

Program has ended

To repeat or input a new planet, press 'Run'



Earth example:

Please input either one of the four terrestrial planets:

'Mercury, 'Venus', 'Earth', or 'Mars'

Optionally, input 'Custom' to choose your own gravitational surface acceleration

Planet: Earth

Please input the distance to the ground in meters: 1000

Please input the mass of the rover in kg: 1100

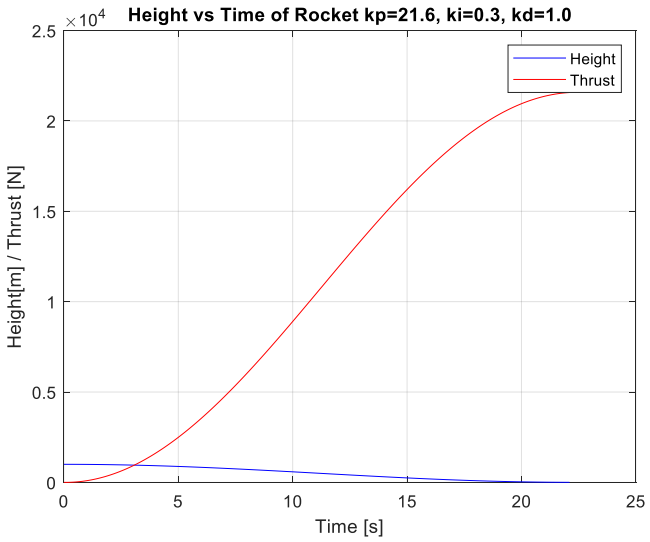
The final time, height, and graph are displayed.

The rover has taken a time of 22 seconds to land successfully.

The rover is coming in at a velocity -3 m/s.

Program has ended

To repeat or input a new planet, press 'Run'



Mars example:

Please input either one of the four terrestrial planets:

'Mercury, 'Venus', 'Earth', or 'Mars'

Optionally, input 'Custom' to choose your own gravitational surface acceleration

Planet: Mars

Please input the distance to the ground in meters: 1000

Please input the mass of the rover in kg: 1100

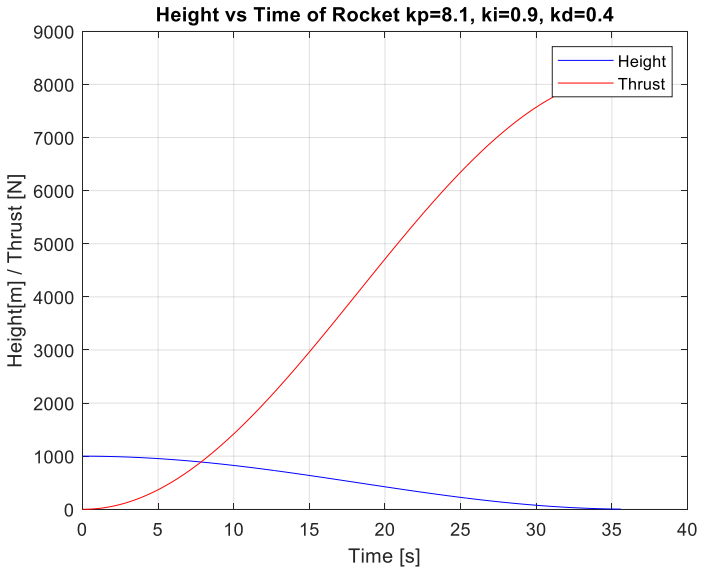
The final time, height, and graph are displayed.

The rover has taken a time of 36 seconds to land successfully.

The rover is coming in at a velocity -3 m/s.

Program has ended

To repeat or input a new planet, press 'Run



Custom Example:

Please input either one of the four terrestrial planets:

'Mercury, 'Venus', 'Earth', or 'Mars'

Optionally, input 'Custom' to choose your own gravitational surface acceleration

Planet: Custom

Please input your custom surface acceleration: 1.8

Please input the distance to the ground in meters: 1000

Please input the mass of the rover in kg: 1100

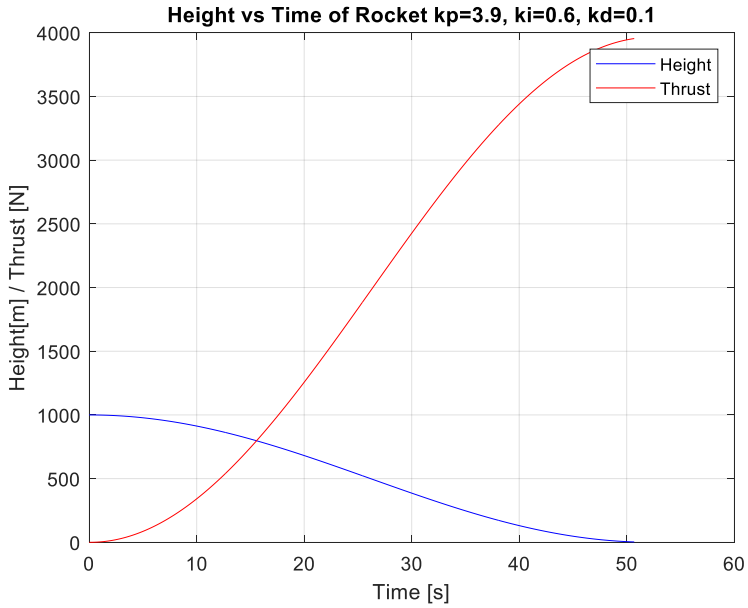
The final time, height, and graph are displayed.

The rover has taken a time of 51 seconds to land successfully.

The rover is coming in at a velocity -3 m/s.

Program has ended

To repeat or input a new planet, press 'Run'


Height vs Time of Rocket kp=3.9, ki=0.6, kd=0.1
Height
Thrust
Height[m] / Thrust [N]
Time [s]
0
500
1000
1500
2000
2500
3000
3500
4000
0
10
20
30
40
50
60

Bibliography

D. David, “Intro to PID Control,” *Cooperative Institute for Research in Environmental Sciences at the University of Colorado Boulder*, 2013. [Online]. Available: http://cires1.colorado.edu/jimenez/CHEM-5181/Lect/LV_Class_2_PID.pdf .

J. Yoon, “Ask Us - Gravity on Other Planets,” *Aerospaceweb.org | Ask Us - Gravity on Other Planets*, 08-May-2005. [Online]. Available: http://www.aerospaceweb.org/question/astronomy/q0227.shtml.

“PID Controller Response”. November 16, 2020. Photograph. ElProCus. https://www.elprocus.com/the-working-of-a-pid-controller/.