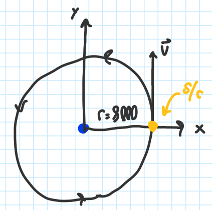
HW7 Astrodynamics | Ramon Perlstein 0101

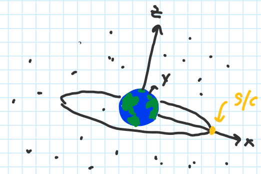
Monday, May 6, 2024 9:43 PM

Q1 :

given :

$$s/c_{pos_{init}} = \begin{bmatrix} 8000 \\ 0 \\ 0 \end{bmatrix} km$$





$\mu_{Earth} = 0.39860 \cdot 10^6$

$$\|\vec{v}\| = \sqrt{\frac{\mu_E}{r_{circ}}} = \sqrt{\frac{0.39860 \cdot 10^6}{8000}} = \underline{\underline{7.0587 \ km/s}}$$

- $T/m = 1 \cdot 10^{-4} \ kN/kg$

a) $$\boxed{s/c_{vel_{init}} = \begin{bmatrix} 0 \\ 7.0587 \\ 0 \end{bmatrix} km/s}$$

b) $F = ma$, $\frac{F}{m} = a$

$$\vec{a} = \underbrace{\frac{-\mu_E}{\|\vec{r}\|^3}}_{\text{Force o' Gravity}} \cdot \vec{r} + \frac{T}{m} \underbrace{\frac{\vec{v}}{\|\vec{v}\|}}_{\substack{\text{direction} \\ \text{of vel}}}$$

also from lecture:

$$a_T = \frac{1}{2} \underbrace{\frac{d(v^2)}{ds}}_{2v \frac{dv}{ds} \to \frac{dv}{dt}} + \frac{\mu_E}{r^2} \underbrace{\frac{dr}{ds}}_{\sin\gamma}$$ (acceleration due to thrust)

unusable in our problem because not fully in terms of t (not all $\frac{d}{dt}$)

c) Given the equation:

$$t_{esc} - \underset{\downarrow 0}{t_0} = \frac{v_0}{a_T}\left(1 - \left(\frac{20 \, a_T^2 \, r_0^2}{v_0^4}\right)^{1/8}\right)$$

recall: $\vec{a} = \frac{\mu_E}{\|\vec{r}\|^3}\vec{r} + \frac{T}{m}\frac{\vec{v}}{\|\vec{v}\|}$

a_T = acceleration due to thrust, so

$a_T = 1 \cdot 10^{-4} \ kN/kg$

Then:

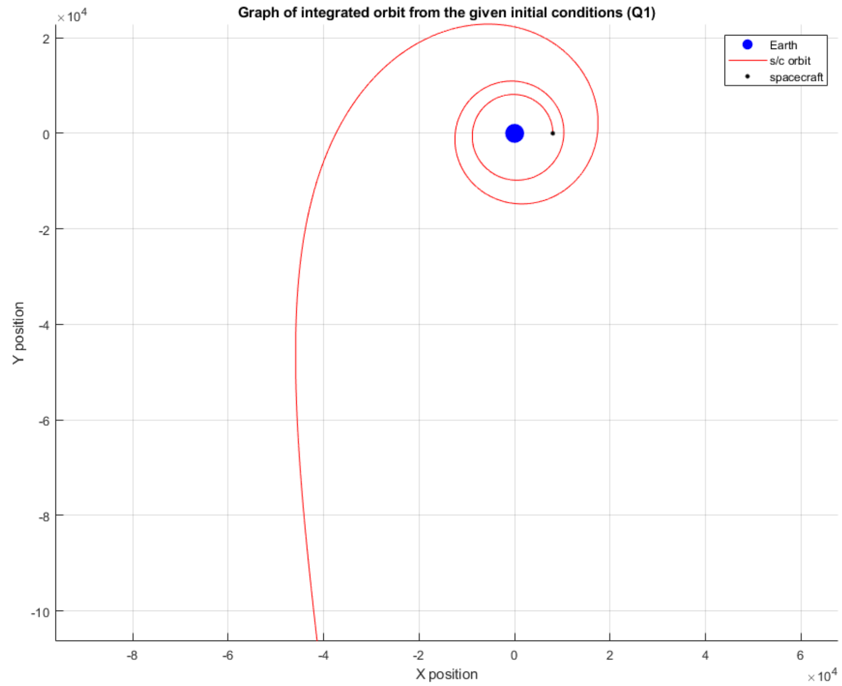
$$t_{esc} = \frac{\|s/c_{vel_{init}}\|}{a_T}\left(1 - \left(\frac{20 \, a_T^2 \, \|s/c_{pos_{init}}\|^2}{\|s/c_{vel_{init}}\|^4}\right)^{1/8}\right)$$

$$= \frac{7.0587}{0.0001}\left(1 - \left(\frac{20(0.0001)^2(8000)^2}{(7.0587)^4}\right)^{1/8}\right)$$

$$= 3.4047286187 \cdot 10^4 \text{ seconds}$$

$$t_{esc} = \boxed{34047 \text{ seconds}}$$

d)



e) given that r_{esc} can be found

$$r_{esc} = \frac{\| s/c_{pos\ init} \| \| s/c_{vel\ init} \|}{(20\, a_T^2 \| s/c_{pos\ init} \|^2)^{1/4}}$$

$$r_{esc} = \frac{(8000)(7.0587)}{(20(0.0001)^2(8000)^2)^{1/4}}$$

$$\boxed{r_{esc} = 29855 \text{ km}}$$

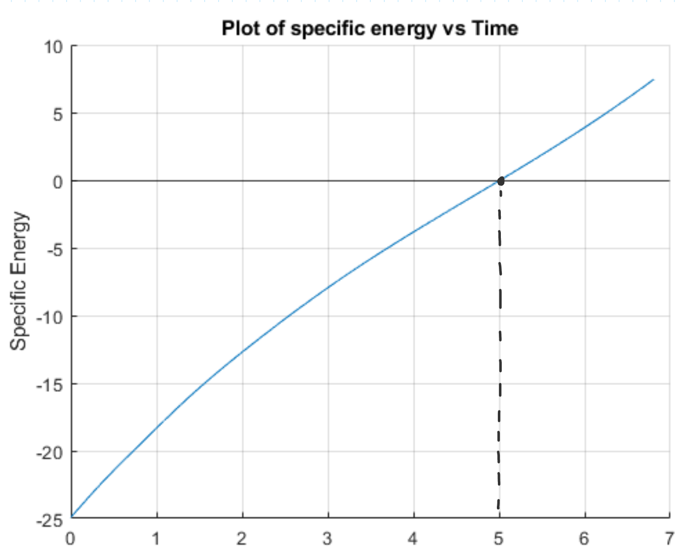
from inspection of integrated data, r_{esc} is reached at 36413 seconds

different from our analytical 34047 seconds by 2366 seconds!

a little less than 1 hour

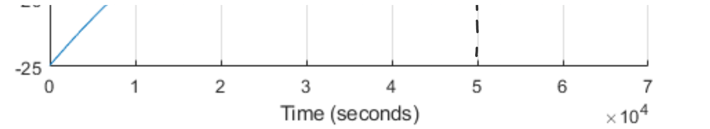
- However, inspection of the change in energy sign, i.e. when $\mathcal{E}$ (specific energy) goes from negative to positive, occurs roughly at 50130 seconds!

Using $\mathcal{E} = \frac{v^2}{2} - \frac{\mu}{r} = \frac{\|\vec{v}\|^2}{2} - \frac{\mu}{\|\vec{r}\|}$

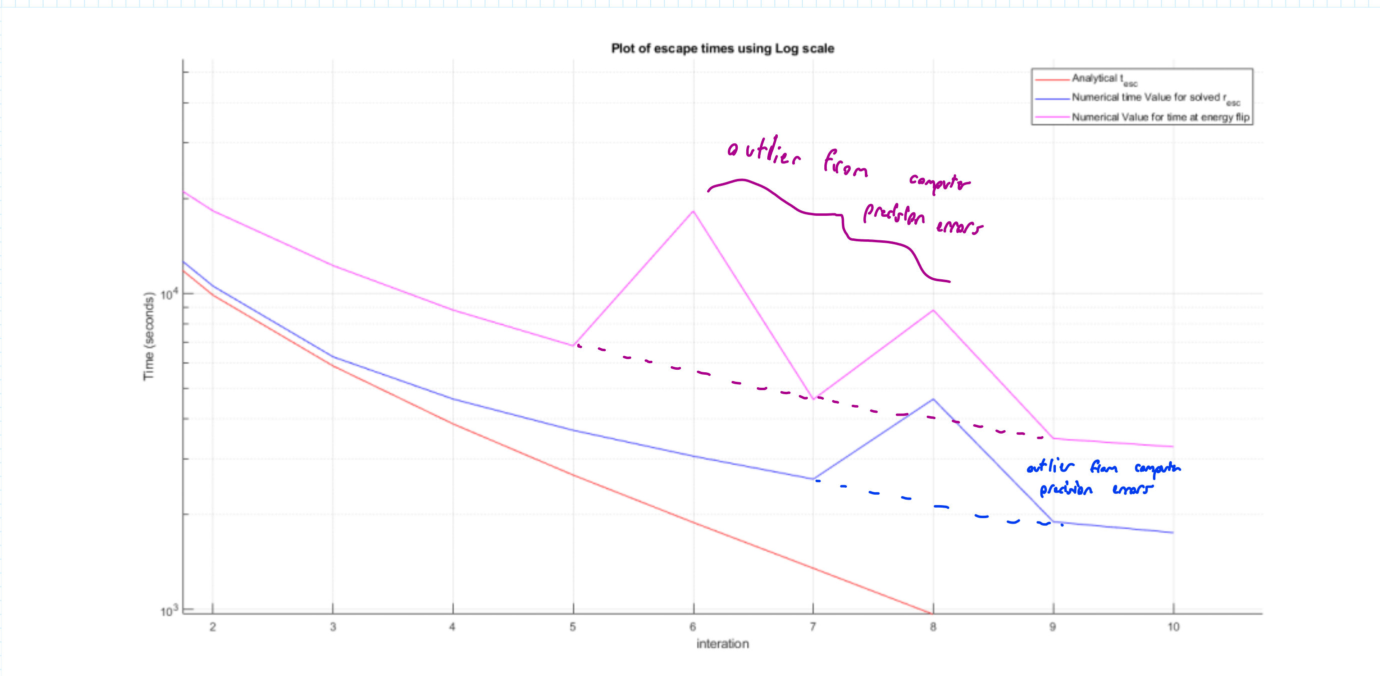


- Both numerically found escape times are greater than the analytical answer. I think the analytical answer is conservative with it's output because it really doesn't take into account the force of gravity on the spacecraft the same way that the numerical integrated does. Since it's only going off of

output because it really doesn't take into account the force of gravity on the spacecraft the same way that the numerical integrated does. Since it's only going off of the kinematics of the problem (other than the acceleration due to thrust being a non-kinematic term since it is a force), it does not account for the extra time it might take for the spacecraft to break from the planets gravity well, giving us a conservative value. TL:DR - The exclusion of the gravity term from the t_esc equation leads to a faster escape time being found. Since the t_esc equation is only influenced by a single force (and not the accounting for the gravitiational force being applied to the craft), it will find a faster
escape time.



f)



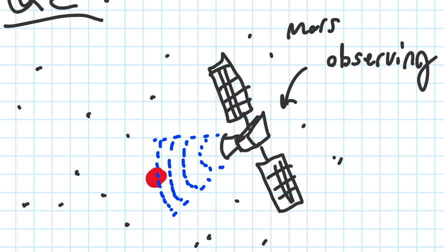
See attached code for more detail

g) From the plot, it is clear that the analytical method is lacking in it's accuracy. As the acceleration due to thrust is increased, we see a larger separation of numerical vs analytical values of the escape time. However, there are two key things to note: the analytical method produces a smooth, continuous curve of solutions, because it gives you exactly one solution. The integration method requires that you find the value at which either the energy changes or the time at which you reach your analytically solved for r_esc. Disregarding the outliers of data in the provided plot, and assuming they are smooth curves, we can still see the growing gap between the analytical solution and the numerical integration soltuion. The main limitation, I believe of the analytical method is that it does not incorporate gravity into the equation at all, and only relies on the input thrust as it's only force. I think this limitation is what causes the analytical method to undershoot, as it requires less time to escape if there is no modeled force of gravity "hold you back" (resisting the spacecrafts power to escape orbit)

HW7 Astrodynamics | Romeo Perlstein, section 0101

Thursday, May 9, 2024 3:02 AM ← peep the horror!

Q2:



Given:

- $r_p = 1000$ km
- $e = 0.25$
- $r_p = a(1-e)$
- $a = \frac{r_p}{(1-e)} = \frac{1000}{(1-e)}$

  $= 1333.3333$ km
- $r_a = a(1+e) = 1666.6667$ km
- $I_{sp} = 250$ sec
- $m_{initial} = 1500$ kg

Using the approximation:

$$\Delta V = I_{sp}\, g_0 \ln\left(\frac{m_i}{m_f}\right)$$

rearranging, we get:

- $\frac{m_i}{m_f} = e^{\left(\frac{\Delta V}{I_{sp} g_0}\right)}$

or

- $\frac{\Delta m}{m_i} = 1 - e^{\left(\frac{-\Delta V}{I_{sp} g_0}\right)}$

must convert g_0 from m/s^2 to km/s^2 since ΔV is km/s

$$e^{\left(\frac{\Delta V}{I_{sp} g_0}\right)} = \frac{0.7725}{(250)(0.00981)}$$

$$\frac{m_i}{m_f} = 1.3702$$

$$m_f = \frac{m_i}{1.3702} = \underline{\underline{1094.7 \text{ kg}}}$$

or

$$\Delta m = m_i \left(1 - e^{\left(\frac{-\Delta V}{I_{sp} g_0}\right)}\right)$$

$$\Delta m = 1500 \left(1 - e^{\left(\frac{-(0.7725)}{(250)(0.00981)}\right)}\right)$$

$$\Delta m = \underline{\underline{405.2781 \text{ kg}}}$$

- first, find ΔV to circularize

$$V_{peri} = \sqrt{\frac{2\mu}{r_p} - \frac{2\mu}{r_p + r_a}}$$

$$\mu_{mars} = 0.042828 \cdot 10^6$$

$$V_{peri} = \sqrt{\frac{2(0.042828 \cdot 10^6)}{1000} - \frac{2(0.042828 \cdot 10^6)}{1000 + 1666.6667}}$$

$$= 7.3168 \text{ km/s}$$

$$V_{circ} = \sqrt{\frac{\mu}{r_p}} = 6.5443 \text{ km/s}$$

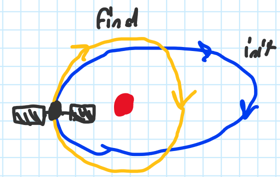
$\boxed{\text{• So, 405.2781 kg of propellant mass was used during the circularization maneuver!}}$

since we are going from eclipse to circular, s/c needs to slow down!

$$\Delta V = |V_{circ} - V_{peri}|$$

$$= |6.5443 - 7.3168|$$

$$= 0.7725 \text{ km/s}$$ ← total ΔV needed to circularize at 1000 km



• Part 2 type thing:

given:

- structural ratio $\epsilon = \frac{m_E}{m_i - m_{pL}} = 0.15$
- $M_p = 0$ after maneuver

· first, we can find n (mass ratio):

$$n = \frac{m_i}{m_f} = \frac{1500}{1047.3} = 1.3072 \quad (\text{sub})$$

· now, find λ (payload ratio):

$$n = \frac{1+\lambda}{\epsilon+\lambda}, \quad n(\epsilon+\lambda) = 1+\lambda$$

$$n\epsilon + n\lambda = 1+\lambda$$

$$n\epsilon = 1+\lambda - n\lambda$$

$$n\epsilon = 1+(1-n)\lambda$$

$$\frac{n\epsilon - 1}{1-n} = \lambda$$

$$\lambda = \frac{n\epsilon - 1}{1-n}$$

$$= \frac{(1.3072)(0.15)-1}{1-(1.3072)}$$

$$\boxed{\lambda = 2.1460}$$

· payload fraction (or ratio) $\boxed{\lambda = 2.1460}$

· payload mass:

$$\lambda = \frac{m_{pL}}{m_i - m_{pL}}$$

$$\lambda m_i - \lambda m_{pL} = m_{pL}$$

$$\lambda m_i = m_{pL}(1+\lambda)$$

$$\frac{\lambda m_i}{1+\lambda} = m_{pL}$$

$$m_{pL} = \frac{2.1460(1500)}{1+2.1460} = \boxed{1023.2 \text{ kg}}$$

$$m_{pl} = \frac{2.1460(1500)}{1+2.1460} = \boxed{\underline{1023.2 \text{ kg}}}$$

$$m_l = \epsilon(m_i - m_{pl}) = \underline{71.5197 \text{ kg}}$$

hmmm...
seems light!

HW 7 Astrodynamics | Ronco Pedelsten section 0101
Thursday, May 9, 2024 4:15 AM

For Q3 :

I could not get my pork chop plot to match with the provided outcome. I believe it is due to my TOF matrix, as the lambert solver is the same as given in the lecture. However, my solver does get stuck at a few inputs, so I'm not sure where the real issue lies :(

pork chop plots are so painful!

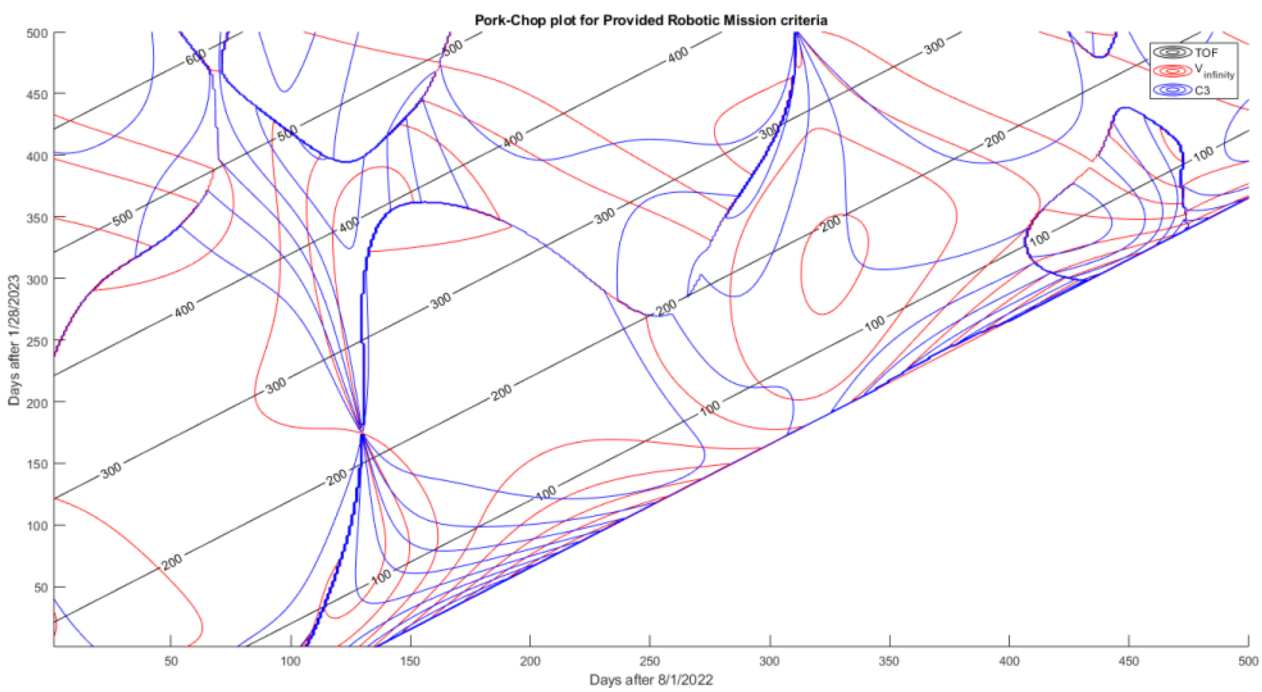


Table of Contents


```
%-------- HW 7 MATLAB code --------%
% Romeo Perlstein, section 0101    %
% Chat it's over... I'm cooked!!!  %

% Hey Eric (I think), or I mean Dr. Frizzell! Hope things have been good,
% PLEASE wish me luck in this class as I am STRUGGLING (Idk why it's been
% so hard to do school work I think I'm just cringe. Also I think Prof.
% Barbee doesn't like me heheh. OH WELL I guess the semester is over

close all
```

Q1

```
mew_earth = 0.39860*10^6; % km^3/s^2

% Equitorial, spherical orbit - velocity is constant
spacecraft_position_initial_1 = [8000;0;0]; % km
```

a

```
spacecraft_velocity_initial_1 = [0;sqrt(mew_earth/
spacecraft_position_initial_1(1));0];
accel_thrust1 = 1*10^-4; % kN/kg
```

b

see attached PDF

c

```
t01 = 0;
term1_1 = (norm(spacecraft_velocity_initial_1)/accel_thrust1);
term2_1 = (20*((accel_thrust1)^2)*(norm(spacecraft_position_initial_1)^2))/
(norm(spacecraft_velocity_initial_1)^4);
tesc1 = term1_1*( 1-term2_1^(1/8) );
```

```
fprintf("Time to escape, in seconds (analytical):\n");
tesc1
fprintf("Time to escape, in minutes (analytical):\n");
tesc1/60
```

d

make a da graph-a

```
%--- ODE func values from HW1---%
tall_er_ant = (10^-13); % Tolerance
step_size = 1; % step size
max_time = 2*tesc1; % max time (0->max_time)
t = [0:step_size:max_time]; % timestep

% ODE options
ODE_options = odeset("RelTol", tall_er_ant, "AbsTol", tall_er_ant);

initial_state = [spacecraft_position_initial_1;spacecraft_velocity_initial_1];


[T1, Y1] = ode45(@myodefun, t, initial_state, ODE_options, mew_earth,
 accel_thrust1);

hold on
plot(0,0, ".b", "MarkerSize", 50, "DisplayName","Earth")
plot(Y1(:,1), Y1(:,2), "-r","DisplayName","s/c orbit")
plot(spacecraft_position_initial_1(1),
 spacecraft_position_initial_1(2), ".k", "MarkerSize",
 10, "DisplayName", "spacecraft")
title("Graph of integrated orbit from the given initial conditions (Q1)")
xlabel("X position");
ylabel("Y position")
legend
grid on
axis equal
fprintf("The final velocity of the propegated orbit is:\n")
len1 = length(Y1(:,4:6));
Y1(len1,4:6)
```

e

find r_esc, then find time using prop'd data

```
term1_1_1 =
 norm(spacecraft_position_initial_1)*norm(spacecraft_velocity_initial_1);
term2_1_1 = 20*(accel_thrust1^2)*(norm(spacecraft_position_initial_1)^2);
resc1 = term1_1_1/(term2_1_1^(1/4));
fprintf("Radius of escape:\n")
resc1

tall_er_ant2 = 1;
for i=1:1:max_time+1
```

```
    spef_energy(i) = (norm(Y1(i,4:6))^2)/2 - mew_earth/norm(Y1(i,1:3));
    if(spef_energy(i) > 0-0.0001 && spef_energy(i) < 0+0.0001)
        time_of_energy_switch = i;
    end
    if ((norm(Y1(i,1:3)) < resc1+1) && (norm(Y1(i,1:3)) > resc1-1))
        time_at_esc_radius = i;
    end
end
figure
hold on
plot(T1, spef_energy)
yline(0, "-k")
grid on
title("Plot of specific energy vs Time")
xlabel("Time (seconds)")
ylabel("Specific Energy")


fprintf("Time to reach calculated escape radius (seconds):\n")
time_at_esc_radius
fprintf("Time of escape calculated (seconds):\n")
tesc1
fprintf("Difference in times (seconds):\n")
time_at_esc_radius - tesc1
fprintf("Time when energy flips from negative to positive (time when it is 0)
 (seconds):\n")
time_of_energy_switch


% From inspection of the integrated data, r_esc is reached at 36413
% seconds, different from our analytical answer of 34047 seconds, differing
% by roughly 2366 seconds (almost an hour!). However, inspection of the
% change in specific energy's sign value (i.e., when specific energy's
% value goes from negative to positive, or when it cross the x-axis) occurs
% at roughly 50130 seconds, which is about 20,000 seconds more than our
% calculated escape time!

% Both numerically found escape times are greater than the analytical
% answer. I think the analytical answer is conservative with it's output
% because it really doesn't take into account the force of gravity on the
% spacecraft the same way that the numerical integrated does. Since it's
% only going off of the kinematics of the problem (other than the
% acceleration due to thrust being a non-kinematic term since it is a
% force), it does not account for the extra time it might take for the
% spacecraft to break from the planets gravity well, giving us a
% conservative value.
%
% TL:DR - The exclusion of the gravity term from the t_esc equation leads
% to a faster escape time being found. Since the t_esc equation is only
% influenced by a single force (and not the accounting for the
% gravitiational force being applied to the craft), it will find a faster
% escape time.
```

Do everything like 10 times.... hurray... FOR LOOP TIME BAYBE

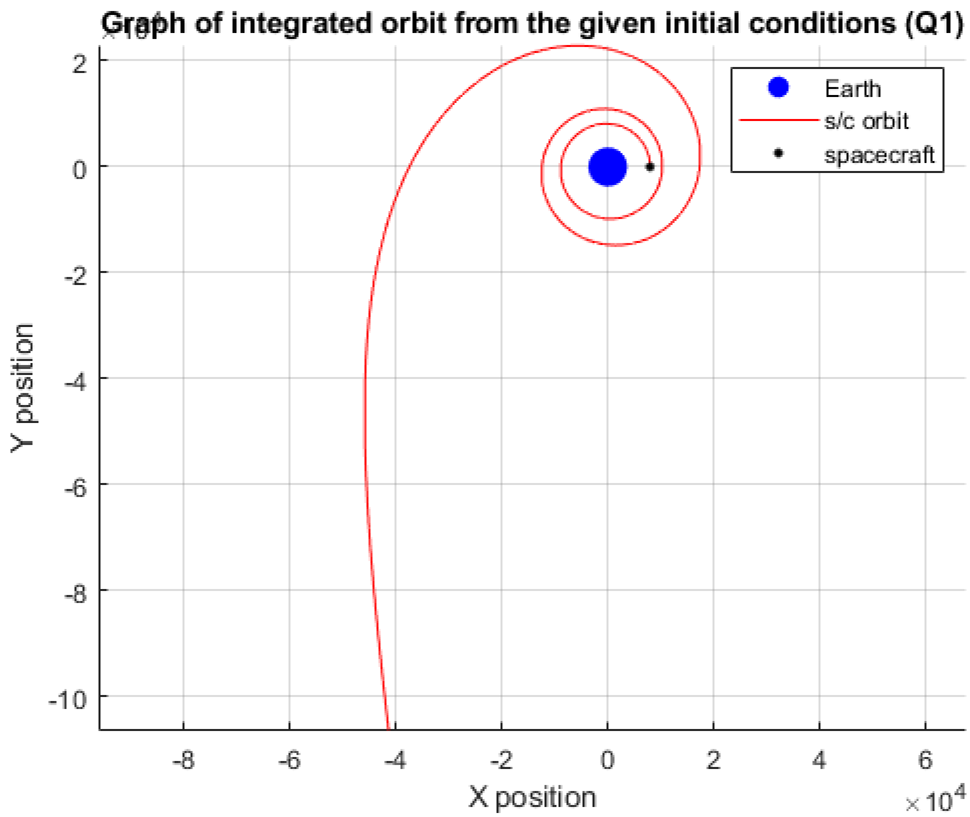
```
accel_thrusts1 =
 [0.00015;0.00025;0.00035;0.00045;0.00055;0.00065;0.00075;0.00085;0.00095;0.001];
loading_time = "Loading: [";
for i=1:1:10
    accel_thrust = accel_thrusts1(i);
    term1_1 = (norm(spacecraft_velocity_initial_1)/accel_thrust);
    term2_1 = (20*((accel_thrust)^2)*(norm(spacecraft_position_initial_1)^2))/
(norm(spacecraft_velocity_initial_1)^4);
    tesc1_multi(i) = term1_1*( 1-term2_1^(1/8) );

    max_time = 10*tesc1; % max time (0->max_time)
    t = [0:step_size:max_time]; % timestep

    [T1, Y1] = ode45(@myodefun, t, initial_state, ODE_options, mew_earth,
 accel_thrust);

    term1_1_1 =
 norm(spacecraft_position_initial_1)*norm(spacecraft_velocity_initial_1);
    term2_1_1 = 20*(accel_thrust^2)*(norm(spacecraft_position_initial_1)^2);
    resc1 = term1_1_1/(term2_1_1^(1/4));

    for ii=1:1:max_time+1
        spef_energy_multi(ii) = (norm(Y1(ii,4:6))^2)/2 - mew_earth/
norm(Y1(ii,1:3));
        if(spef_energy_multi(ii) > 0-0.001 && spef_energy_multi(ii) < 0+0.001)
            time_of_energy_switch_multi(i) = ii;
        end
        if ((norm(Y1(ii,1:3)) < resc1+1) && (norm(Y1(ii,1:3)) > resc1-1))
            time_at_esc_radius_multi(i) = ii;
        end
    end
    loading_time = loading_time + "=";
    fprintf(loading_time + "]\n")
end

if(length(time_at_esc_radius_multi) ~= 10)
    fprintf("You effed up!")
end
if(length(time_of_energy_switch_multi) ~= 10)
    fprintf("You effed up boy!")
end
figure
hold on
title("Plot of escape times using Log scale")
plot([1:1:10], tesc1_multi, "-r", DisplayName="Analytical t_e_s_c")
plot([1:1:10], time_at_esc_radius_multi, "-b", DisplayName="Numerical time
 Value for solved r_e_s_c")
plot([1:1:10], time_of_energy_switch_multi, "-m", DisplayName="Numerical Value
 for time at energy flip")
xlabel("interation")
ylabel("Time (seconds)")
grid on
set(gca,"yscale","log")
legend
```

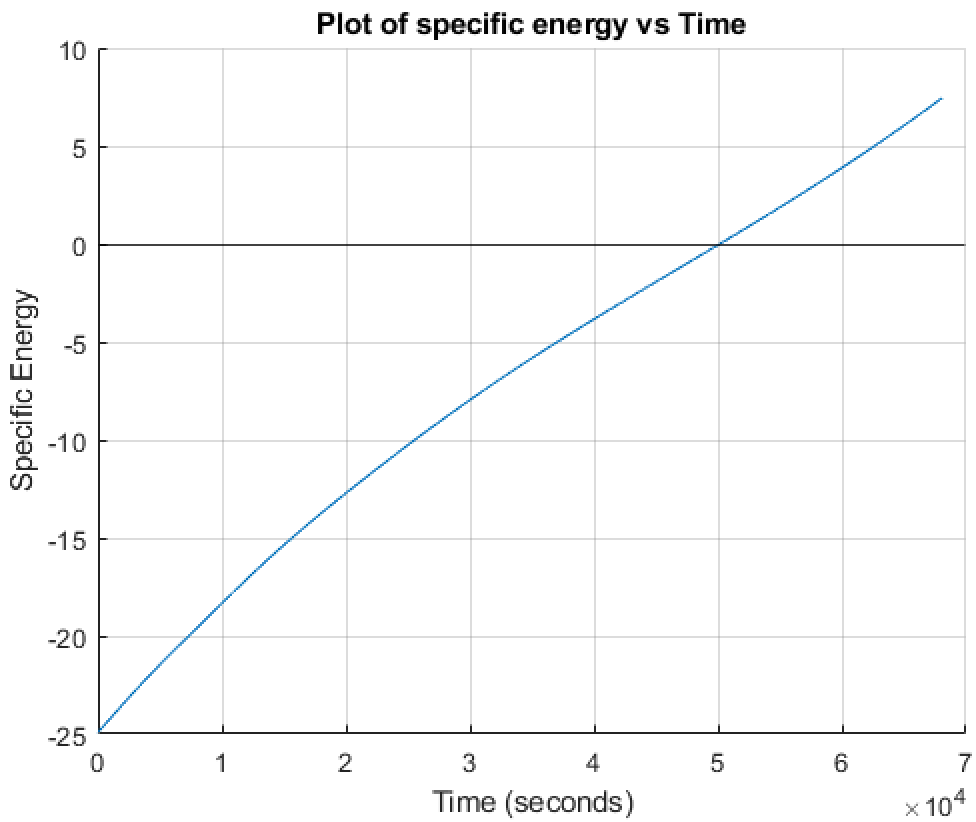
g

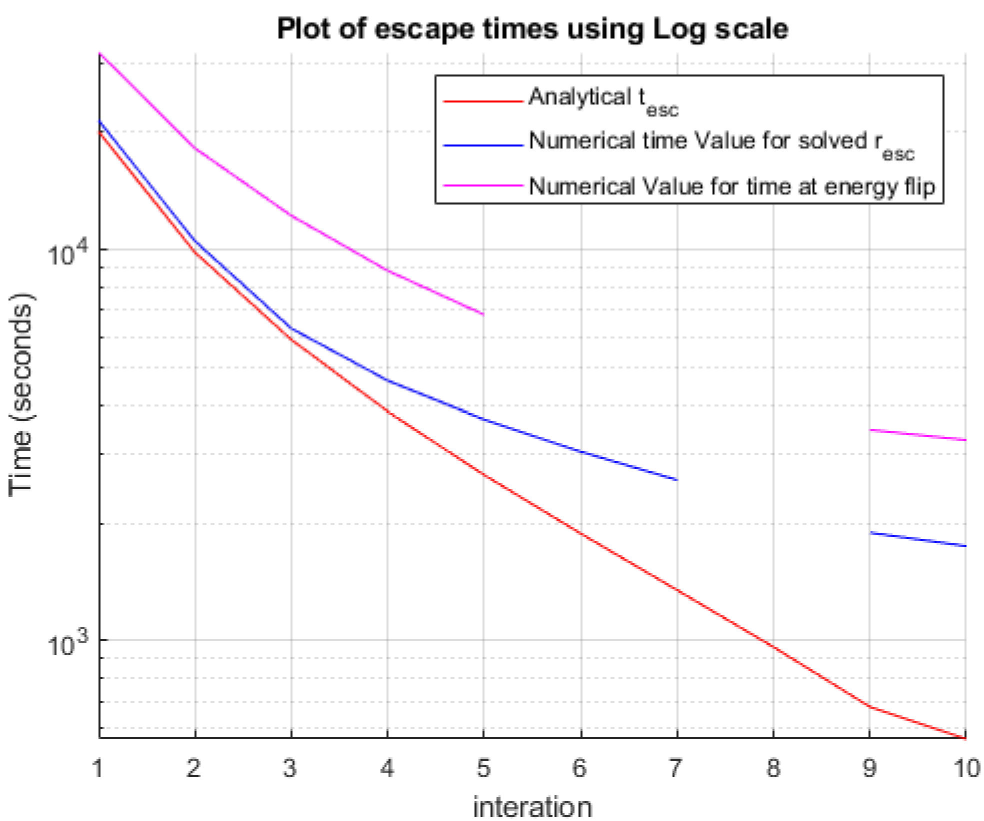
From the plot, it is clear that the analytical method is lacking in it's accuracy. As the acceleration due to thrust is increased, we see a larger separation of numerical vs analytical values of the escape time. However, there are two key things to note: the analytical method produces a smooth, continuous curve of solutions, because it gives you exactly one solution. The integration method requires that you find the value at which either the energy changes or the time at which you reach your analytically solved for r_esc. Disregarding the outliers of data in the provided plot, and assuming they are smooth curves, we can still see the growing gap between the analytical solution and the numerical integration soltuion. The main limitation, I believe of the analytical method is that it does not incorporate gravity into the equation at all, and only relies on the input thrust as it's only force. I think this limitation is what causes the analytical method to undershoot, as it requires less time to escape if there is no modeled force of gravity "hold you back" (resisting the spacecrafts power to escape orbit)

```
Time to escape, in seconds (analytical):

tesc1 =

   3.4047e+04

Time to escape, in minutes (analytical):

ans =

  567.4548

The final velocity of the propegated orbit is:

ans =

    0.5396   -4.6487         0

Radius of escape:

resc1 =

   2.9855e+04

Time to reach calculated escape radius (seconds):

time_at_esc_radius =

       36413

Time of escape calculated (seconds):

tesc1 =

   3.4047e+04

Difference in times (seconds):

ans =
```

```
   2.3657e+03

Time when energy flips from negative to positive (time when it is 0)
 (seconds):

time_of_energy_switch =

       50130

Loading: [=]
Loading: [==]
Loading: [===]
Loading: [====]
Loading: [=====]
Loading: [======]
Loading: [=======]
Loading: [========]
Loading: [=========]
Loading: [==========]
```




Plot of specific energy vs Time
Specific Energy
Time (seconds)
$\times 10^4$
10
5
0
-5
-10
-15
-20
-25
0
1
2
3
4
5
6
7


Plot of escape times using Log scale
Analytical t_{esc}
Numerical time Value for solved r_{esc}
Numerical Value for time at energy flip
Time (seconds)
10^4
10^3
interation
1
2
3
4
5
6
7
8
9
10

Q2

Find stuff with things

```
mew_mars = 0.042828*10^6;

% Given
rp = 1000;
e = 0.25;
a = rp/(1-e);
ra = a*(1+e);

% Get the periapsis speed
v_peri = sqrt((2*mew_mars)/rp - (2*mew_mars)/(rp+ra));
v_circ = sqrt(mew_mars/rp);
deltaV = v_peri-v_circ;

% now solve for propellent change
Isp = 250;
g = 9.81*(10^-3);
mi = 1500;
mi_over_mf = exp(deltaV/(Isp*g));
mf = mi/mi_over_mf;

deltaM = mi*(1-exp(-deltaV/(Isp*g)));
mf-mi;
```

part 2

given:

```
struct_ratio = 0.15;

% Find n (mass ratio)
n = mi/mf;

% now, find lambda (payload ratio)
lambda = (n*struct_ratio-1)/(1-n);

% For shits and giggles, get payload mass
mpl = (lambda*mi)/(1+lambda);
me = struct_ratio*(mi-mpl);
```

Q3

```
mew_sun = 132712*10^6; % km^3/s^2

% Create a 2D matrix containing all of our TOF values
starting_depart_date = ymdhms2jd(2022, 8, 1, 12, 0, 0);
starting_arrival_date = ymdhms2jd(2023, 1, 28, 12, 0, 0);
```

```
TOF_matrix(500,500) = 0;
TOF_matrix_check(500,500) = 0;
Julian_matrix(500,500) = 0;
for i=0:1:499
    for ii=0:1:499
        TOF_matrix(ii+1,i+1) = ((starting_arrival_date+ii) -
 (starting_depart_date+i))*24*60*60;
        TOF_matrix_check(ii+1,i+1) = (starting_arrival_date+ii) -
 (starting_depart_date+i);
        if(TOF_matrix_check(ii+1,i+1) <= 0)
            TOF_matrix(ii+1,i+1) = 0;
            TOF_matrix_check(ii+1,i+1) = 0;
        end
    end
end

% LAUNCH IS X AXIS, ARRIVAL IS Y
V_infinity_matrix(500,500) = 0;
C3_martix(500,500) = 0;
for i=0:1:499
    for ii=0:1:499
        % Find the position vectors of earth and mars
        [r1_vec, vel_earth] = findEarth(starting_depart_date+i);
        [r2_vec, vel_mars] = findMars(starting_arrival_date+ii);
        TOF = TOF_matrix(ii+1, i+1);
        % If our number is less than 45 degrees, we're close to an
        % impossible TOF (and out of bounds of the problem statement
        % anyway), so skip it and set our velocities to 0
        if(TOF_matrix_check(ii+1,i+1) <= 45)
%             fprintf("skipping\n")
            V_infinity_matrix(ii+1,i+1) = 0;
            C3_martix(ii+1,i+1) = 0;
%         elseif(TOF_matrix_check(i+1,ii+1) == 0)
%             fprintf("skipping2\n")
%             V_infinity_matrix(ii+1,i+1) = 0;
%             C3_martix(ii+1,i+1) = 0;
        else % If we're all good in the hood, actually find everything
            % Fist case, short way
            [v1_vec, v2_vec, stuck1] = romeosEpicLambartSolvor(r1_vec, r2_vec,
 TOF, "short", mew_sun);
            V_infinity_earth1 = v1_vec - vel_earth;
            V_infinity_mars1 = v2_vec - vel_mars;
            % Get the magnitude:
            val1 = norm(V_infinity_earth1) + norm(V_infinity_mars1);

            % Second case, long way
            [v1_vec, v2_vec, stuck2] = romeosEpicLambartSolvor(r1_vec, r2_vec,
 TOF, "long", mew_sun);
            V_infinity_earth2 = v1_vec - vel_earth;
            V_infinity_mars2 = v2_vec - vel_mars;
            % Get the magnitude:
            val2 = norm(V_infinity_earth2) + norm(V_infinity_mars2);

            if(stuck1 == true && stuck2 == true)
```

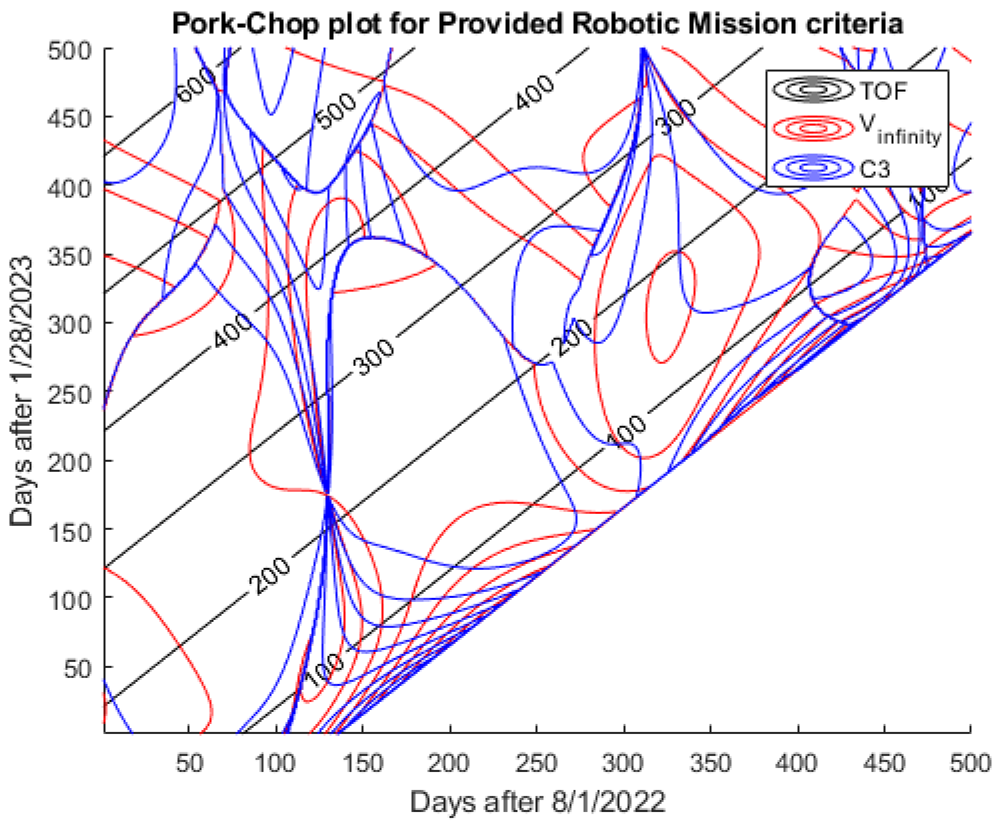
```
%                   fprintf("skipping1\n")
                V_infinity_matrix(ii+1,i+1) = 0;
                C3_martix(ii+1,i+1) = 0;
            end

            % now find which mag sum is smaller, and keep it:
            if (val1 < val2)
                C3_REAL = (norm(V_infinity_earth1))^2;
                V_infinity_REAL = norm(V_infinity_mars1);
            elseif (val2 < val1)
                C3_REAL = (norm(V_infinity_earth2))^2;
                V_infinity_REAL = norm(V_infinity_mars2);
            elseif (val1 == val2)
                C3_REAL = (norm(V_infinity_earth1))^2;
                V_infinity_REAL = norm(V_infinity_mars1);
            end
            % Save the value into our matricies for plotting purposes!
            V_infinity_matrix(ii+1,i+1) = V_infinity_REAL;
            C3_martix(ii+1,i+1) = C3_REAL;

            % If we got stuck, just skip and set the value to 0, it's
            % probably not meant to be (I have no idea why it's getting
            % stuck, end me please!
            if(stuck1 == true && stuck2 == true)
%                   fprintf("skipping1\n")
                V_infinity_matrix(ii+1,i+1) = 0;
                C3_martix(ii+1,i+1) = 0;
            end
        end
%           fprintf("Loading, please wait...\nIterations complete: " +
 int2str(i) +", " + int2str(ii) + "\n")
    end
end

figure
hold on
% contour(TOF_matrix, "-k", DisplayName="TOF");
[C, h] = contour(TOF_matrix_check, "-k", DisplayName="TOF");
clabel(C, h)

contour(V_infinity_matrix, "-r", DisplayName="V_i_n_f_i_n_i_t_y")
contour(C3_martix, "-b", DisplayName="C3")
legend
title("Pork-Chop plot for Provided Robotic Mission criteria")
xlabel("Days after 8/1/2022")
ylabel("Days after 1/28/2023")

% Contours are not like the example, but after hours of debugging and I
% can't seem to find out why. OH WELL I GUESS... I FUCKING HATE LAMBERT
% AHHHHH I HATE LAMBERT (its probably my TOF and not lambert)
```

```matlab
% 2-body prop solver, pulled from ENAE301! Ah good times... good times.
function ydot = myodefun(t, y, mew, thrust)
    r_mag = norm(y(1:3));
    ydot(1,1) = y(4);
    ydot(2,1) = y(5);
    ydot(3,1) = y(6);

    % Get the unit vector of velocity vector
    unit_vec = [y(4);y(5);y(6)]/(norm([y(4);y(5);y(6)]));

    % Get the thrust vector
    thrust_vec = unit_vec*thrust;

    % Add it to accel
    ydot(4,1) = (-mew/r_mag^3)*y(1) + thrust_vec(1);
    ydot(5,1) = (-mew/r_mag^3)*y(2) + thrust_vec(2);
    ydot(6,1) = (-mew/r_mag^3)*y(3) + thrust_vec(3);
end
```

```
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
```

```
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
```

```
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
Got stuck in a loop, returning 0's
```



Published with MATLAB® R2022b