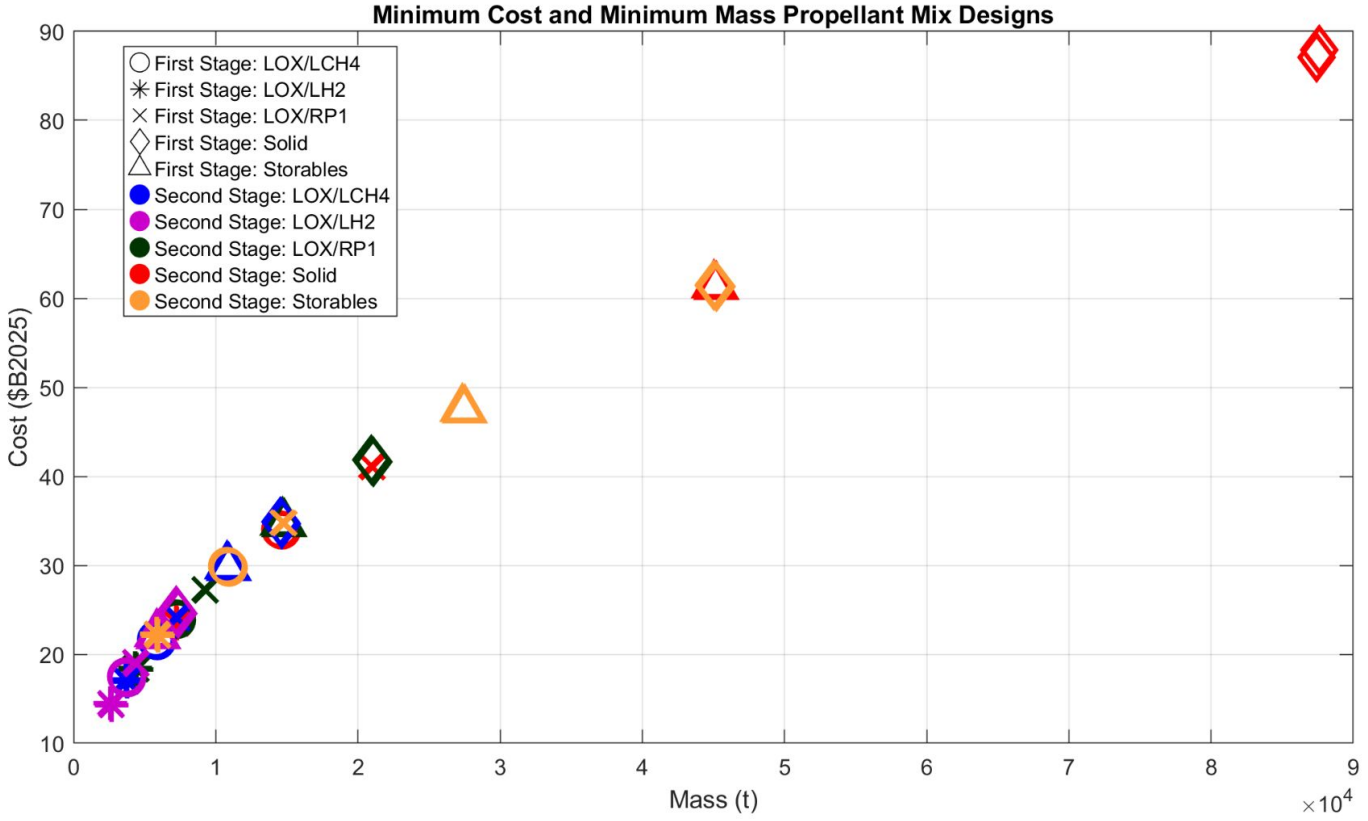
Team #15

Sara Carter · Sophie Jack · Ben Eber · Spyridon Mazis · Chris Witherspoon

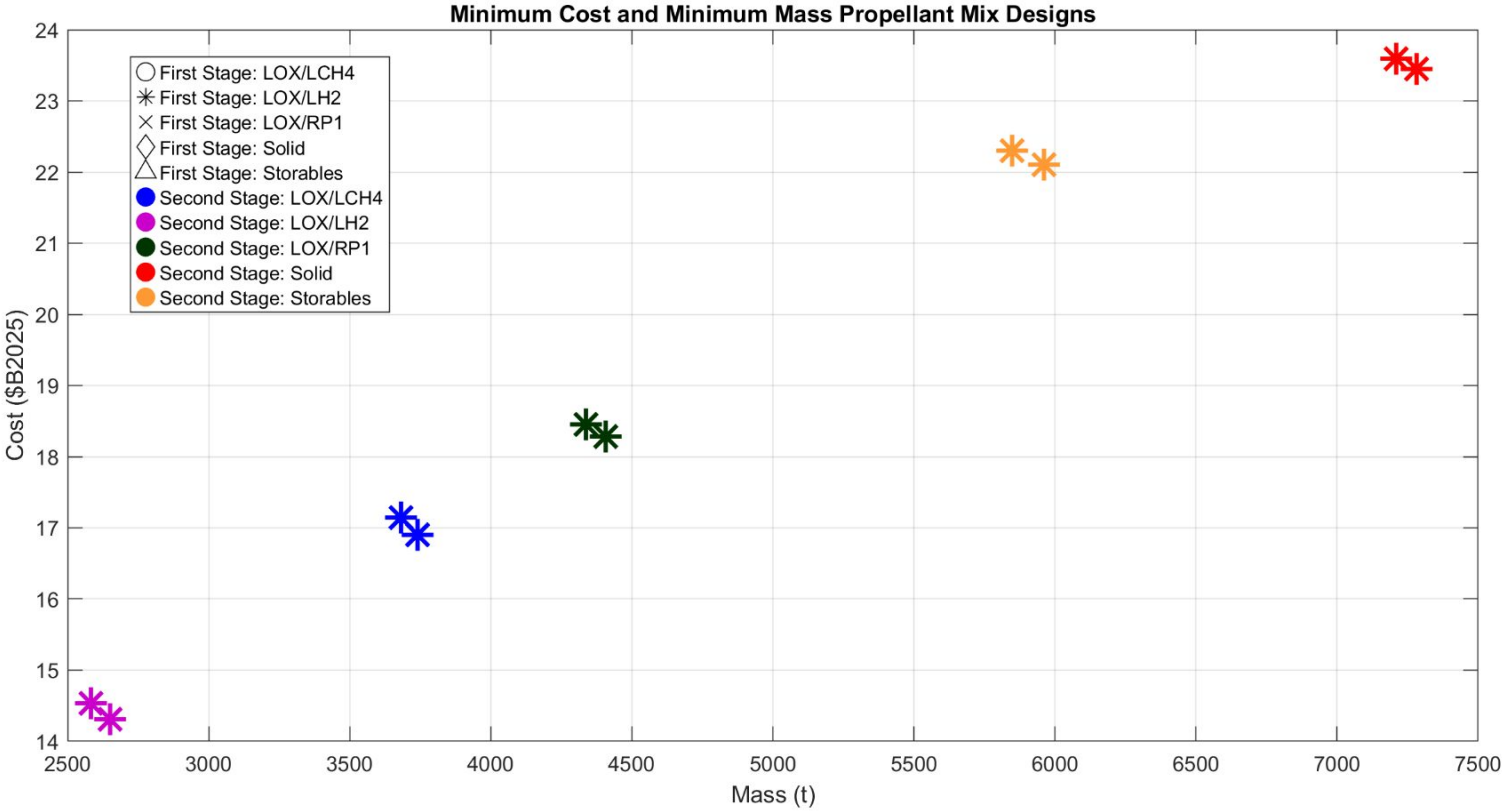
Matrix of Resposibility:

First stage prop. (columns) - Second stage prop. (rows)	LOX/LCH4	LOX/LH2	LOX/RP1	Solid	Storables
LOX/LCH4	Ben	Ben	Ben	Ben	Ben
LOX/LH2	Sara	Sara	Sara	Sara	Sara
LOX/RP1	Spyros	Spyros	Spyros	Spyros	Spyros
Solid	Sophie	Sophie	Sophie	Sophie	Sophie
Storables	Chris	Chris	Chris	Chris	Chris

Full Graph of Results



Top Mass/Cost Results



Trade Study and Conclusion

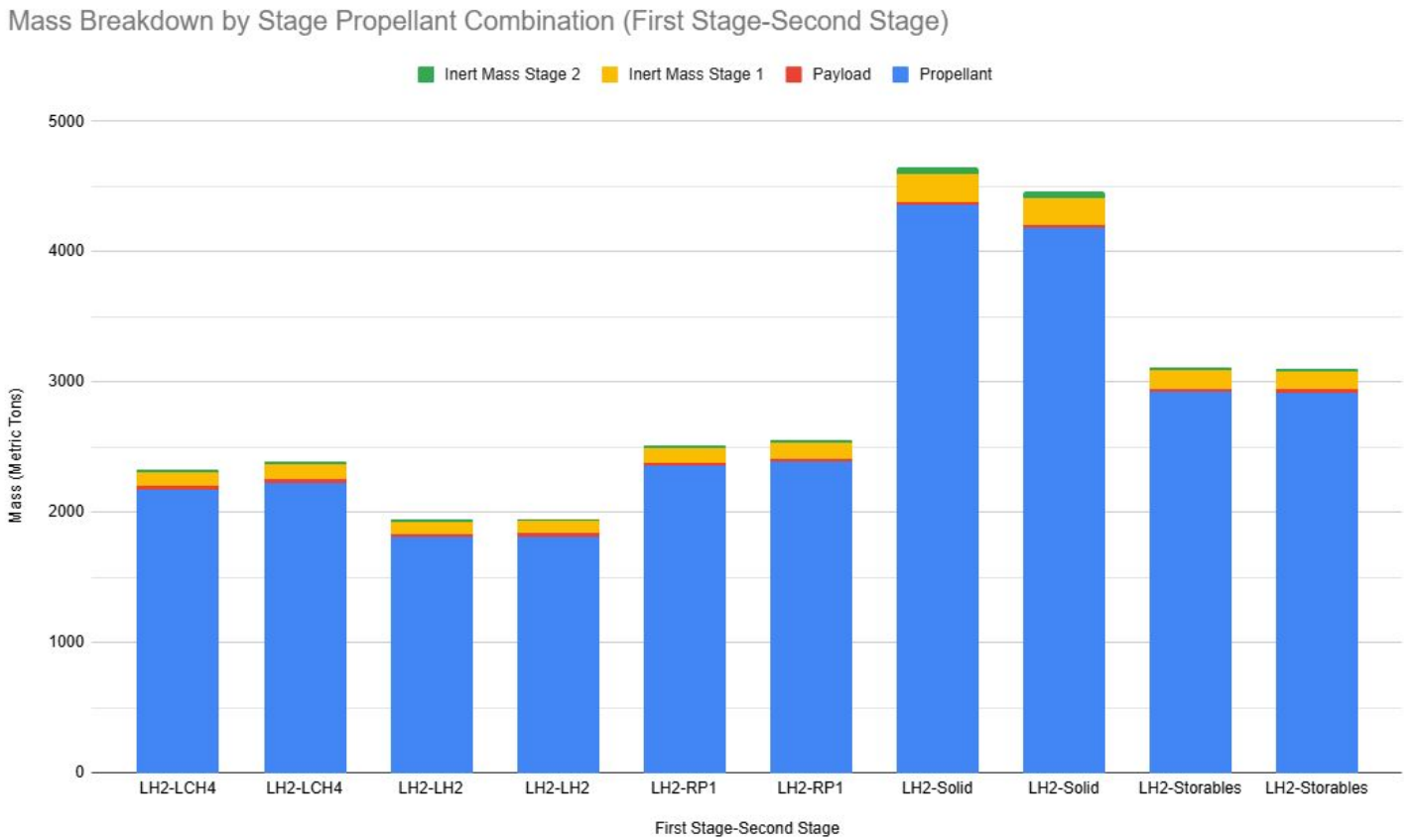
- From the data, the LOX/LH_2 + LOX/LH_2 combination was most optimized for mass and cost
  - Minimum Mass: [2582 (t), $B2025: 14.53]
  - Minimum Cost: [2650 (t), $B2025: 14.31]
- H_2 is temperamental
  - It needs cryogenic cooling, the technology for which is not available
- Higher I_{sp} in electric, but provides low thrust
  - Isp doesn't paint a full picture of how to get a launch vehicle from the ground into space
  - Future analyses will paint a fuller picture by deriving $\dot{m}$ from T2
- Our analysis is dependent on consistent $\boldsymbol{\delta}$ = 0.08, this might not be reflective of actual performance

Driving “top” results

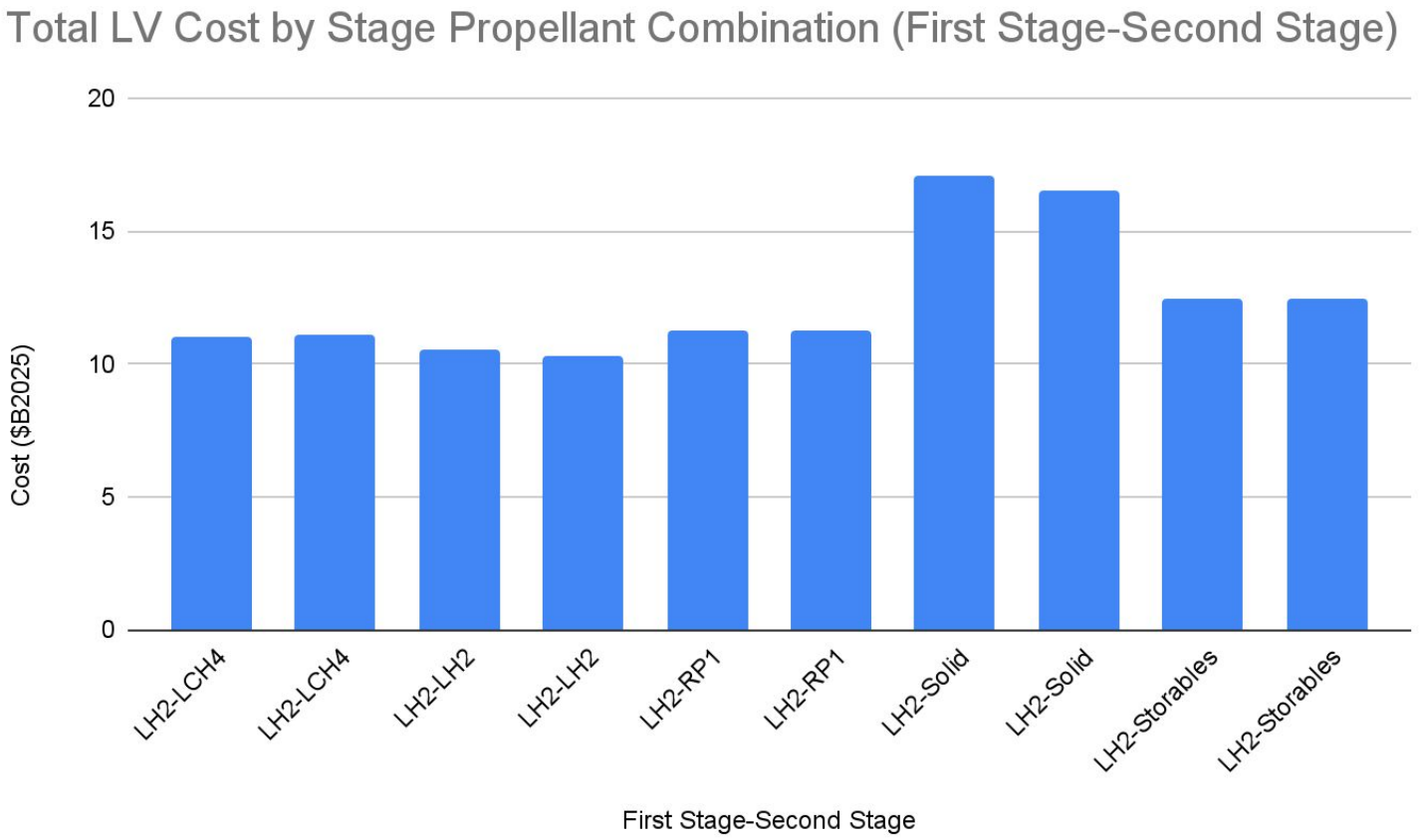
- The Isp of the first stage seems to be a strong driving force.
  - The higher the initial Isp the lower the cost and minimum mass
- In accordance with what we have learned in lecture, the higher the Isp, the higher the force per kg provided
- The Isp of the second stage also makes an impact on the minimum cost and mass as a two stage rocket uses all of its resources.
  - The second stage Isp seems to be the deciding factor if, within the subsect of the first Isp, the launch vehicle will be at the top or bottom mass/cost for the group.
- Looking at the ΔV equation, I_{sp} directly correlates to its value

$\Delta V_i = - V_{e,i} * \ln(m_{f,i}/m_{in,i})$	$I_{sp} * g_o = V_e$

2.3 System Level Results



2.3 System Level Results



2.3 Optimal Mass and Cost Results

	Propellant 1st-2nd	Mass (MT)	Cost ($B2023)	Mass Margin	Propellant Storage
Min Mass	LOX/LH2-LOX/LH2	1,980	10.5	25.33%	Cryogenic
Min Cost	LOX/LH2-LOX/LH2	2,520	10.34	27.11%	Cryogenic

LOX/LH2-LOX/LH2 is the most optimal mass and cost configurations. In comparison to the next least cost and mass estimates, LOX/LH2-LOX/LCH4, the cost of LOX/LH2-LOX/LH2 is almost 1 billion dollars less. The mass of LOX/LH2- LOX/LH2 is about half of the mass of LOX/LH2-LOX/LCH4.

The mass margins are also roughly at the 30% mark that was the requirement.

2.3 Overall Optimal Design Discussion

- The optimal design out of the two optimal mass and cost results is LOX/LH2 first stage and LOX/LH2 second stage for minimum cost (X = 0.54)

- From Submission 1, it was decided that for a LV, cost should be optimized over mass, so the minimum cost should be given design preference

- We chose to minimize cost because, as stated in early lectures, the biggest hindrance from reaching space for a single-use launch vehicle, is price, since there is an inability to reuse materials and therefore a lot of material waste produced

- Minimizing the cost would also decrease the likelihood of our mission getting cancelled due to shortcomings in the budget, which is far more likely than a cancellation due to a lack of materials

2.4 Trade Study Reflection

(1) The biggest discrepancy in the study's trends is in the cost calculations. Compared with Table 1.5 some mass combinations saved upwards of 10 ($B2025) when compared to the costs of the current systems-level analysis.

(2) Cyclical calculations allowed for the optimization of mass. When doing calculations with mass margin, then the inert mass is lessened, which in turn lessens the mass necessary for the launch vehicles. There are also other portions which influenced the mass, including the calculations for systems like avionics and gimbals, which are not calculated with great detail. In this submission, we also had the opportunity to optimize for tank size which we didn't do in submission 1, allowing us to create a more constrained design for the rocket which more accurately defines the true cost and weight.

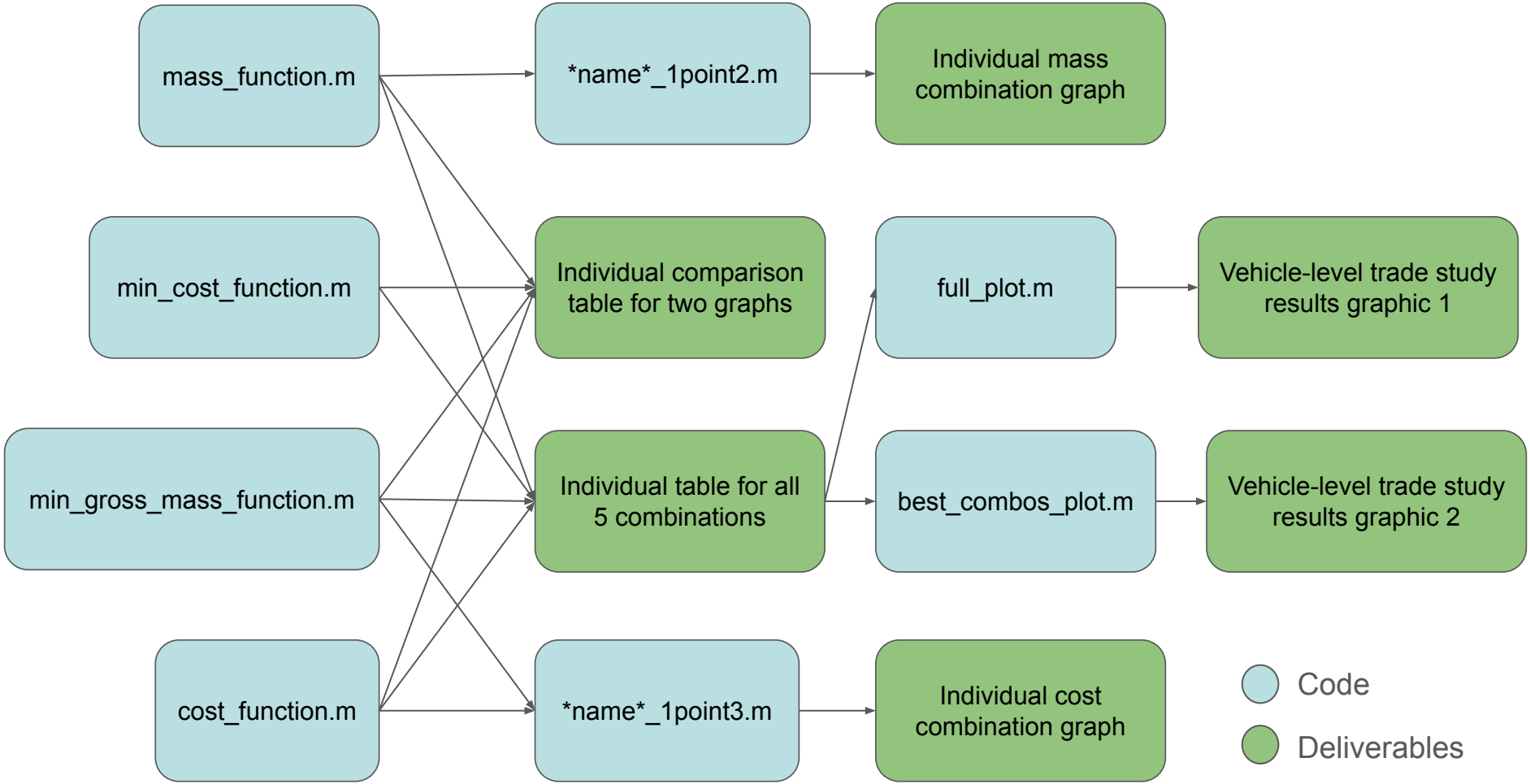
2.4 Trade Study Reflection (Continued)

(3) There are several important factors that are missing to improve this trade study.

- Engine and propellant efficiencies, as well as engine reliability. We are not taking into account real world scenarios, we assume all engines work perfectly for the whole duration of their burn time and do not leave room for failures.
- The number of engines required in the vehicle-level design might not be accounted for in the inert mass of the vehicle (number of engines physically fitting on rocket area).
- The group did not factor the possibility of engine failures due to the high range of number of engines. More engines equate to a higher possibility of failure.

All these factors could greatly hinder the performance of the lift vehicle and its DeltaV.

Code Flow Chart - Submission 1



Exported Code - Team #15 Submission 1

Note: The MATLAB functions (mass_function, cost_function, min_cost_function, and min_gross_mass_funtion) throw an error when being published to a PDF because the function is not being called within each MATLAB file. So, there is an error in each of these files that should be ignored. The error is not thrown when properly called.

Note: The CSV files being read in full_plot and best_combos_plot are each member's Table 5.

```
function [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X, delta1, delta2)
    % Returns the inert mass of stage 1, inert mass of stage 2, propellant
    % mass of stage 1, propellant mass of stage 2, and total mass given
    % specific impulse of stage 1, specific impulse of stage 2, and delV
    % fraction (delta1, delta2 are assumed to be 0.08)

    % Set constants based on mission requirements
    deltaV_tot = 12300; % m/s
    g0 = 9.81; % m/s^2
    m_pl = 26000; % kg

    % Solve for deltaV1 and deltaV2 from given delV fraction
    deltaV1 = X*deltaV_tot;
    deltaV2 = (1-X)*deltaV_tot;

    % Solve for mass ratio of stages 1 and 2 (m_i/m_f definition)
    R1 = exp(deltaV1/(g0*Isp1));
    R2 = exp(deltaV2/(g0*Isp2));

    % If (1 - R1*delta1) or (1 - R2*delta2) is less than 0, then the design
    % is structurally impossible based on the definitions
    % for m_2 and m_1 that follow, return NaN for all outputs
    if (1 - R1*delta1) <= 0 || (1 - R2*delta2) <= 0
        m_in1 = NaN; m_in2 = NaN; m_pr1 = NaN; m_pr2 = NaN; m0 = NaN;
        return
    end

    % Solve for total masses of stage 1 and stage 2
    m_2 = R2*m_pl/(1 - R2*delta2);
    m_1 = R1*m_2/(1 - R1*delta1);

    % Solve for inert masses of stage 1 and stage 2
    m_in1 = delta1*m_1;
    m_in2 = delta2*m_2;

    % Solve for propellant masses of stage 1 and stage 2
    m_pr1 = m_1 - m_in1 - m_2;
    m_pr2 = m_2 - m_pl - m_in2;

    % Assign total mass as the total mass of stage 1 (m_1 includes m_2)
    m0 = m_1;
end
```

```
Not enough input arguments.

Error in mass_function (line 13)
    deltaV1 = X*deltaV_tot;
              ^
```

Published with MATLAB® R2024b

```
function [stageCost_nr1, stageCost_nr2] = cost_function(Isp1, Isp2, X, delta1, delta2)
    % Returns the total cost given specific impulse of stage 1,
    % specific impulse of stage 2, and delV fraction (delta1, delta2 are
    % assumed to be 0.08)

    % Call mass function to get the inert masses of stage 1 and 2 for the
    % cost equation, then add the two stage costs together
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X, delta1, delta2);
    stageCost_nr1 = 13.52*m_in1^0.55;
    stageCost_nr2 = 13.52*m_in2^0.55;

end
```

```
Not enough input arguments.

Error in cost_function (line 8)
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X, delta1, delta2);
                                                     ^^^^
```

Published with MATLAB® R2024b

```
function [min_X, min_cost] = min_cost_function(Isp1, Isp2, delta1, delta2)
    % Returns the minimum cost given specific impulse of stage 1,
    % specific impulse of stage 2, and delV fraction (delta1, delta2 are
    % assumed to be 0.08)

    % Find minimum cost by iteratively calling cost_function and comparing
    % total cost for each X
    min_cost = realmax;
    for X = 0:0.001:1
        [stageCost_nr1, stageCost_nr2] = cost_function(Isp1, Isp2, X, delta1, delta2);
        total_cost = stageCost_nr1 + stageCost_nr2;
        if total_cost < min_cost
            min_cost = total_cost;
            min_X = X;
        end
    end
end
```

```
Not enough input arguments.

Error in min_cost_function (line 10)
        [stageCost_nr1, stageCost_nr2] = cost_function(Isp1, Isp2, X, delta1, delta2);
                                                       ^^^^
```

Published with MATLAB® R2024b

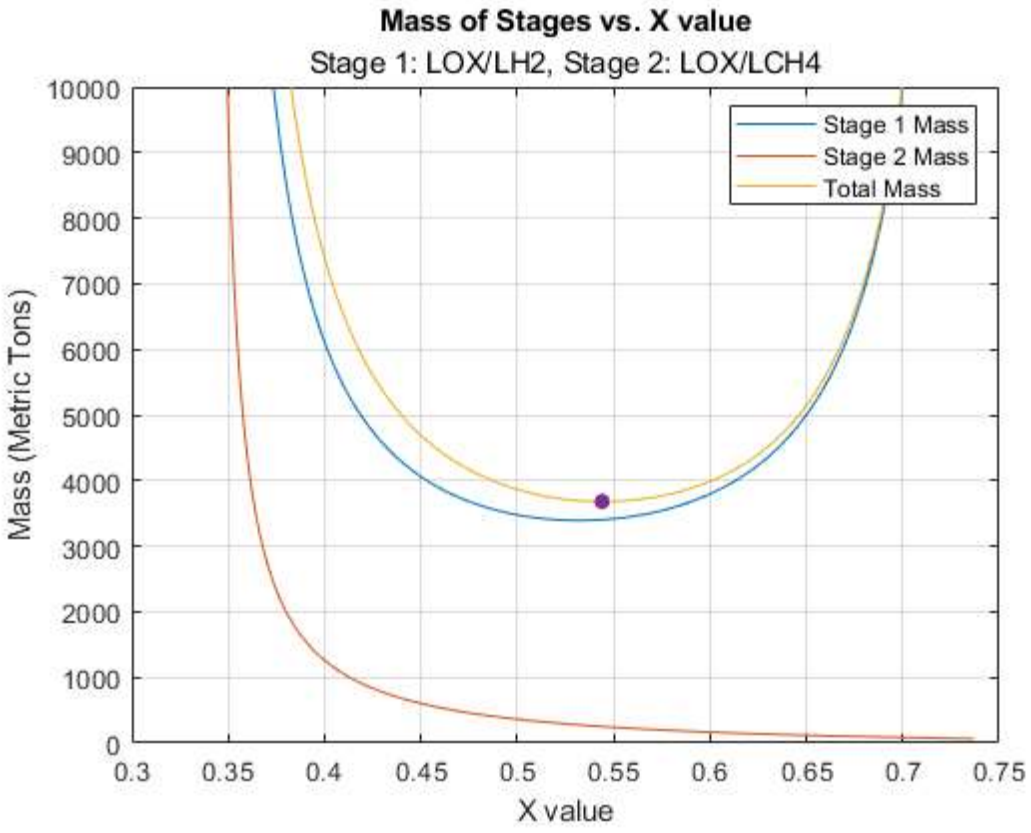
```
function [min_X, min_gross_mass] = min_gross_mass_function(Isp1, Isp2, delta1, delta2)
    % Returns the minimum gross mass given specific impulse of stage 1,
    % and specific impulse of stage 2 (delta1, delta2 are
    % assumed to be 0.08)

    % Initialize minimum gross mass
    min_gross_mass = realmax;

    % Find minimum gross mass by iteratively calling mass_function and comparing
    % gross mass for each X
    for X = 0:0.001:1
        [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X, delta1, delta2);
        if m0 < min_gross_mass
            min_gross_mass = m0;
            min_X = X;
        end
    end

end
```

```
Not enough input arguments.

Error in min_gross_mass_function (line 12)
        [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X, delta1, delta2);
                                                         ^^^^
```

Published with MATLAB® R2024b

```
% I have the LOX/LCH4 second stage row
% the first stage I am selecting is LOX/LH2 and the
% second stage is LOX/LCH4
addpath('..')
% given Constants
Delta_1 = 0.08;
Delta_2 = 0.08;
Isp_1 = 366;
Isp_2 = 327;
m_PL = 26000;

% the range of x from 0 to 1 at 0.01 intervals and the
% number of x values we will have
X = 0:0.001:1;
X_length = 1001;

% the arrays
X_array = zeros(1,X_length);
M_stage1 = zeros(1,X_length);
M_stage2 = zeros(1,X_length);
M_total = zeros(1,X_length);

% filling the arrays
S = 1;
while S <= X_length
    X_Position = X(S);
    % call mass function from 1.1
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp_1, Isp_2, X_Position, Delta_1, Delta_2);
    % defind and populate the arrays from these values
    M_stage1(S) = m_in1 + m_pr1;
    M_stage2(S) = m_in2 + m_pr2;
    M_total(S) = m0;
    S = S + 1;
end

% finding the minimum gross mass
Minimum_M0 = inf;
Minimum_M0X = NaN;

S = 2;
while S <= X_length
    if (M_stage1(S) > 0) && (M_stage2(S) > 0) && (M_total(S) < Minimum_M0)
        Minimum_M0 = M_total(S);
        Minimum_M0X = X(S);
    end
    S = S + 1;
end

% Converting to metric tons

M_stage1 = M_stage1 / 1000;
M_stage2 = M_stage2 / 1000;
M_total = M_total / 1000;
Minimum_M0 = Minimum_M0 / 1000;

% making the plots
```

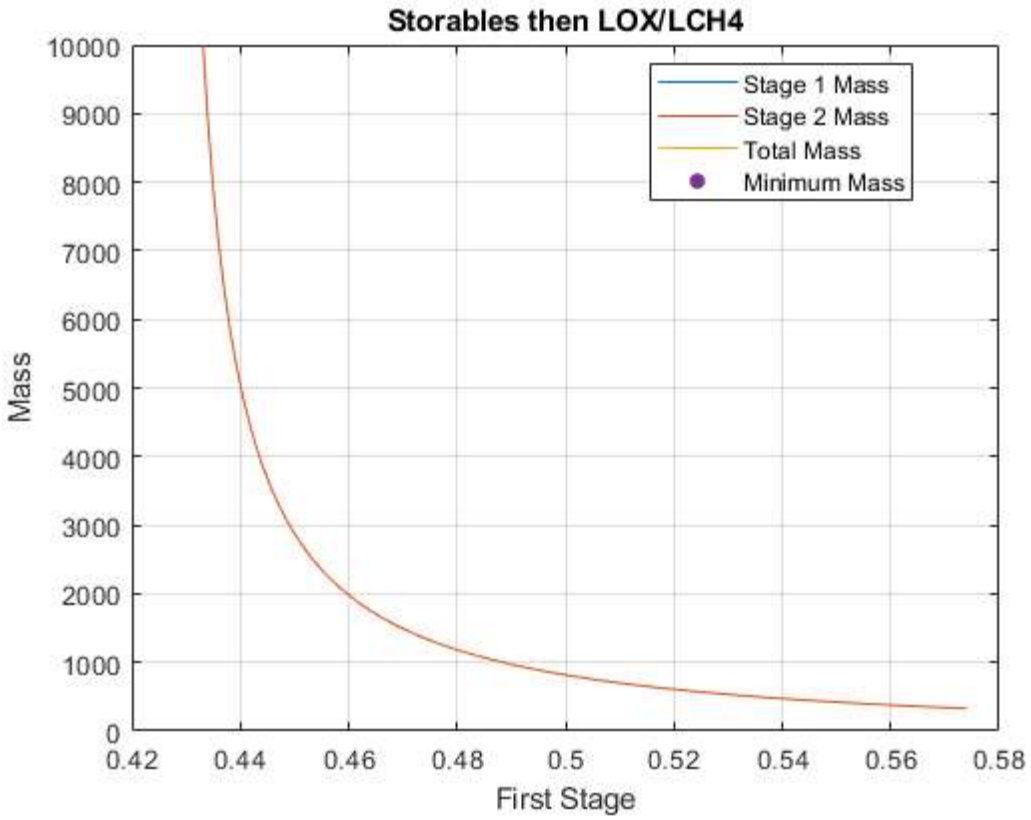
```
plot(X, M_stage1);
hold on;
plot(X, M_stage2);
plot(X, M_total);
plot(Minimum_M0X, Minimum_M0, '.', 'MarkerSize',20)
ylim([0,1e4]);
xlabel('X value');
ylabel('Mass (Metric Tons)');
title('Mass of Stages vs. X value');
subtitle('Stage 1: LOX/LH2, Stage 2: LOX/LCH4')
legend('Stage 1 Mass', 'Stage 2 Mass', 'Total Mass');
grid on;
```



Published with MATLAB® R2024b

```
% TP 1
% Chris Witherspoon
% 1.2

addpath('..')

% Delta constants
d1 = 0.08 ;
d2 = 0.08 ;

% Specific impulse (storables)
isp = 285 ;
isp_comp = 285 ; % Second

% Payload mass
mpl = 26000 ;

% X variable array
x = 0:0.001:1 ;

% Stage arrays initializations
first_stage = [] ;
second_stage = [] ;
tot_mass = [] ;

% Array population
i = 1 ; % initialize the variable to loop
while i <= length(x)
    [min1, min2, mpr1, mpr2, m0] = mass_function(isp, isp_comp, x(i), d1, d2);
    first_stage(end+1) = min1 + mpr1 ;
    second_stage(end+1) = min2 + mpr2 ;
    tot_mass(end+1) = m0 ;
    i = i + 1 ;
end

% Value placeholder for the mass
min_m0 = realmax ;
min_mox = NaN ;
% Check the min value
i = 1 ; %
while  i <= length(x)
    if first_stage(i) > 0 && second_stage(i) > 0 && tot_mass(i) < min_m0
        min_m0 = tot_mass(i) ;
        min_mox = x(i) ;
    end
    i = i + 1 ;
end

% Plotting the graph
plot(x,first_stage/1000) ;
hold on ;
grid on ;
title("Storables then LOX/LCH4")
plot(x, second_stage/1000) ;
plot(x, tot_mass/1000) ;
plot(min_mox, min_m0/1000, '.', MarkerSize = 20) ;
```

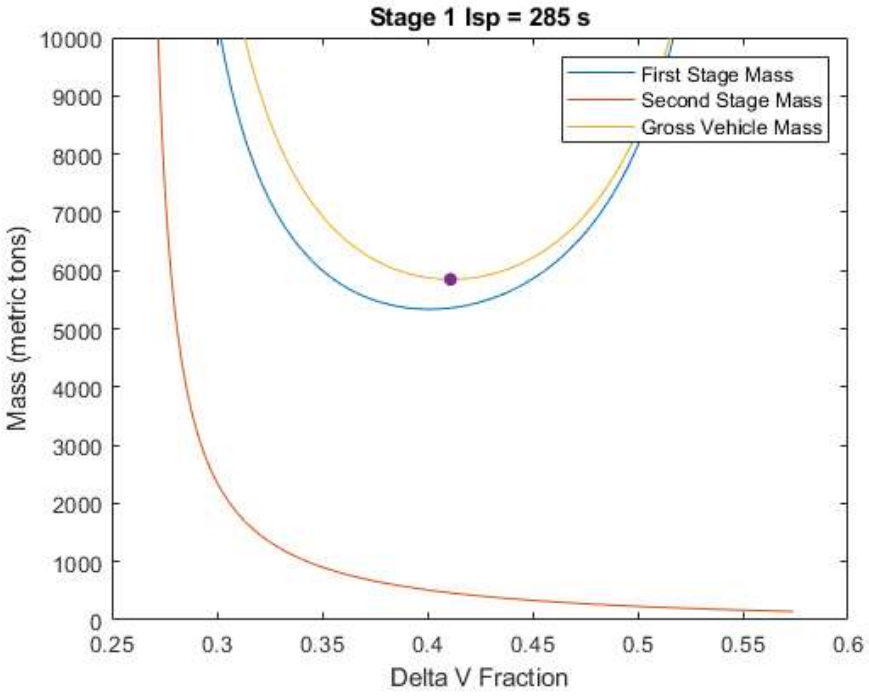
```
legend('Stage 1 Mass','Stage 2 Mass','Total Mass','Minimum Mass','Location','best');
ylim([0, 1e4]) ;
xlabel("First Stage") ;
ylabel("Mass") ;
```



Published with MATLAB® R2024b

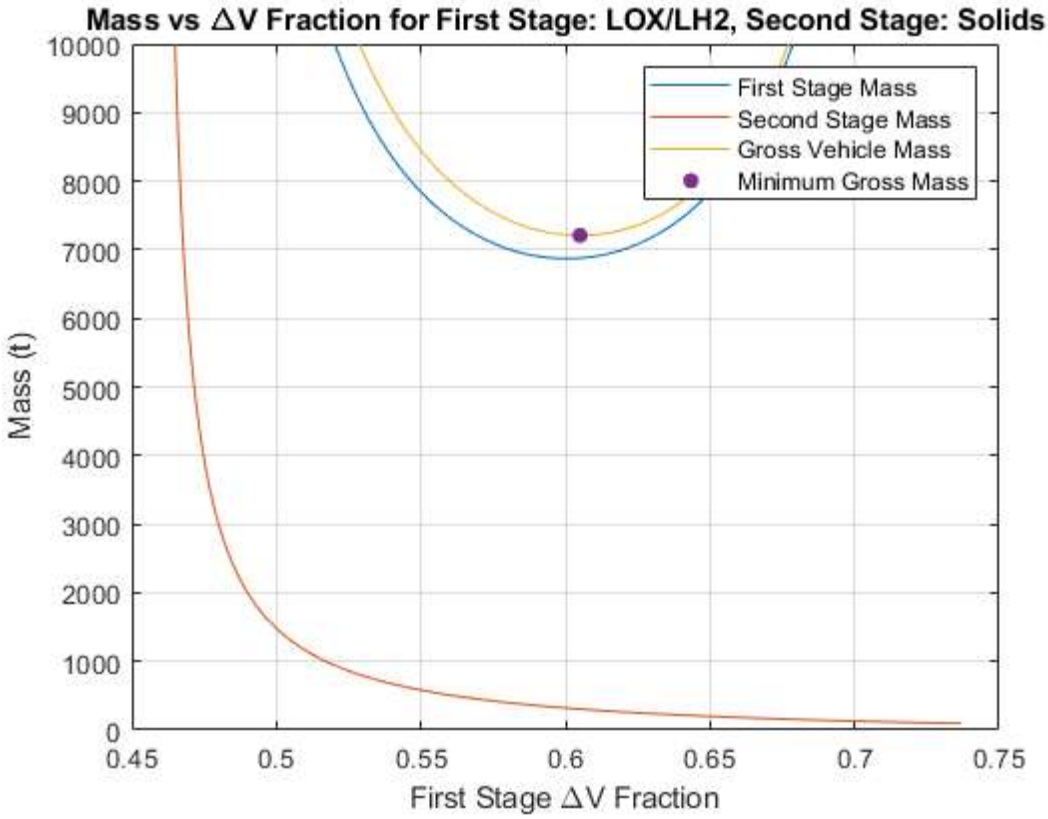
```
%Sara C 1.2a Code

%Looking at mixture of:
%Stage 2: LOX/LH2
%Varied Stage 1: LOX/LCH4, LOX/LH2, LOX/RP1, Solid, Storables

%Variables & Given Parameters
delta1 = 0.08;
delta2 = 0.08;
payload = 26000; %kg
%Specific Impulse
Isp_s2 = 366; %stage 2
%LCH4, LH2, RP1, solids, storables
Isp = [327, 366, 311, 269, 285]; %stage 1 varies


%initialize X (this is delta_V ratio/fraction)
X=0:0.001:1;

for isp = 1:length(Isp)
    %mass arrays
    m_s1 = [];
    m_s2 = [];
    m_0 = [];
    min_gross_mass = realmax;

    for i = 1:length(X)
        %calling mass function to populate mass arrays
        [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp(isp), Isp_s2, X(i), delta1, delta2);
        m_s1 = [m_s1 (m_pr1 + m_in1)];
        m_s2 = [m_s2 (m_pr2 + m_in2)];
        m_0 = [m_0 m0];

        %min gross mass function - returns the x,y minimum of the m_0 array
        if m_s1(i) > 0 && m_s2(i) > 0 && m_0(i) < min_gross_mass
            min_gross_mass = m_0(i); %convert to metric tons
            x_min = X(i);

        end
    end

    %kg to metric tons
    m_s1 = m_s1/1000;
    m_s2 = m_s2/1000;
    m_0 = m_0/1000;
    min_gross_mass = min_gross_mass/1000;

    %output - min mass and corresponding deltaV
    fprintf('Stage 1 Isp = %d s --> Optimum at X = %.3f, Min Gross Mass = %.2f metric tons\n', Isp(isp), x_min, min_gross_mass);

    %plotting
    plot(X, m_s1);
    hold on;
    plot(X, m_s2);
    plot(X, m_0);
    plot(x_min, min_gross_mass, '.', MarkerSize=20);
    ylim([0, 1e4]);
    title(sprintf('Stage 1 Isp = %d s', Isp(isp)))
    legend("First Stage Mass", "Second Stage Mass", "Gross Vehicle Mass")
    xlabel("Delta V Fraction");
    ylabel("Mass (metric tons)")
    hold off;

end
```

```
Stage 1 Isp = 327 s --> Optimum at X = 0.456, Min Gross Mass = 3680.72 metric tons
Stage 1 Isp = 366 s --> Optimum at X = 0.500, Min Gross Mass = 2582.05 metric tons
Stage 1 Isp = 311 s --> Optimum at X = 0.438, Min Gross Mass = 4337.65 metric tons
Stage 1 Isp = 269 s --> Optimum at X = 0.395, Min Gross Mass = 7209.12 metric tons
Stage 1 Isp = 285 s --> Optimum at X = 0.411, Min Gross Mass = 5848.23 metric tons
```



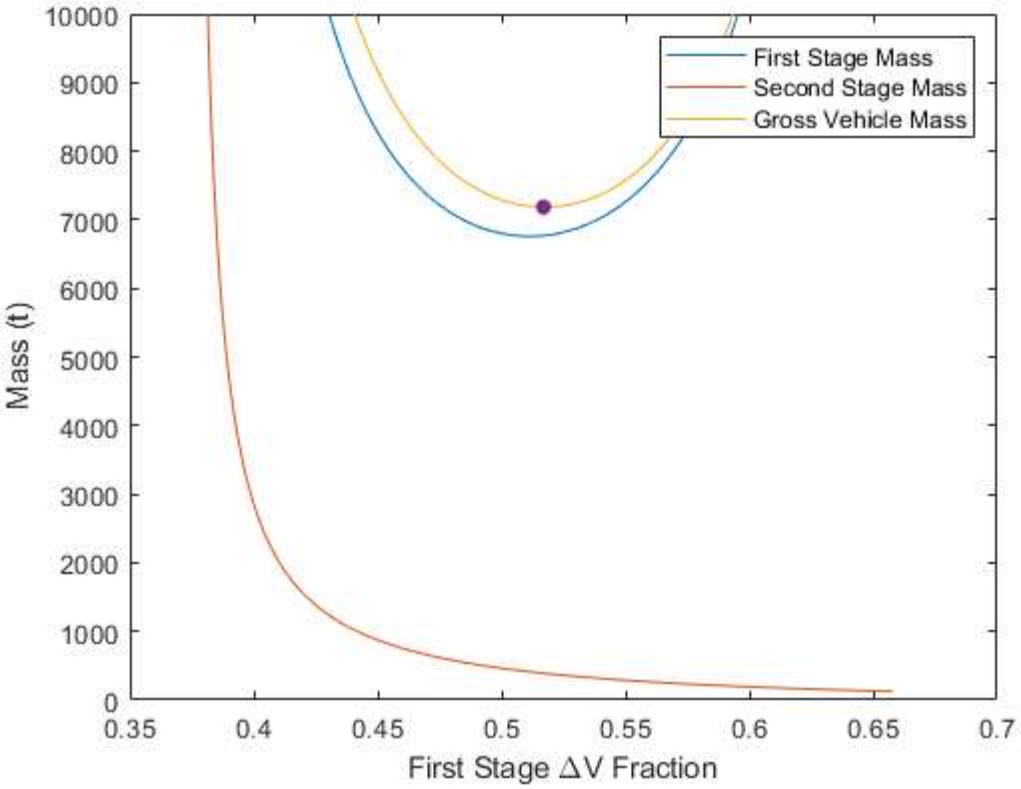
Published with MATLAB® R2024b

```
% solids row
% combo: first prop: LOX/LH2, second prop: solid
addpath('..')

% Set constants based on requirements and propellant combination
delta1 = 0.08;
delta2 = 0.08;
Isp1 = 366;
Isp2 = 269;
m_pl = 26000;

% Initialize/populate arrays
X = 0:0.001:1;
m_stage1_arr = [];
m_stage2_arr = [];
m0_arr = [];

% For each X value, call mass_function and populate the arrays for mass of
% stage 1, mass of stage 2, and total mass
for i = 1:length(X)
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X(i), delta1, delta2);
    m_stage1_arr(end+1) = m_in1 + m_pr1;
    m_stage2_arr(end+1) = m_in2 + m_pr2;
    m0_arr(end+1) = m0;
end

% Find the minimum gross mass, similar to min_gross_mass_function, but save
% the associated X for the minimum gross mass
min_m0 = realmax;
for i = 1:length(m0_arr)
    if m_stage1_arr(i) > 0 && m_stage2_arr(i) > 0 && m0_arr(i) < min_m0
        min_m0 = m0_arr(i);
        min_m0_X = X(i);
    end
end

% Plot mass of stage 1, mass of stage 2, and total mass in metric tons
% as a function of X. Also plot as a point the minimum gross mass
plot(X, m_stage1_arr/1000)
hold on
plot(X, m_stage2_arr/1000)
plot(X, m0_arr/1000)
plot(min_m0_X, min_m0/1000, '.', MarkerSize=20)
ylim([0, 1e4])
grid on
legend("First Stage Mass", "Second Stage Mass", "Gross Vehicle Mass", "Minimum Gross Mass")
xlabel("First Stage \DeltaV Fraction")
ylabel("Mass (t)")
title("Mass vs \DeltaV Fraction for First Stage: LOX/LH2, Second Stage: Solids")
```



Published with MATLAB® R2024b

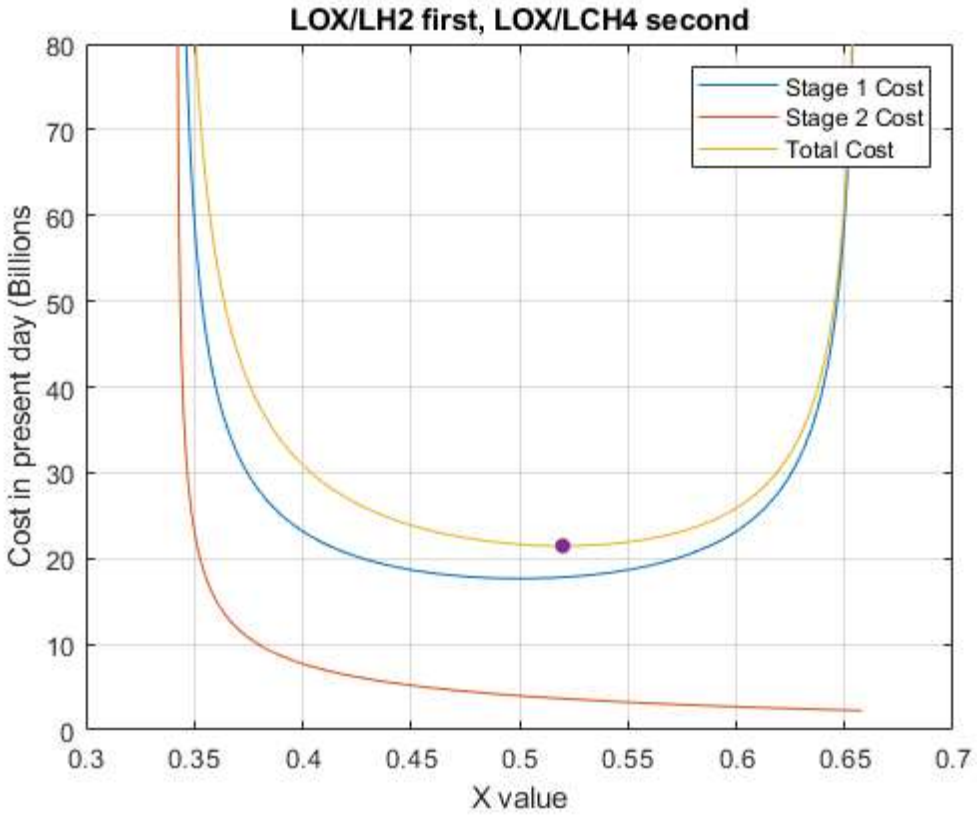
```
% LOX/RP1 row
% combo: first prop: LOX/LCH4, second prop: LOX/RPI
addpath('..')

% Set constants based on requirements and propellant combination
delta1 = 0.08;
delta2 = 0.08;
Isp1 = 327;
Isp2 = 311;
m_pl = 26000;

% Initialize/populate arrays
X = 0:0.001:1;
m_stage1_arr = [];
m_stage2_arr = [];
m0_arr = [];

% For each X value, call mass_function and populate the arrays for mass of
% stage 1, mass of stage 2, and total mass
for i = 1:length(X)
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(Isp1, Isp2, X(i), delta1, delta2);
    m_stage1_arr(end+1) = m_in1 + m_pr1;
    m_stage2_arr(end+1) = m_in2 + m_pr2;
    m0_arr(end+1) = m0;
end

% Find the minimum gross mass, similar to min_gross_mass_function, but save
% the associated X for the minimum gross mass
min_m0 = realmax;
for i = 1:length(m0_arr)
    if m_stage1_arr(i) > 0 && m_stage2_arr(i) > 0 && m0_arr(i) < min_m0
        min_m0 = m0_arr(i);
        min_m0_X = X(i);
    end
end

% Plot mass of stage 1, mass of stage 2, and total mass in metric tons
% as a function of X. Also plot as a point the minimum gross mass
plot(X, m_stage1_arr/1000)
hold on
plot(X, m_stage2_arr/1000)
plot(X, m0_arr/1000)
plot(min_m0_X, min_m0/1000, '.', MarkerSize=20)
ylim([0, 1e4])
legend("First Stage Mass", "Second Stage Mass", "Gross Vehicle Mass")
xlabel("First Stage \DeltaV Fraction")
ylabel("Mass (t)")
```



Published with MATLAB® R2024b

```
% I have the LOX/LCH4 second stage row
% the first stage I am selecting is LOX/LH2 and the
% second stage is LOX/LCH4
addpath('..')
% given Constants
Delta_1 = 0.08;
Delta_2 = 0.08;
Isp_1 = 327;
Isp_2 = 327;
m_PL = 26000;

% the range of x from 0 to 1 at 0.01 intervals and the
% number of x values we will have
X = 0:0.001:1;
X_length = 1001;

% the arrays
Stage1_cost = zeros(1,X_length);
Stage2_cost = zeros(1,X_length);

% filling the arrays
S = 1;
while S <= X_length
    X_Position = X(S);
    % call cost function
    [stageCost_nr1, stageCost_nr2] = cost_function(Isp_1, Isp_2, X_Position, Delta_1, Delta_2);
    % defind and populate the arrays from these values
    Stage1_cost(S) = stageCost_nr1;
    Stage2_cost(S) = stageCost_nr2;

    S = S + 1;
end
%total it all together
Total_cost = Stage1_cost + Stage2_cost;

% finding the minimum cost
Minimum_Cost = inf;
Minimum_X = NaN;

S = 2;
while S <= X_length
    if (Stage1_cost(S) > 0) && (Stage2_cost(S) > 0) && (Total_cost(S) < Minimum_Cost)
        Minimum_Cost = Total_cost(S);
        Minimum_X = X(S);
    end
    S = S + 1;
end

% Converting to billion dollars

Stage1_cost = Stage1_cost / 1000;
Stage2_cost = Stage2_cost / 1000;
Total_cost = Total_cost / 1000;
Minimum_Cost = Minimum_Cost / 1000;

% making the plots
```

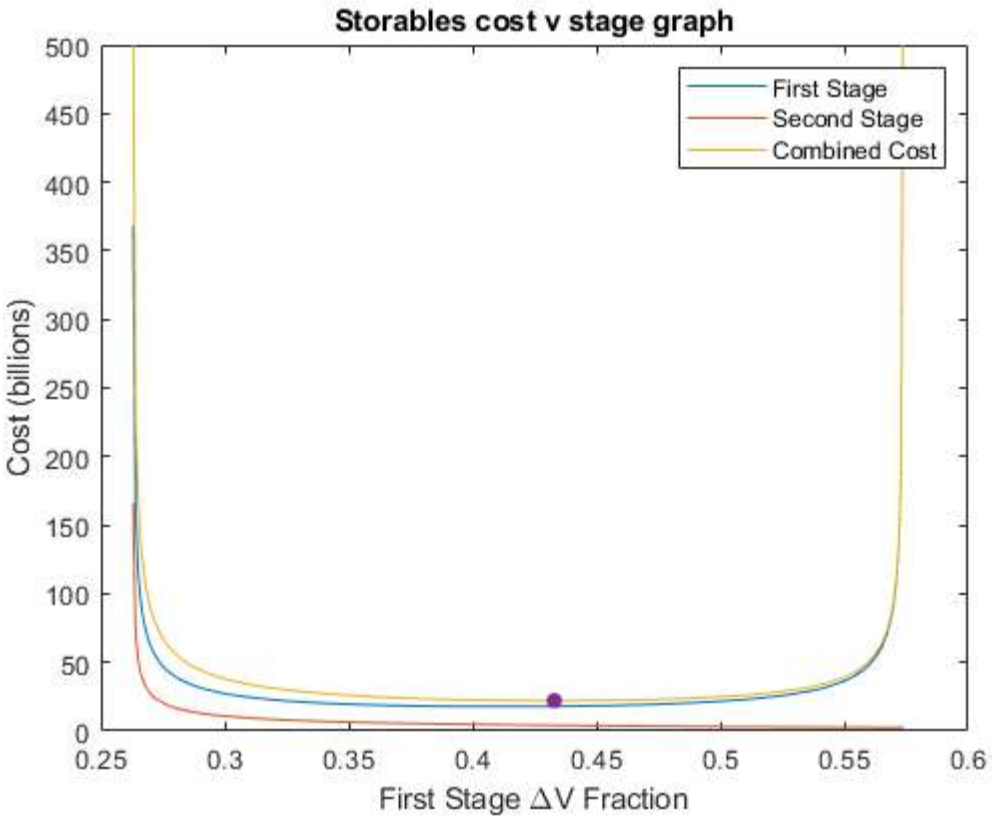
```
plot(X, Stage1_cost);
hold on;
plot(X, Stage2_cost);
plot(X, Total_cost);
plot(Minimum_X, Minimum_Cost, '.', 'MarkerSize',20)
ylim([0,80]);
xlabel('X value');
ylabel('Cost in present day (Billions');
title('LOX/LH2 first, LOX/LCH4 second');
legend('Stage 1 Cost', 'Stage 2 Cost', 'Total Cost');
grid on;
```



Published with MATLAB® R2024b

```
% TP 1
% Chris Witherspoon
% 1.3

addpath('..')

% Delta constants
d1 = 0.08 ;
d2 = 0.08 ;

% Specific impulse (storables)
isp = 285 ;
isp_comp = 366 ; % 2nd liquid

% Payload mass
mpl = 26000 ;

% Initialize the x array
x = 0:0.001:1 ;

% Create an empty array for later use
first_stage = [];
second_stage = [];

% Cost function loop
i = 1 ;
while i <= length(x)
    [first_stage_cost, second_stage_cost] = cost_function(isp, isp_comp, x(i), d1, d2);
    first_stage(end+1) = first_stage_cost;
    second_stage(end+1) = second_stage_cost;
    i = i + 1 ;
end

% Create an array for the total cost
total_cost = first_stage + second_stage ;

% Initialize the value to change for later
min = realmax;

% Find the value of the minimum cost
i = 1 ;
while i<= length(total_cost)
    if ~isnan(total_cost(i)) && total_cost(i) < min
        min = total_cost(i);
        min_cost_val = x(i);
    end
    i = i + 1 ;
end


% Plot the graph
plot(x, first_stage/1000)
hold on
plot(x, second_stage/1000)
plot(x, total_cost/1000)
plot(min_cost_val, min/1000, '.', MarkerSize=20)
```

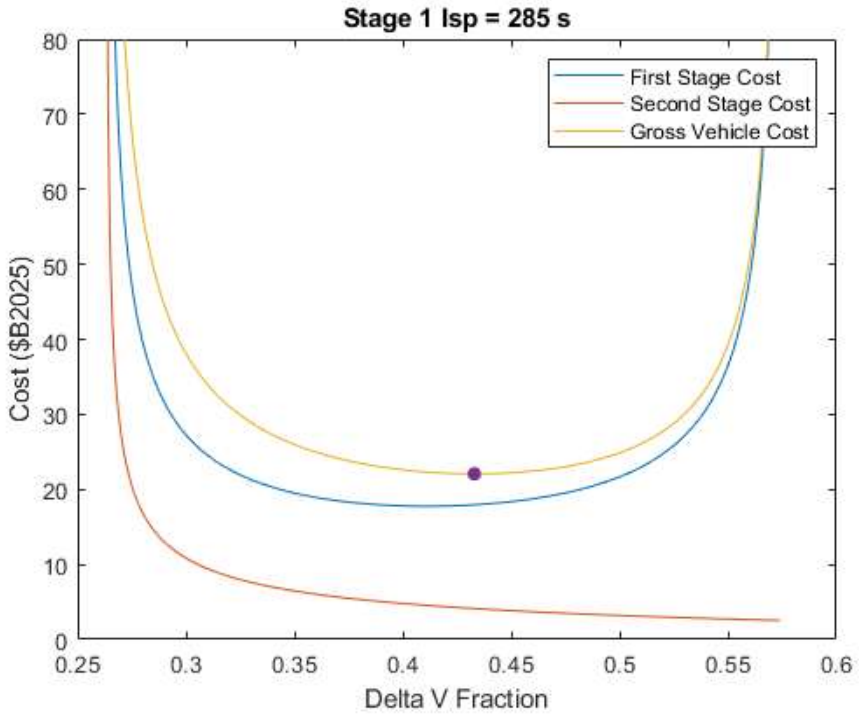
```
ylim([0, 500])
legend("First Stage", "Second Stage", "Combined Cost")
xlabel("First Stage \DeltaV Fraction")
ylabel("Cost (billions)")
title("Storables cost v stage graph") % Storables is the consistent
```



Published with MATLAB® R2024b

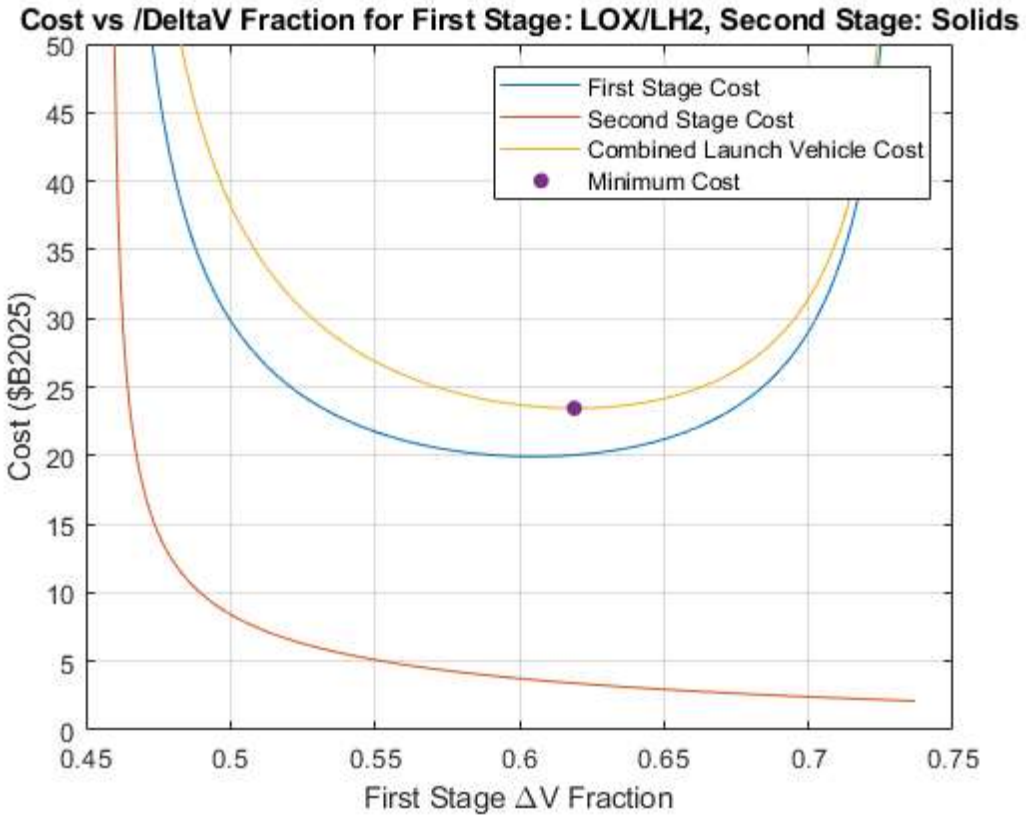
```
%Sara C 1.3a Code
%cost analysis

%Looking at mixture of:
%Stage 2: LOX/LH2
%Varied Stage 1: LOX/LCH4, LOX/LH2, LOX/RP1, Solid, Storables

%Variables & Given Parameters
delta1 = 0.08;
delta2 = 0.08;
payload = 26000; %kg
%Specific Impulse
Isp_s2 = 366; %stage 2
%LCH4, LH2, RP1, solids, storables
Isp = [327, 366, 311, 269, 285]; %stage 1 varies


%initialize X (this is delta_V ratio/fraction)
X=0:0.001:1;

for isp = 1:length(Isp)
    %mass arrays
    cost_s1 = [];
    cost_s2 = [];
    cost_tot = [];
    min_gross_cost = realmax;

    for i = 1:length(X)
        %calling mass function to populate mass arrays
        [stageCost_nr1, stageCost_nr2] = cost_function(Isp(isp), Isp_s2, X(i), delta1, delta2);
        cost_s1 = [cost_s1 stageCost_nr1];
        cost_s2 = [cost_s2 stageCost_nr2];
        cost_tot = [cost_tot (stageCost_nr1 + stageCost_nr2)];

        %min gross mass function - returns the x,y minimum of the m_0 array
        if cost_s1(i) > 0 && cost_s2(i) > 0 && cost_tot(i) < min_gross_cost
            min_gross_cost = cost_tot(i); %convert to metric tons
            x_min = X(i);
        end
    end

    %millions to billions
    cost_s1 = cost_s1/1000;
    cost_s2 = cost_s2/1000;
    cost_tot = cost_tot/1000;
    min_gross_cost = min_gross_cost/1000;

    %output - min cost and corresponding deltaV
    fprintf('Stage 1 Isp = %d s --> Optimum at X = %.3f, Min Gross Cost = %.2f $B2025 \n', Isp(isp), x_min, min_gross_cost);

    %plotting
    plot(X, cost_s1);
    hold on;
    plot(X, cost_s2);
    plot(X, cost_tot);
    plot(x_min, min_gross_cost, '.', MarkerSize=20);
    ylim([0, 80]);
    title(sprintf('Stage 1 Isp = %d s', Isp(isp)));
    legend("First Stage Cost", "Second Stage Cost", "Gross Vehicle Cost")
    xlabel("Delta V Fraction");
    ylabel("Cost ($B2025)")
    hold off;

end
```

```
Stage 1 Isp = 327 s --> Optimum at X = 0.487, Min Gross Cost = 17.30 $B2025
Stage 1 Isp = 366 s --> Optimum at X = 0.540, Min Gross Cost = 14.31 $B2025
Stage 1 Isp = 311 s --> Optimum at X = 0.466, Min Gross Cost = 18.87 $B2025
Stage 1 Isp = 269 s --> Optimum at X = 0.414, Min Gross Cost = 24.61 $B2025
Stage 1 Isp = 285 s --> Optimum at X = 0.433, Min Gross Cost = 22.07 $B2025
```



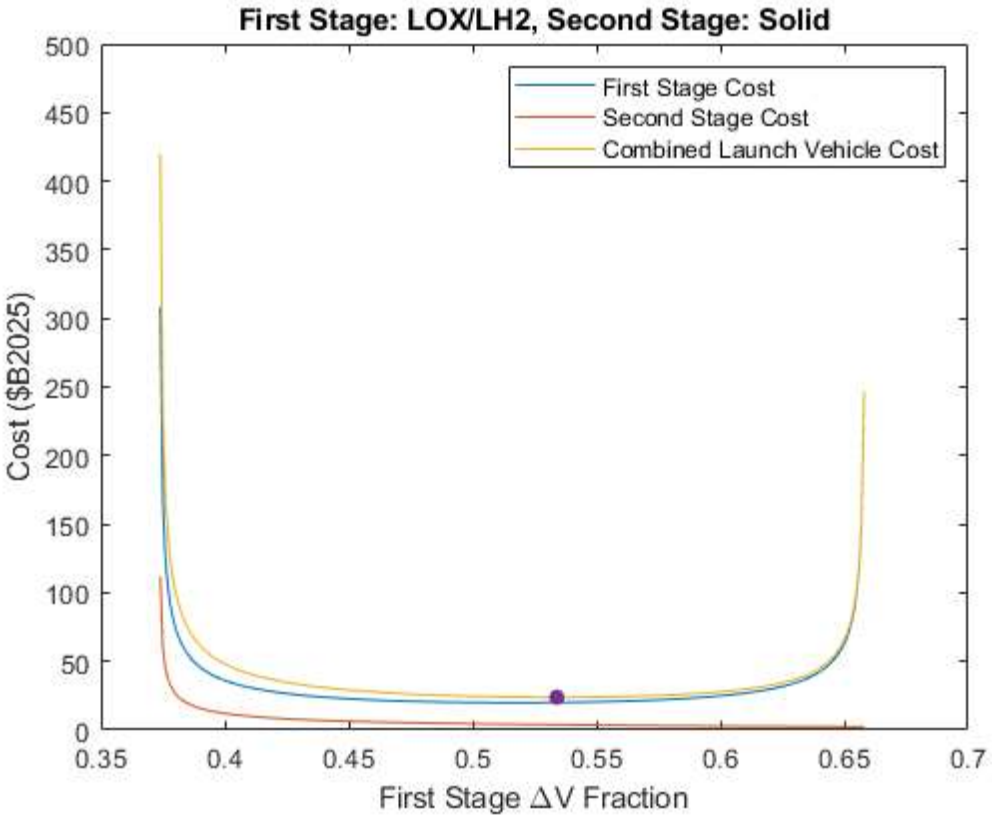
Published with MATLAB® R2024b

```
% solids row
% combo: first prop: LOX/LH2, second prop: solid
addpath('..')

% Set constants based on requirements and propellant combination
delta1 = 0.08;
delta2 = 0.08;
Isp1 = 366;
Isp2 = 269;
m_pl = 26000;

% Initialize/populate arrays
X = 0:0.001:1;
cost_stage1_arr = [];
cost_stage2_arr = [];

% For each X value, call cost_function and populate the arrays for cost of
% stage 1 and mass of stage 2
for i = 1:length(X)
    [stageCost_nr1, stageCost_nr2] = cost_function(Isp1, Isp2, X(i), delta1, delta2);
    cost_stage1_arr(end+1) = stageCost_nr1;
    cost_stage2_arr(end+1) = stageCost_nr2;
end

% Add the two arrays to get an array of the total cost
total_cost = cost_stage1_arr + cost_stage2_arr;

% Find the minimum gross mass, similar to min_cost_function, but save
% the associated X for the minimum total cost
min_cost = realmax;
for i = 1:length(total_cost)
    if ~isnan(total_cost(i)) && total_cost(i) < min_cost
        min_cost = total_cost(i);
        min_cost_X = X(i);
    end
end

% Plot mass of stage 1, mass of stage 2, and total mass in billions of dollars
% as a function of X. Also plot as a point the minimum total cost
plot(X, cost_stage1_arr/1000)
hold on
plot(X, cost_stage2_arr/1000)
plot(X, total_cost/1000)
plot(min_cost_X, min_cost/1000, '.', MarkerSize=20)
grid on
ylim([0, 50])
legend("First Stage Cost", "Second Stage Cost", "Combined Launch Vehicle Cost", "Minimum Cost")
xlabel("First Stage \DeltaV Fraction")
ylabel("Cost ($B2025)")
title("Cost vs /DeltaV Fraction for First Stage: LOX/LH2, Second Stage: Solids")
```



Published with MATLAB® R2024b

```
% LOX/RP1 row
% combo: first prop: LOX/LCH4, second prop: LOX/RPI
addpath('..')

% Set constants based on requirements and propellant combination
delta1 = 0.08;
delta2 = 0.08;
Isp1 = 327;
Isp2 = 311;
m_pl = 26000;

% Initialize/populate arrays
X = 0:0.001:1;
cost_stage1_arr = [];
cost_stage2_arr = [];

% For each X value, call cost_function and populate the arrays for cost of
% stage 1 and mass of stage 2
for i = 1:length(X)
    [stageCost_nr1, stageCost_nr2] = cost_function(Isp1, Isp2, X(i), delta1, delta2);
    cost_stage1_arr(end+1) = stageCost_nr1;
    cost_stage2_arr(end+1) = stageCost_nr2;
end

% Add the two arrays to get an array of the total cost
total_cost = cost_stage1_arr + cost_stage2_arr;

% Find the minimum gross mass, similar to min_cost_function, but save
% the associated X for the minimum total cost
min_cost = realmax;
for i = 1:length(total_cost)
    if ~isnan(total_cost(i)) && total_cost(i) < min_cost
        min_cost = total_cost(i);
        min_cost_X = X(i);
    end
end

% Plot mass of stage 1, mass of stage 2, and total mass in billions of dollars
% as a function of X. Also plot as a point the minimum total cost
plot(X, cost_stage1_arr/1000)
hold on
plot(X, cost_stage2_arr/1000)
plot(X, total_cost/1000)
plot(min_cost_X, min_cost/1000, '.', MarkerSize=20)
ylim([0, 500])
legend("First Stage Cost", "Second Stage Cost", "Combined Launch Vehicle Cost")
xlabel("First Stage \DeltaV Fraction")
ylabel("Cost ($B2025)")
title("First Stage: LOX/LH2, Second Stage: Solid")
```



Published with MATLAB® R2024b

```
solid = readmatrix('solid.csv', 'ThousandsSeparator', ',');
RP1 = readmatrix('RP1.csv', 'ThousandsSeparator', ',');
LCH4 = readmatrix('LCH4.csv', 'ThousandsSeparator', ',');
LH2 = readmatrix('LH2.csv', 'ThousandsSeparator', ',');
storable = readmatrix('storable.csv', 'ThousandsSeparator', ',');

LCH4_min_mass_MASS_solid = solid(1,1);
LH2_min_mass_MASS_solid  = solid(1,2);
RP1_min_mass_MASS_solid  = solid(1,3);
solid_min_mass_MASS_solid  = solid(1,4);
storable_min_mass_MASS_solid  = solid(1,5);

LCH4_min_mass_COST_solid  = solid(2,1);
LH2_min_mass_COST_solid  = solid(2,2);
RP1_min_mass_COST_solid  = solid(2,3);
solid_min_mass_COST_solid  = solid(2,4);
storable_min_mass_COST_solid  = solid(2,5);

LCH4_min_cost_COST_solid  = solid(3,1);
LH2_min_cost_COST_solid  = solid(3,2);
RP1_min_cost_COST_solid  = solid(3,3);
solid_min_cost_COST_solid  = solid(3,4);
storable_min_cost_COST_solid  = solid(3,5);

LCH4_min_cost_MASS_solid  = solid(4,1);
LH2_min_cost_MASS_solid  = solid(4,2);
RP1_min_cost_MASS_solid  = solid(4,3);
solid_min_cost_MASS_solid  = solid(4,4);
storable_min_cost_MASS_solid  = solid(4,5);

LCH4_min_mass_MASS_RP1 = RP1(1,1);
LH2_min_mass_MASS_RP1 = RP1(1,2);
RP1_min_mass_MASS_RP1 = RP1(1,3);
solid_min_mass_MASS_RP1 = RP1(1,4);
storable_min_mass_MASS_RP1 = RP1(1,5);

LCH4_min_mass_COST_RP1 = RP1(2,1);
LH2_min_mass_COST_RP1 = RP1(2,2);
RP1_min_mass_COST_RP1 = RP1(2,3);
solid_min_mass_COST_RP1 = RP1(2,4);
storable_min_mass_COST_RP1 = RP1(2,5);

LCH4_min_cost_COST_RP1 = RP1(3,1);
LH2_min_cost_COST_RP1 = RP1(3,2);
RP1_min_cost_COST_RP1 = RP1(3,3);
solid_min_cost_COST_RP1 = RP1(3,4);
storable_min_cost_COST_RP1 = RP1(3,5);

LCH4_min_cost_MASS_RP1 = RP1(4,1);
LH2_min_cost_MASS_RP1 = RP1(4,2);
RP1_min_cost_MASS_RP1 = RP1(4,3);
solid_min_cost_MASS_RP1 = RP1(4,4);
storable_min_cost_MASS_RP1 = RP1(4,5);

LCH4_min_mass_MASS_LCH4 = LCH4(1,1);
LH2_min_mass_MASS_LCH4 = LCH4(1,2);
RP1_min_mass_MASS_LCH4 = LCH4(1,3);
solid_min_mass_MASS_LCH4 = LCH4(1,4);
storable_min_mass_MASS_LCH4 = LCH4(1,5);

LCH4_min_mass_COST_LCH4 = LCH4(2,1);
LH2_min_mass_COST_LCH4 = LCH4(2,2);
RP1_min_mass_COST_LCH4 = LCH4(2,3);
solid_min_mass_COST_LCH4 = LCH4(2,4);
storable_min_mass_COST_LCH4 = LCH4(2,5);

LCH4_min_cost_COST_LCH4 = LCH4(3,1);
LH2_min_cost_COST_LCH4 = LCH4(3,2);
RP1_min_cost_COST_LCH4 = LCH4(3,3);
solid_min_cost_COST_LCH4 = LCH4(3,4);
storable_min_cost_COST_LCH4 = LCH4(3,5);

LCH4_min_cost_MASS_LCH4 = LCH4(4,1);
LH2_min_cost_MASS_LCH4 = LCH4(4,2);
RP1_min_cost_MASS_LCH4 = LCH4(4,3);
solid_min_cost_MASS_LCH4 = LCH4(4,4);
storable_min_cost_MASS_LCH4 = LCH4(4,5);

LCH4_min_mass_MASS_LH2 = LH2(1,1);
LH2_min_mass_MASS_LH2 = LH2(1,2);
RP1_min_mass_MASS_LH2 = LH2(1,3);
solid_min_mass_MASS_LH2 = LH2(1,4);
storable_min_mass_MASS_LH2 = LH2(1,5);

LCH4_min_mass_COST_LH2 = LH2(2,1);
```

```
LH2_min_mass_COST_LH2 = LH2(2,2);
RP1_min_mass_COST_LH2 = LH2(2,3);
solid_min_mass_COST_LH2 = LH2(2,4);
storable_min_mass_COST_LH2 = LH2(2,5);

LCH4_min_cost_COST_LH2 = LH2(3,1);
LH2_min_cost_COST_LH2 = LH2(3,2);
RP1_min_cost_COST_LH2 = LH2(3,3);
solid_min_cost_COST_LH2 = LH2(3,4);
storable_min_cost_COST_LH2 = LH2(3,5);

LCH4_min_cost_MASS_LH2 = LH2(4,1);
LH2_min_cost_MASS_LH2 = LH2(4,2);
RP1_min_cost_MASS_LH2 = LH2(4,3);
solid_min_cost_MASS_LH2 = LH2(4,4);
storable_min_cost_MASS_LH2 = LH2(4,5);

LCH4_min_mass_MASS_storable = storable(1,1);
LH2_min_mass_MASS_storable = storable(1,2);
RP1_min_mass_MASS_storable = storable(1,3);
solid_min_mass_MASS_storable = storable(1,4);
storable_min_mass_MASS_storable = storable(1,5);

LCH4_min_mass_COST_storable = storable(2,1);
LH2_min_mass_COST_storable = storable(2,2);
RP1_min_mass_COST_storable = storable(2,3);
solid_min_mass_COST_storable = storable(2,4);
storable_min_mass_COST_storable = storable(2,5);

LCH4_min_cost_COST_storable = storable(3,1);
LH2_min_cost_COST_storable = storable(3,2);
RP1_min_cost_COST_storable = storable(3,3);
solid_min_cost_COST_storable = storable(3,4);
storable_min_cost_COST_storable = storable(3,5);

LCH4_min_cost_MASS_storable = storable(4,1);
LH2_min_cost_MASS_storable = storable(4,2);
RP1_min_cost_MASS_storable = storable(4,3);
solid_min_cost_MASS_storable = storable(4,4);
storable_min_cost_MASS_storable = storable(4,5);


figure;
plot(LCH4_min_mass_MASS_solid, LCH4_min_mass_COST_solid, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
hold on
plot(LCH4_min_cost_MASS_solid, LCH4_min_cost_COST_solid, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')

plot(LH2_min_mass_MASS_solid, LH2_min_mass_COST_solid, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
plot(LH2_min_cost_MASS_solid, LH2_min_cost_COST_solid, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')

plot(RP1_min_mass_MASS_solid, RP1_min_mass_COST_solid, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
plot(RP1_min_cost_MASS_solid, RP1_min_cost_COST_solid, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')

plot(solid_min_mass_MASS_solid, solid_min_mass_COST_solid, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
plot(solid_min_cost_MASS_solid, solid_min_cost_COST_solid, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')

plot(storable_min_mass_MASS_solid, storable_min_mass_COST_solid, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
plot(storable_min_cost_MASS_solid, storable_min_cost_COST_solid, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')


plot(LCH4_min_mass_MASS_RP1, LCH4_min_mass_COST_RP1, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(LCH4_min_cost_MASS_RP1, LCH4_min_cost_COST_RP1, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(LH2_min_mass_MASS_RP1, LH2_min_mass_COST_RP1, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(LH2_min_cost_MASS_RP1, LH2_min_cost_COST_RP1, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(RP1_min_mass_MASS_RP1, RP1_min_mass_COST_RP1, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(RP1_min_cost_MASS_RP1, RP1_min_cost_COST_RP1, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(solid_min_mass_MASS_RP1, solid_min_mass_COST_RP1, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(solid_min_cost_MASS_RP1, solid_min_cost_COST_RP1, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(storable_min_mass_MASS_RP1, storable_min_mass_COST_RP1, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(storable_min_cost_MASS_RP1, storable_min_cost_COST_RP1, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(LCH4_min_mass_MASS_LCH4, LCH4_min_mass_COST_LCH4, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(LCH4_min_cost_MASS_LCH4, LCH4_min_cost_COST_LCH4, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')

plot(LH2_min_mass_MASS_LCH4, LH2_min_mass_COST_LCH4, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(LH2_min_cost_MASS_LCH4, LH2_min_cost_COST_LCH4, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')

plot(RP1_min_mass_MASS_LCH4, RP1_min_mass_COST_LCH4, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(RP1_min_cost_MASS_LCH4, RP1_min_cost_COST_LCH4, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')

plot(solid_min_mass_MASS_LCH4, solid_min_mass_COST_LCH4, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(solid_min_cost_MASS_LCH4, solid_min_cost_COST_LCH4, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
```

```
plot(storable_min_mass_MASS_LCH4, storable_min_mass_COST_LCH4, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(storable_min_cost_MASS_LCH4, storable_min_cost_COST_LCH4, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')

plot(LCH4_min_mass_MASS_LH2, LCH4_min_mass_COST_LH2, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(LCH4_min_cost_MASS_LH2, LCH4_min_cost_COST_LH2, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(LH2_min_mass_MASS_LH2, LH2_min_mass_COST_LH2, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(LH2_min_cost_MASS_LH2, LH2_min_cost_COST_LH2, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(RP1_min_mass_MASS_LH2, RP1_min_mass_COST_LH2, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(RP1_min_cost_MASS_LH2, RP1_min_cost_COST_LH2, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(solid_min_mass_MASS_LH2, solid_min_mass_COST_LH2, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(solid_min_cost_MASS_LH2, solid_min_cost_COST_LH2, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(storable_min_mass_MASS_LH2, storable_min_mass_COST_LH2, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(storable_min_cost_MASS_LH2, storable_min_cost_COST_LH2, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(LCH4_min_mass_MASS_storable, LCH4_min_mass_COST_storable, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(LCH4_min_cost_MASS_storable, LCH4_min_cost_COST_storable, 'o', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

plot(LH2_min_mass_MASS_storable, LH2_min_mass_COST_storable, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(LH2_min_cost_MASS_storable, LH2_min_cost_COST_storable, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

plot(RP1_min_mass_MASS_storable, RP1_min_mass_COST_storable, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(RP1_min_cost_MASS_storable, RP1_min_cost_COST_storable, 'x', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

plot(solid_min_mass_MASS_storable, solid_min_mass_COST_storable, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(solid_min_cost_MASS_storable, solid_min_cost_COST_storable, 'diamond', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

plot(storable_min_mass_MASS_storable, storable_min_mass_COST_storable, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(storable_min_cost_MASS_storable, storable_min_cost_COST_storable, '^', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

red = plot(nan, nan, 'o', 'MarkerFaceColor', 'r', 'MarkerEdgeColor', 'r', 'MarkerSize', 10);
blue = plot(nan, nan, 'o', 'MarkerFaceColor', 'b', 'MarkerEdgeColor', 'b', 'MarkerSize', 10);
green = plot(nan, nan, 'o', 'MarkerFaceColor', [0 52 0]/255, 'MarkerEdgeColor', [0 52 0]/255, 'MarkerSize', 10);
orange = plot(nan, nan, 'o', 'MarkerFaceColor', [255 153 51]/255, 'MarkerEdgeColor', [255 153 51]/255, 'MarkerSize', 10);
purple = plot(nan, nan, 'o', 'MarkerFaceColor', [204 0 204]/255, 'MarkerEdgeColor', [204 0 204]/255, 'MarkerSize', 10);

circle = plot(nan, nan, 'o', 'Color', 'k', 'MarkerSize', 10);
star = plot(nan, nan, '*', 'Color', 'k', 'MarkerSize', 10);
x = plot(nan, nan, 'x', 'Color', 'k', 'MarkerSize', 10);
diamond = plot(nan, nan, 'diamond', 'Color', 'k', 'MarkerSize', 10);
triangle = plot(nan, nan, '^', 'Color', 'k', 'MarkerSize', 10);

legend([circle, star, x, diamond, triangle, blue, purple, green, red, orange], 'First Stage: LOX/LCH4', 'First Stage: LOX/LH2', 'First Stage: LOX/RP1', 'First Stage

xlabel('Mass (t)')
ylabel('Cost ($B2025)')
title('Minimum Cost and Minimum Mass Propellant Mix Designs')
grid on
```

```
Error using readmatrix (line 171)
Unable to find or open 'solid.csv'. Check the path and filename or file permissions.

Error in full_plot (line 1)
solid = readmatrix('solid.csv', 'ThousandsSeparator', ',');
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

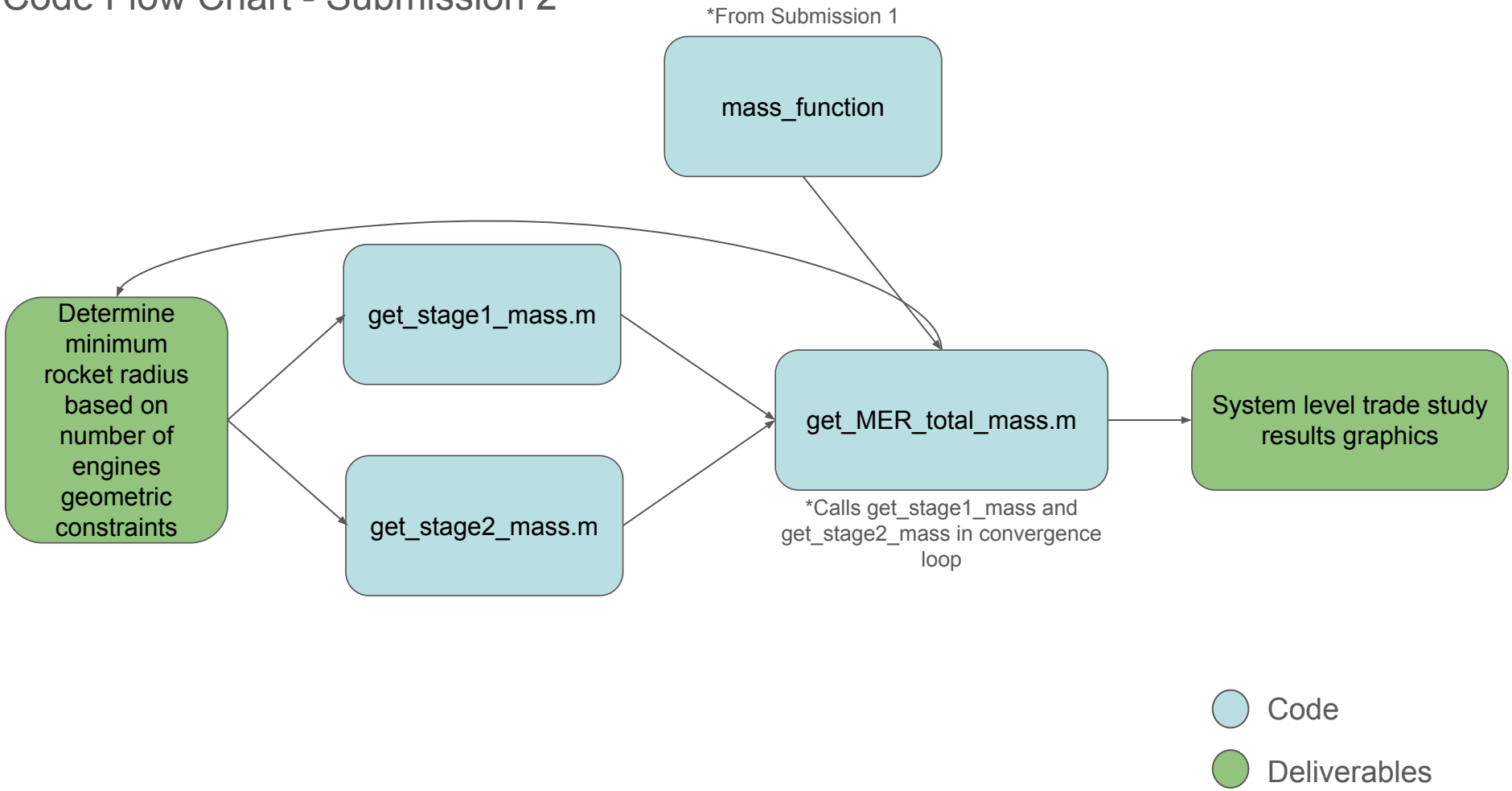
Published with MATLAB® R2024b

```
solid = readmatrix('solid.csv', 'ThousandsSeparator', ',');
RP1 = readmatrix('RP1.csv', 'ThousandsSeparator', ',');
LCH4 = readmatrix('LCH4.csv', 'ThousandsSeparator', ',');
LH2 = readmatrix('LH2.csv', 'ThousandsSeparator', ',');
storable = readmatrix('storable.csv', 'ThousandsSeparator', ',');

LH2_min_mass_MASS_solid  = solid(1,2);
LH2_min_mass_COST_solid  = solid(2,2);
LH2_min_cost_COST_solid  = solid(3,2);
LH2_min_cost_MASS_solid  = solid(4,2);

LH2_min_mass_MASS_RP1 = RP1(1,2);
LH2_min_mass_COST_RP1 = RP1(2,2);
LH2_min_cost_COST_RP1 = RP1(3,2);
LH2_min_cost_MASS_RP1 = RP1(4,2);

LH2_min_mass_MASS_LCH4 = LCH4(1,2);
LH2_min_mass_COST_LCH4 = LCH4(2,2);
LH2_min_cost_COST_LCH4 = LCH4(3,2);
LH2_min_cost_MASS_LCH4 = LCH4(4,2);

LH2_min_mass_MASS_LH2 = LH2(1,2);
LH2_min_mass_COST_LH2 = LH2(2,2);
LH2_min_cost_COST_LH2 = LH2(3,2);
LH2_min_cost_MASS_LH2 = LH2(4,2);

LH2_min_mass_MASS_storable = storable(1,2);
LH2_min_mass_COST_storable = storable(2,2);
LH2_min_cost_COST_storable = storable(3,2);
LH2_min_cost_MASS_storable = storable(4,2);

figure;
plot(LH2_min_mass_MASS_solid, LH2_min_mass_COST_solid, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')
hold on
plot(LH2_min_cost_MASS_solid, LH2_min_cost_COST_solid, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'r')

plot(LH2_min_mass_MASS_RP1, LH2_min_mass_COST_RP1, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)
plot(LH2_min_cost_MASS_RP1, LH2_min_cost_COST_RP1, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [0 52 0]/255)

plot(LH2_min_mass_MASS_LCH4, LH2_min_mass_COST_LCH4, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')
plot(LH2_min_cost_MASS_LCH4, LH2_min_cost_COST_LCH4, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', 'b')

plot(LH2_min_mass_MASS_LH2, LH2_min_mass_COST_LH2, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)
plot(LH2_min_cost_MASS_LH2, LH2_min_cost_COST_LH2, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [204 0 204]/255)

plot(LH2_min_mass_MASS_storable, LH2_min_mass_COST_storable, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)
plot(LH2_min_cost_MASS_storable, LH2_min_cost_COST_storable, '*', 'MarkerSize', 15, 'LineWidth', 2, 'Color', [255 153 51]/255)

red = plot(nan, nan, 'o', 'MarkerFaceColor', 'r', 'MarkerEdgeColor', 'r', 'MarkerSize', 10);
blue = plot(nan, nan, 'o', 'MarkerFaceColor', 'b', 'MarkerEdgeColor', 'b', 'MarkerSize', 10);
green = plot(nan, nan, 'o', 'MarkerFaceColor', [0 52 0]/255, 'MarkerEdgeColor', [0 52 0]/255, 'MarkerSize', 10);
orange = plot(nan, nan, 'o', 'MarkerFaceColor', [255 153 51]/255, 'MarkerEdgeColor', [255 153 51]/255, 'MarkerSize', 10);
purple = plot(nan, nan, 'o', 'MarkerFaceColor', [204 0 204]/255, 'MarkerEdgeColor', [204 0 204]/255, 'MarkerSize', 10);

circle = plot(nan, nan, 'o', 'Color', 'k', 'MarkerSize', 10);
star = plot(nan, nan, '*', 'Color', 'k', 'MarkerSize', 10);
x = plot(nan, nan, 'x', 'Color', 'k', 'MarkerSize', 10);
diamond = plot(nan, nan, 'diamond', 'Color', 'k', 'MarkerSize', 10);
triangle = plot(nan, nan, '^', 'Color', 'k', 'MarkerSize', 10);

legend([circle, star, x, diamond, triangle, blue, purple, green, red, orange], 'First Stage: LOX/LCH4', 'First Stage: LOX/LH2', 'First Stage: LOX/RP1', 'First Stage

xlabel('Mass (t)')
ylabel('Cost ($B2025)')
title('Minimum Cost and Minimum Mass Propellant Mix Designs')
grid on
```

```
Error using readmatrix (line 171)
Unable to find or open 'solid.csv'. Check the path and filename or file permissions.

Error in best_combos_plot (line 1)
solid = readmatrix('solid.csv', 'ThousandsSeparator', ',');
        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

Published with MATLAB® R2024b

Code Flow Chart - Submission 2



Exported Code - Team #15 Submission 2

Note: The MATLAB functions throw an error when being published to a PDF because the function is not being called within each MATLAB file. So, there is an error in each of these files that should be ignored. The error is not thrown when properly called.

Contents


```
function [num_engines_stage1, num_engines_stage2, stage1_only_total_mass, stage2_only_total_mass, total_mass, total_height, stage1_T_to_W, stage2_T_to_W] = get_MER_
```

```
    addpath("..\vehicle_level_analysis_tool\")

    % Set thrust to weight constants
    T_to_W_first = 1.3;
    T_to_W_second = 0.76;

    % Set delta, g0, payload mass, total deltaV, and tolerance constants
    delta1 = 0.08;
    delta2 = 0.08;
    g0 = 9.81; % m/s^2
    deltaV = 12300; % m/s
    M_l = 26000; % kg
    tol = 0.01;

    % Logic to set first stage Isp and thrust
    if first_stage == "LCH4"
        stage1_Isp = 327; % s
        stage1_thrust = 2.26e6; % N
    elseif first_stage == "LH2"
        stage1_Isp = 366; % s
        stage1_thrust = 1.86e6; % N
    elseif first_stage == "RP1"
        stage1_Isp = 311; % s
        stage1_thrust = 1.92e6; % N
    elseif first_stage == "solid"
        stage1_Isp = 269; % s
        stage1_thrust = 4.5e6; % N
    elseif first_stage == "storables"
        stage1_Isp = 285; % s
        stage1_thrust = 1.75e6; % N
    end

    % Logic to set second stage Isp and thrust
    if second_stage == "LCH4"
        stage2_Isp = 327; % s
        stage2_thrust = 0.745e6; % N
    elseif second_stage == "LH2"
        stage2_Isp = 366;
        stage2_thrust = 0.099e6; % N
    elseif second_stage == "RP1"
        stage2_Isp = 311; % s
        stage2_thrust = 0.061e6; % N
    elseif second_stage == "solid"
        stage2_Isp = 269; % s
        stage2_thrust = 2.94e6; % N
    elseif second_stage == "storables"
        stage2_Isp = 285; % s
        stage2_thrust = 0.067e6; % N
    end
```

```
Not enough input arguments.

Error in get_MER_total_mass (line 17)
    if first_stage == "LCH4"
       ^^^^^^^^^^
```

SECOND STAGE COMPUTATIONS %%

Get initial guess from mass_function

```
    deltaV2_frac = deltaV*(1-X);
    r = exp(-deltaV2_frac/(g0*stage2_Isp));
    [m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(stage1_Isp, stage2_Isp, X, delta1, delta2);

    % Set propellant and total mass (with mass margin)
    M_p = m_pr2;
    M_0 = 1.3*m_in2 + M_l + M_p;

    % Call get_stage2_mass for initial guess using guess from mass_function
    stage2_total_mass = get_stage2_mass(second_stage, M_p, M_0, 1, true);

    % Compute initial guess for number of engines
    num_engines_stage2 = ceil(stage2_total_mass*g0*T_to_W_second/stage2_thrust);
```

```
% Set single engine thrust and residual that will be continuously
% checked in convergence loop
stage2_thrust_single = stage2_thrust;
residual = realmax;

if second_stage ~= "solid"
    while residual > tol

        % Set M_p and M_0 using mass margin
        M_p = stage2_total_mass*(1-r);
        M_0 = (stage2_total_mass - M_p- M_l)*1.3 + M_l + M_p;

        % Call get_stage2_mass, compute number of engines considering
        % mass margin, compute residual
        [stage2_total_mass, stage2_height] = get_stage2_mass(second_stage, M_p, M_0, num_engines_stage2, false);
        margin_stage2_total_mass = (stage2_total_mass - M_p - M_l)*1.3 + M_l + M_p;
        num_engines_stage2 = ceil(margin_stage2_total_mass*g0*T_to_W_second/stage2_thrust_single);
        residual = abs(margin_stage2_total_mass - M_0);
    end
else
    % Slightly different for solids, not an convergence loop
    % Compute mass with margin and required engines. Run
    % get_stage2_mass. Check to see if new required number of engines
    % is the same as before (most likely will be). If not, make it the
    % new number of engines that satisfy thrust to weight ratio
    margin_stage2_total_mass = (stage2_total_mass - M_l- M_p)*1.3 + M_l + M_p;
    num_engines_required = ceil(margin_stage2_total_mass*g0*T_to_W_second/stage2_thrust_single);
    [stage2_total_mass, stage2_height] = get_stage2_mass(second_stage, M_p, M_0, num_engines_required, false);
    margin_stage2_total_mass = (stage2_total_mass - M_l - M_p)*1.3 + M_l + M_p;
    num_engines_required_recompued = ceil(margin_stage2_total_mass*g0*T_to_W_second/stage2_thrust_single);
    if ceil(num_engines_required) ~= ceil(num_engines_required_recompued)
        num_engines_stage2 = ceil(num_engines_required_recompued);
    else
        num_engines_stage2 = ceil(num_engines_required);
    end
end
```

FIRST STAGE COMPUTATIONS %%

Get initial guess from mass_function

```
deltaV1_frac = deltaV*X;
r = exp(-deltaV1_frac/(g0*stage1_Isp));
[m_in1, m_in2, m_pr1, m_pr2, m0] = mass_function(stage1_Isp, stage2_Isp, X, delta1, delta2);

% Set propellant and total mass (with mass margin)
M_p = m_pr1;
M_0 = (m0 - M_l - m_pr1)*1.3 + M_l + m_pr1;

% Call get_stage1_mass for initial guess using guess from mass_function
stage1_total_mass = get_stage1_mass(first_stage, M_p, M_0, stage2_total_mass, 1, true);

% Compute initial guess for number of engines
num_engines_stage1 = stage1_total_mass*g0*T_to_W_first/stage1_thrust;

% Set single engine thrust and residual that will be continuously
stage1_thrust_single = stage1_thrust;
residual = realmax;

if first_stage ~= "solid"
    while residual > tol

        % Set M_p and M_0 using mass margin
        M_p = stage1_total_mass*(1-r);
        M_0 = (stage1_total_mass - M_p - margin_stage2_total_mass)*1.3 + margin_stage2_total_mass + M_p;

        % Call get_stage1_mass, compute number of engines considering
        % mass margin and stage2 mass, compute residual
        [stage1_total_mass, stage1_height] = get_stage1_mass(first_stage, M_p, M_0, margin_stage2_total_mass, num_engines_stage1, false);
        margin_stage1_total_mass = (stage1_total_mass - M_p - margin_stage2_total_mass)*1.3 + margin_stage2_total_mass + M_p;
        num_engines_stage1 = ceil(margin_stage1_total_mass*g0*T_to_W_first/stage1_thrust_single);
        residual = abs(margin_stage1_total_mass - M_0);
    end
else
    % Slightly different for solids, not an convergence loop
    % Compute mass with margin and required engines. Run
    % get_stage2_mass. Check to see if new required number of engines
    % is the same as before (most likely will be). If not, make it the
    % new number of engines that satisfy thrust to weight ratio
    margin_stage1_total_mass = (stage1_total_mass - M_p - margin_stage2_total_mass)*1.3 + margin_stage2_total_mass + M_p;
    num_engines_required = ceil(margin_stage1_total_mass*g0*T_to_W_first/stage1_thrust_single);
    [stage1_total_mass, stage1_height] = get_stage1_mass(first_stage, M_p, M_0, stage2_total_mass, num_engines_required, false);
    margin_stage1_total_mass = (stage1_total_mass - M_p - margin_stage2_total_mass)*1.3 + margin_stage2_total_mass + M_p;
    num_engines_required_recompued = ceil(margin_stage1_total_mass*g0*T_to_W_first/stage1_thrust_single);
    if ceil(num_engines_required) ~= ceil(num_engines_required_recompued)
        num_engines_stage1 = ceil(num_engines_required_recompued);
```

```
        else
            num_engines_stage1 = ceil(num_engines_required);
        end
    end

    % Compute total height
    total_height = stage1_height + stage2_height;

    % Compute avionics mass and set as workspace variable
    mass_avionics = 10*stage1_total_mass^(0.361);
    assignin('base', 'mass_avionics', mass_avionics);

    % Compute stage only masses
    stage1_only_total_mass = margin_stage1_total_mass - margin_stage2_total_mass - M_l + mass_avionics*1.3;
    stage2_only_total_mass = margin_stage2_total_mass - M_l + mass_avionics*1.3;

    % Compute total masses
    stage1_total_mass = margin_stage1_total_mass + mass_avionics*1.3;
    stage2_total_mass = margin_stage2_total_mass + mass_avionics*1.3;
    total_mass = stage1_total_mass;

    % Get number of engines per stage
    num_engines_stage1 = ceil(num_engines_stage1);
    num_engines_stage2 = ceil(num_engines_stage2);

    % Output thrust to weight to ensure it meets requirement
    stage2_T_to_W = num_engines_stage2*stage2_thrust_single/g0/stage2_total_mass;
    stage1_T_to_W = num_engines_stage1*stage1_thrust_single/g0/stage1_total_mass;
```

```
end
```

Published with MATLAB® R2024b

```
function [stage1_total_mass, stage1_height] = get_stage1_mass(first_stage, M_p, M_0, stage2_total_mass, num_engines, init)

    % Set density constants
    rho_LH2 = 71;
    rho_LOX = 1140;
    rho_RP1 = 820;
    rho_LCH4 = 423;
    rho_solid = 1680;
    rho_N2O4 = 1442;
    rho_UDMH = 791;

    % Set tank constants
    radius = 6.4; % m
    cap_height = 1; % m
    payload_cone_height = 10; % m
    payload_cyl_height = 10; % m
    engine_space = 3; % m

    % Constants that depend on propellant choice
    if first_stage == "LCH4"
        stage1_ratio = 3.6;
        stage1_oxidizer_rho = rho_LOX;
        stage1_fuel_rho = rho_LCH4;
        stage1_thrust_single = 2.26e6; % N
        stage1_nozzle_exp = 34.34;
        chamber_pressure_1 = 35.16e6; % Pa
    elseif first_stage == "LH2"
        stage1_ratio = 6.03;
        stage1_oxidizer_rho = rho_LOX;
        stage1_fuel_rho = rho_LH2;
        stage1_thrust_single = 1.86e6; % N
        stage1_nozzle_exp = 78;
        chamber_pressure_1 = 20.64e6; % Pa
    elseif first_stage == "RP1"
        stage1_ratio = 2.72;
        stage1_oxidizer_rho = rho_LOX;
        stage1_fuel_rho = rho_RP1;
        stage1_thrust_single = 1.92e6; % N
        stage1_nozzle_exp = 37;
        chamber_pressure_1 = 25.8e6; % Pa
    elseif first_stage == "solid"
        stage1_thrust_single = 4.5e6; % N
        stage1_nozzle_exp = 16;
        chamber_pressure_1 = 10.5e6; % Pa
    elseif first_stage == "storables"
        stage1_ratio = 2.67;
        stage1_oxidizer_rho = rho_N2O4;
        stage1_fuel_rho = rho_UDMH;
        stage1_thrust_single = 1.75e6; % N
        stage1_nozzle_exp = 26.2;
        chamber_pressure_1 = 15.7e6; % Pa
    end

    % Set thrust according to number of engines
    if init
        stage1_thrust = stage1_thrust_single;
    else
        stage1_thrust = stage1_thrust_single*num_engines;
    end

    % Compute stage1 tank mass
    if first_stage == "solid"
        solid_volume = M_p/rho_solid;
        stage1_tank_mass = 12.16*solid_volume;
    else
        mass_split = M_p/(stage1_ratio+1);
        mass_oxidizer = stage1_ratio*mass_split;
        mass_fuel = mass_split;
        volume_oxidizer = mass_oxidizer/stage1_oxidizer_rho;
        volume_fuel = mass_fuel/stage1_fuel_rho;
        if first_stage == "LH2"
            stage1_tank_mass = 12.16*volume_oxidizer + 9.09*volume_fuel;
        else
            stage1_tank_mass = 12.16*(volume_oxidizer + volume_fuel);
        end
    end

    % Compute stage1 tank volume assuming cylinder and two sphere caps 1m
    % tall each
    if first_stage ~= "solid"
        ox_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        ox_cyl_vol = volume_oxidizer - ox_cap_vol;
        ox_cyl_height = ox_cyl_vol/(pi*radius^2);
        ox_cap_surf_area = 2*(pi*(radius^2 + cap_height^2));
        ox_cyl_surf_area = 2*pi*radius*ox_cyl_height;
```

```
        fuel_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        fuel_cyl_vol = volume_fuel - fuel_cap_vol;
        fuel_cyl_height = fuel_cyl_vol/(pi*radius^2);
        fuel_cap_surf_area = 2*(pi*(radius^2 + cap_height^2));
        fuel_cyl_surf_area = 2*pi*radius*fuel_cyl_height;
    end

    % Compute insulation from tank volume, edge cases for storables and
    % solids
    if first_stage ~= "solid" && first_stage ~= "storables"
        LOX_stage1_insulation_mass = 1.123*(ox_cap_surf_area + ox_cyl_surf_area);
        if first_stage == "LH2"
            LH2_stage1_insulation_mass = 2.88*(fuel_cap_surf_area + fuel_cyl_surf_area);
            stage1_insulation_mass = LOX_stage1_insulation_mass + LH2_stage1_insulation_mass;
        elseif first_stage == "LCH4"
            LCH4_stage1_insulation_mass = 1.123*(fuel_cap_surf_area + fuel_cyl_surf_area);
            stage1_insulation_mass = LOX_stage1_insulation_mass + LCH4_stage1_insulation_mass;
        else
            stage1_insulation_mass = LOX_stage1_insulation_mass;
        end
    elseif first_stage == "solid"
        stage1_insulation_mass = 0;
        solid_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        solid_cyl_vol = solid_volume - solid_cap_vol;
        solid_cyl_height = solid_cyl_vol/(pi*radius^2);
    else
        stage1_insulation_mass = 0;
    end

    % Set engine and casing mass, dependent on propellant
    if first_stage ~= "solid"
        stage1_engine_mass = 7.81e-4*stage1_thrust + 3.37e-5*stage1_thrust*sqrt(stage1_nozzle_exp) + 59;
        stage1_casing_mass = 0;
    else
        stage1_engine_mass = 0;
        stage1_casing_mass = 0.135*M_p;
    end

    % Compute payload and aft fairing areas
    interstage_fairing_area = 2*pi*radius*(engine_space + cap_height);
    aft2_fairing_area = 2*pi*radius*(engine_space + cap_height);

    interstage_fairing_mass = 4.95*interstage_fairing_area^(1.15);
    stage1_aft_fairing_mass = 4.95*aft2_fairing_area^(1.15);

    % Compute intertank fairing mass and overall height dependent on propellant
    if first_stage ~= "solid"
        intertank2_fairing_area = 2*pi*radius*(2*cap_height);
        stage1_intertank_fairing_mass = 4.95*intertank2_fairing_area^(1.15);
        stage1_height = payload_cone_height + payload_cyl_height + 4*cap_height + ox_cyl_height + fuel_cyl_height + engine_space;
    else
        stage1_intertank_fairing_mass = 0;
        stage1_height = payload_cone_height + payload_cyl_height + 2*cap_height + solid_cyl_height + engine_space;
    end

    % Compute wiring, thrust structure, and gimbals masses
    stage1_mass_wiring = 1.058*sqrt(M_0)*stage1_height^(0.25);

    stage1_mass_thrust_struct = 2.25e-4*stage1_thrust;

    stage1_mass_gimbals = 237.8*(stage1_thrust/chamber_pressure_1)^(0.9375);

    % Compute total mass
    stage1_total_mass = M_p + stage1_mass_wiring + stage1_tank_mass + stage1_insulation_mass + stage1_engine_mass + stage1_mass_thrust_struct + stage1_casing_mass +

    % Assign workspace variables
    assignin('base', 'stage1_propellant_mass', M_p);
    assignin('base', 'stage1_tank_mass', stage1_tank_mass);
    assignin('base', 'stage1_mass_wiring', stage1_mass_wiring);
    assignin('base', 'stage1_insulation_mass', stage1_insulation_mass);
    assignin('base', 'stage1_engine_mass', stage1_engine_mass);
    assignin('base', 'stage1_mass_thrust_struct', stage1_mass_thrust_struct);
    assignin('base', 'stage1_casing_mass', stage1_casing_mass);
    assignin('base', 'stage1_mass_gimbals', stage1_mass_gimbals);
    assignin('base', 'interstage_fairing_mass', interstage_fairing_mass);
    assignin('base', 'stage1_intertank_fairing_mass', stage1_intertank_fairing_mass);
    assignin('base', 'stage1_aft_fairing_mass', stage1_aft_fairing_mass);

end
```

```
Not enough input arguments.

Error in get_stage1_mass (line 20)
```

```
    if first_stage == "LCH4"
       ^^^^^^^^^^^
```

Published with MATLAB® R2024b

```
function [stage2_total_mass, stage2_height] = get_stage2_mass(second_stage, M_p, M_0, num_engines, init)

    % Set density constants
    rho_LH2 = 71;
    rho_LOX = 1140;
    rho_RP1 = 820;
    rho_LCH4 = 423;
    rho_solid = 1680;
    rho_N2O4 = 1442;
    rho_UDMH = 791;

    % Set tank constants
    M_l = 26000; % kg
    radius = 6.4; % m
    cap_height = 1; % m
    payload_cone_height = 10; % m
    payload_cyl_height = 10; % m
    engine_space = 3; % m

    % Constants that depend on propellant choice
    if second_stage == "LCH4"
        stage2_ratio = 3.6;
        stage2_oxidizer_rho = rho_LOX;
        stage2_fuel_rho = rho_LCH4;
        stage2_thrust_single = 0.745e6; % N
        stage2_nozzle_exp = 45;
        chamber_pressure_2 = 10.1e6; % Pa
    elseif second_stage == "LH2"
        stage2_ratio = 6.03;
        stage2_oxidizer_rho = rho_LOX;
        stage2_fuel_rho = rho_LH2;
        stage2_thrust_single = 0.099e6; % N
        stage2_nozzle_exp = 84;
        chamber_pressure_2 = 4.2e6; % Pa
    elseif second_stage == "RP1"
        stage2_ratio = 2.72;
        stage2_oxidizer_rho = rho_LOX;
        stage2_fuel_rho = rho_RP1;
        stage2_thrust_single = 0.061e6; % N
        stage2_nozzle_exp = 14.5;
        chamber_pressure_2 = 6.77e6; % Pa
    elseif second_stage == "solid"
        stage2_thrust_single = 2.94e6; % N
        stage2_nozzle_exp = 56;
        chamber_pressure_2 = 5e6; % Pa
    elseif second_stage == "storables"
        stage2_ratio = 2.67;
        stage2_oxidizer_rho = rho_N2O4;
        stage2_fuel_rho = rho_UDMH;
        stage2_thrust_single = 0.067e6; % N
        stage2_nozzle_exp = 81.3;
        chamber_pressure_2 = 14.7e6; % Pa
    end

    % Set thrust according to number of engines
    if init
        stage2_thrust = stage2_thrust_single;
    else
        stage2_thrust = stage2_thrust_single*num_engines;
    end

    % Compute stage2 tank mass
    if second_stage == "solid"
        solid_volume = M_p/rho_solid;
        stage2_tank_mass = 12.16*solid_volume;
    else
        mass_split = M_p/(stage2_ratio+1);
        mass_oxidizer = stage2_ratio*mass_split;
        mass_fuel = mass_split;
        volume_oxidizer = mass_oxidizer/stage2_oxidizer_rho;
        volume_fuel = mass_fuel/stage2_fuel_rho;
        if second_stage == "LH2"
            stage2_tank_mass = 12.16*volume_oxidizer + 9.09*volume_fuel;
        else
            stage2_tank_mass = 12.16*(volume_oxidizer + volume_fuel);
        end
    end

    % Compute stage2 tank volume assuming cylinder and two sphere caps 1m
    % tall each
    if second_stage ~= "solid"
        ox_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        ox_cyl_vol = volume_oxidizer - ox_cap_vol;
        ox_cyl_height = ox_cyl_vol/(pi*radius^2);
        ox_cap_surf_area = 2*(pi*(radius^2 + cap_height^2));
```

```
        ox_cyl_surf_area = 2*pi*radius*ox_cyl_height;

        fuel_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        fuel_cyl_vol = volume_fuel - fuel_cap_vol;
        fuel_cyl_height = fuel_cyl_vol/(pi*radius^2);
        fuel_cap_surf_area = 2*(pi*(radius^2 + cap_height^2));
        fuel_cyl_surf_area = 2*pi*radius*fuel_cyl_height;
    end

    % Compute insulation from tank volume, edge cases for storables and
    % solids
    if second_stage ~= "solid" && second_stage ~= "storables"
        LOX_stage2_insulation_mass = 1.123*(ox_cap_surf_area + ox_cyl_surf_area);
        if second_stage == "LH2"
            LH2_stage2_insulation_mass = 2.88*(fuel_cap_surf_area + fuel_cyl_surf_area);
            stage2_insulation_mass = LOX_stage2_insulation_mass + LH2_stage2_insulation_mass;
        elseif second_stage == "LCH4"
            LCH4_stage2_insulation_mass = 1.123*(fuel_cap_surf_area + fuel_cyl_surf_area);
            stage2_insulation_mass = LOX_stage2_insulation_mass + LCH4_stage2_insulation_mass;
        else
            stage2_insulation_mass = LOX_stage2_insulation_mass;
        end
    elseif second_stage == "solid"
        stage2_insulation_mass = 0;
        solid_cap_vol = 2*(pi*cap_height)*(3*radius^2 + cap_height^2)/6;
        solid_cyl_vol = solid_volume - solid_cap_vol;
        solid_cyl_height = solid_cyl_vol/(pi*radius^2);
    else
        stage2_insulation_mass = 0;
    end

    % Set engine and casing mass, dependent on propellant
    if second_stage ~= "solid"
        stage2_engine_mass = 7.81e-4*stage2_thrust + 3.37e-5*stage2_thrust*sqrt(stage2_nozzle_exp) + 59;
        stage2_casing_mass = 0;
    else
        stage2_engine_mass = 0;
        stage2_casing_mass = 0.135*M_p;
    end

    % Compute payload and aft fairing areas
    payload_fairing_area = pi*radius*sqrt(radius^2 + payload_cone_height^2) + 2*pi*radius*payload_cyl_height;
    aft2_fairing_area = 2*pi*radius*(engine_space + cap_height);

    payload_fairing_mass = 4.95*payload_fairing_area^(1.15);
    stage2_aft_fairing_mass = 4.95*aft2_fairing_area^(1.15);

    % Compute intertank fairing mass and overall height dependent on propellant
    if second_stage ~= "solid"
        intertank2_fairing_area = 2*pi*radius*(2*cap_height);
        stage2_intertank2_fairing_mass = 4.95*intertank2_fairing_area^(1.15);
        stage2_height = payload_cone_height + payload_cyl_height + 4*cap_height + ox_cyl_height + fuel_cyl_height + engine_space;
    else
        stage2_intertank2_fairing_mass = 0;
        stage2_height = payload_cone_height + payload_cyl_height + 2*cap_height + solid_cyl_height + engine_space;
    end

    % Compute wiring, thrust structure, and gimbals masses
    stage2_mass_wiring = 1.058*sqrt(M_0)*stage2_height^(0.25);

    stage2_mass_thrust_struct = 2.25e-4*stage2_thrust;

    stage2_mass_gimbals = 237.8*(stage2_thrust/chamber_pressure_2)^(0.9375);

    % Compute total mass
    stage2_total_mass = M_p + stage2_mass_wiring + stage2_tank_mass + stage2_insulation_mass + stage2_engine_mass + stage2_mass_thrust_struct + stage2_casing_mass +

    % Assign workspace variables
    assignin('base', 'stage2_propellant_mass', M_p);
    assignin('base', 'stage2_tank_mass', stage2_tank_mass);
    assignin('base', 'stage2_mass_wiring', stage2_mass_wiring);
    assignin('base', 'stage2_insulation_mass', stage2_insulation_mass);
    assignin('base', 'stage2_engine_mass', stage2_engine_mass);
    assignin('base', 'stage2_mass_thrust_struct', stage2_mass_thrust_struct);
    assignin('base', 'stage2_casing_mass', stage2_casing_mass);
    assignin('base', 'stage2_mass_gimbals', stage2_mass_gimbals);
    assignin('base', 'payload_fairing_mass', payload_fairing_mass);
    assignin('base', 'stage2_intertank2_fairing_mass', stage2_intertank2_fairing_mass);
    assignin('base', 'stage2_aft_fairing_mass', stage2_aft_fairing_mass);

end
```

```
Not enough input arguments.

Error in get_stage2_mass (line 21)
```

```
    if second_stage == "LCH4"
       ^^^^^^^^^^^^
```

Published with MATLAB® R2024b