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_
```

```
    % Sophie, Spyros, Chris
    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 18)
    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