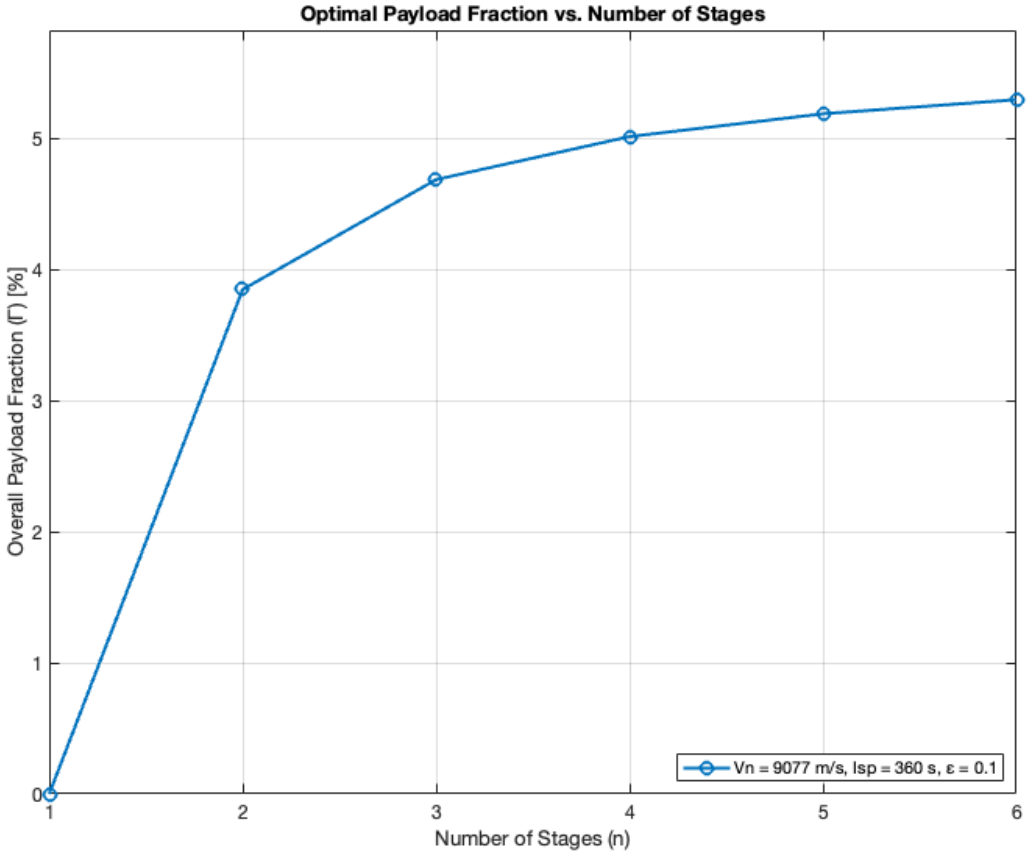
```
%=========================================================================
% Analysis: Payload Fraction vs. Number of Stages
%=========================================================================
clear; clc; close all;

% Define universal constants and vehicle parameters
g0 = 9.80665; % Standard gravity in m/s^2

% Define a constant "high-performance" launch vehicle technology
% Using parameters from the PDF's Section 8.3 example
Vn_req = 9077;      % Required Delta-V to LEO, m/s
Isp_tech = 360;     % Specific Impulse, sec
epsilon_tech = 0.1; % Structural Coefficient

% We will test for 1 to 6 stages
num_stages_vec = 1:6;
payload_fraction_vec = zeros(size(num_stages_vec));

fprintf('Calculating optimal payload fraction for different numbers of
stages...\n');
fprintf('Vehicle Technology: Isp = %d s, Epsilon = %.2f\n', Isp_tech,
epsilon_tech);
fprintf('Mission Requirement: Delta-V = %d m/s\n\n', Vn_req);

% Loop through each number of stages
for i = 1:length(num_stages_vec)
    n = num_stages_vec(i);
    fprintf('Testing n = %d stage(s): ', n);

    % Use a try-catch block in case the mission is impossible for a given n
    try
        [~, ~, ~, Gamma] = optimizeRocket(Vn_req, n, Isp_tech, epsilon_tech,
g0);
        payload_fraction_vec(i) = Gamma;
        fprintf('Optimal Payload Fraction = %.4f\n', Gamma);
    catch ME
        % If optimizeRocket throws an error (mission impossible), Gamma is 0
        payload_fraction_vec(i) = 0;
        fprintf('Mission is impossible. %s\n', ME.message);
    end
end

% --- Plotting the Results ---
figure('Name', 'Payload Fraction vs. Number of Stages', 'Position', [100,
100, 800, 600]);
plot(num_stages_vec, payload_fraction_vec * 100, 'o-', 'LineWidth', 2,
'MarkerSize', 8);
grid on;
title('Optimal Payload Fraction vs. Number of Stages');
xlabel('Number of Stages (n)');
ylabel('Overall Payload Fraction (Γ) [%]');
xticks(num_stages_vec);
```

```
ylim([0, max(payload_fraction_vec * 100) * 1.1]);
legend(sprintf('Vn = %d m/s, Isp = %d s, ε = %.1f', Vn_req, Isp_tech,
epsilon_tech), 'Location', 'southeast');
set(gca, 'FontSize', 12);

fprintf('\nPlot generated successfully.\n');

% --- Add the optimizeRocket function at the end ---
% (The same function from the previous answer is needed here)
function [alpha, lambda, R, Gamma] = optimizeRocket(Vn_req, n, Isp, epsilon,
g0)
    if isscalar(Isp), Isp = ones(1, n) * Isp; elseif length(Isp) ~= n,
error('Isp vector must be a scalar or have length n.'); end
    if isscalar(epsilon), epsilon = ones(1, n) * epsilon; elseif
length(epsilon) ~= n, error('Epsilon vector must be a scalar or have length
n.'); end
    C = Isp * g0;
    Vn_max_possible = sum(C .* log(1 ./ epsilon));
    if Vn_req >= Vn_max_possible
        error('Required Delta-V (%.f m/s) is >= max possible (%.f m/s).',
Vn_req, Vn_max_possible);
    end
    alpha_equation = @(a) sum(C .* log((C - a) ./ (epsilon .* C))) - Vn_req;
    upper_bound = min(C .* (1 - epsilon));
    search_interval = [0, upper_bound - 1e-6];
    options = optimset('Display','off');
    alpha = fzero(alpha_equation, search_interval, options);
    lambda = (alpha .* epsilon) ./ (C - C .* epsilon - alpha);
    R = (1 + lambda) ./ (epsilon + lambda);
    Gamma = prod(lambda ./ (1 + lambda));
end
```

```
Calculating optimal payload fraction for different numbers of stages...
Vehicle Technology: Isp = 360 s, Epsilon = 0.10
Mission Requirement: Delta-V = 9077 m/s

Testing n = 1 stage(s): Mission is impossible. Required Delta-V (9077 m/s) is
>= max possible (8129 m/s).
Testing n = 2 stage(s): Optimal Payload Fraction = 0.0385
Testing n = 3 stage(s): Optimal Payload Fraction = 0.0468
Testing n = 4 stage(s): Optimal Payload Fraction = 0.0501
Testing n = 5 stage(s): Optimal Payload Fraction = 0.0519
Testing n = 6 stage(s): Optimal Payload Fraction = 0.0529

Plot generated successfully.
```



Published with MATLAB® R2024b