Super Pressure Balloon Performance Estimation

14 January 2024

SUMMARY

This documents derives the equations necessary and provides the Matlab code to compute basic performance metrics for small super pressure balloons. Based on the knowledge of the gas used, the mass of the balloon material and payload, the fully inflated volume of the balloon, and the chosen free lift weight, an estimate of the balloon's final float altitude and internal vs ambient pressure difference is computed.

CONTEXT

A super pressure balloon should not be confused with a high altitude balloon. A hight altitude balloon is characterized by it's constantly expanding volume as it rises - maintaining an internal vs. ambient pressure equilibrium until the balloon bursts, typically a few hours after launch. A super pressure balloon by comparison is made with a non-stretch material like foil or mylar that once the gas expands to fill its volume, must be able to withstand a pressure difference at the balloon's float altitude. By carefully choosing the balloon's volume, payload mass, and free lift weight, it is possible to settle at a float altitude where the internal pressure of the balloon doesn't exceed its hoop stress limit.

LIMITATIONS

This document only provides an estimate of the performance metrics (float altitude and internal vs. ambient balloon pressure difference) based on the standard atmosphere. Real life conditions will likely vary significantly from the standard STP reference values used in these calculations, but it should provide a good starting point. Additional refinement could be empirically developed by measuring pressure and temperature while in flight.

CONSTANTS

R_{He}	Gas Constant (2077 for Helium)	$J \cdot Kg^{-1}K^{-1}$

KNOWN VARIABLES

$V_{balloon}$	Fully Inflated Balloon Volume	m^3
m_f	Free Lift Weight	Kg
m_p	Payload Mass	Kg
m_b	Balloon Material Mass	Kg

UNKNOWN VARIABLES

m_{He}	Mass of Needed Gas (Helium) to Satisfy m_f	Kg
V_{He}	Volume of m_{He} of Gas (Helium)	m^3
h_i	Fully Inflated Balloon Altitude	m
h_f	Final Balloon Float Altitude	m
D_{air}	Density of the Air that Corresponds to the Balloon's Altitude. Reverse Lookup Altitude Based on STP Table.	$Kg \cdot m^{-3}$
P_f	Balloon's Internal Pressure at Float Altitude	Pa
ΔP_f	Pressure Difference Between Balloon's Internal Pressure and Ambient Pressure	Pa

APPROACH

In order to understand how we compute the final float altitude, we need to understand the 3 principal phases a super pressure balloon goes through after launch.

- PHASE 1: RISE STATE - During this phase the balloon is nowhere near fully inflated. The volume of Helium in the balloon is substantially less than what the balloon can handle. The internal pressure is the same as the ambient pressure. Because we have chosen a free lift weight (m_f), the buoyant force (F_b) is greater than the force of gravity (F_g) and the balloon will rise.

- PHASE 2: FULLY INFLATED STATE - During this phase the balloon has reached its fully inflated state. The volume of Helium in the balloon is equivalent to what the balloon can handle. The internal pressure is the same as the ambient pressure. However, the balloon will still continue to rise because F_b still exceeds F_g. As the balloon continues to rise, F_b decreases while the pressure of the balloon ($P_{balloon}$) increases.

- PHASE 3: FLOAT ALTITUDE STATE - Eventually F_b and F_g cancel each other out, and the balloon reaches a steady state altitude that corresponds to a specific atmospheric density. From this atmospheric density, the altitude can be computed from a standard atmosphere table.

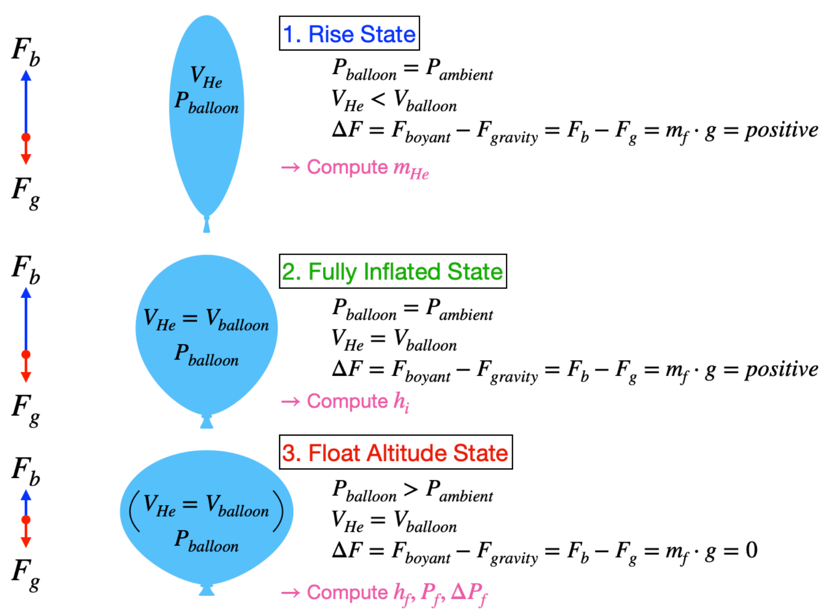

Figure 1. Phases of a Super Pressure Balloon

STEP 0. Pick Free lift weight m_f in Kg.

STEP 1. Compute Helium required to meet m_f (in terms of both Mass (m_{He}) and Volume (V_{He}).

Establish Equations,

$$\Delta F = F_{boyant} - F_{gravity} = F_b - F_g$$

$$\Delta F = m_f \cdot g$$

$$F_b = V_{He} \cdot D_{air} \cdot g$$

$$F_g = (m_b + m_p + m_{He}) \cdot g$$

Note, $V = (m \cdot R \cdot T)/P$

Compute answers,

$$\Delta F = m_f \cdot g = F_b - F_g$$

$$m_f \cdot g = \left[V_{He} \cdot D_{air} \cdot g\right] - \left[m_b + m_p + m_{He}\right] g$$

$$V_{He} \rightarrow (m_{He} \cdot R \cdot T)/P$$

$$m_f = \left[\frac{m_{He} \cdot R_{He} \cdot T \cdot D_{air}}{P}\right] - m_b - m_p - m_{He}$$

$$m_f + m_b + m_p = \left[\frac{m_{He} \cdot R_{He} \cdot T \cdot D_{air}}{P}\right] - m_{He}$$

$$m_f + m_b + m_p = m_{He}\left[\frac{R_{He} \cdot T \cdot D_{air}}{P} - 1\right]$$

$$\boxed{m_{He} = \frac{m_f + m_b + m_p}{\left[\frac{R_{He} \cdot T \cdot D_{air}}{P} - 1\right]} \qquad V_{He} = \frac{m_{He} \cdot R_{He} + T}{P}}$$

STEP 2. Compute Fully Inflated Altitude

Establish Equations,

$$\Delta F = F_{boyant} - F_{gravity} = F_b - F_g = m_f \cdot g$$

$$F_b = V_{He} \cdot D_{air} \cdot g$$

$$F_g = \left[m_b + m_p + m_{He}\right] \cdot g$$

Compute answers,

$$m_f \cdot g = \left[V_{He} \cdot D_{air} \cdot g\right] - \left[m_b + m_p + m_{He}\right] g$$

$$V_{He} \rightarrow V_{balloon}$$

$$m_f = \left[V_{balloon} \cdot D_{air}\right] - \left[m_b + m_p + m_{He}\right]$$

$$m_f + m_b + m_p + m_{He} = V_{balloon} \cdot D_{air}$$

$$D_{air} = \frac{m_f + m_b + m_p + m_{He}}{V_{balloon}}$$

$$P_{balloon} = P_{ambient} \,|_{D_{air}}$$

$$h_i \leftarrow STP \,|_{D_{air}}$$

STEP 3. Compute Float Altitude

Establish Equations,

$$\Delta F = F_{boyant} - F_{gravity} = F_b - F_g = 0$$

$$F_b = V_{He} \cdot D_{air} \cdot g$$

$$F_g = \left[m_b + m_p + m_{He} \right] \cdot g$$

Compute answers,

$$F_b = F_g$$

$$\left[V_{He} \cdot D_{air} \cdot g \right] = \left[m_b + m_p + m_{He} \right] g$$

$$V_{He} \rightarrow V_{balloon}$$

$$V_{balloon} \cdot D_{air} = m_b + m_p + m_{He}$$

$$D_{air} = \frac{m_b + m_p + m_{He}}{V_{balloon}}$$

$$P_f = \frac{m_{He} \cdot R_{He} \cdot T_{ambient} \mid_{D_{air}}}{V_{balloon}}$$

$$\Delta P_f = P_f - P \mid_{D_{air}}$$

$$h_f \leftarrow STP \mid_{D_{air}}$$

MATLAB CODE

Two Matlab functions and one script is provided to help estimate the performance of a super pressure balloon.

- isa_search.m - a function that does a reverse lookup of Matlab's Aerospace Toolbox standard atmosphere table to find the altitude that corresponds to a specific density. If the Aerospace Toolbox is not available, manual lookup from an STP table is necessary. This function is transparent to the user and is called by b_calc.m

- b_calc.m - a function that computes key balloon performance metrics. This function calls isa_search.m.

- b_calc_demo.m - An example script that shows how to use the b_calc function. Calling b_calc over a span of several payload sizes ranging from 5 grams to 100 grams, the figure below is computed.

```
b_calc(2077, 1.225, 101325, 288, 0.0024, 0.037, mass, 0.1);
```

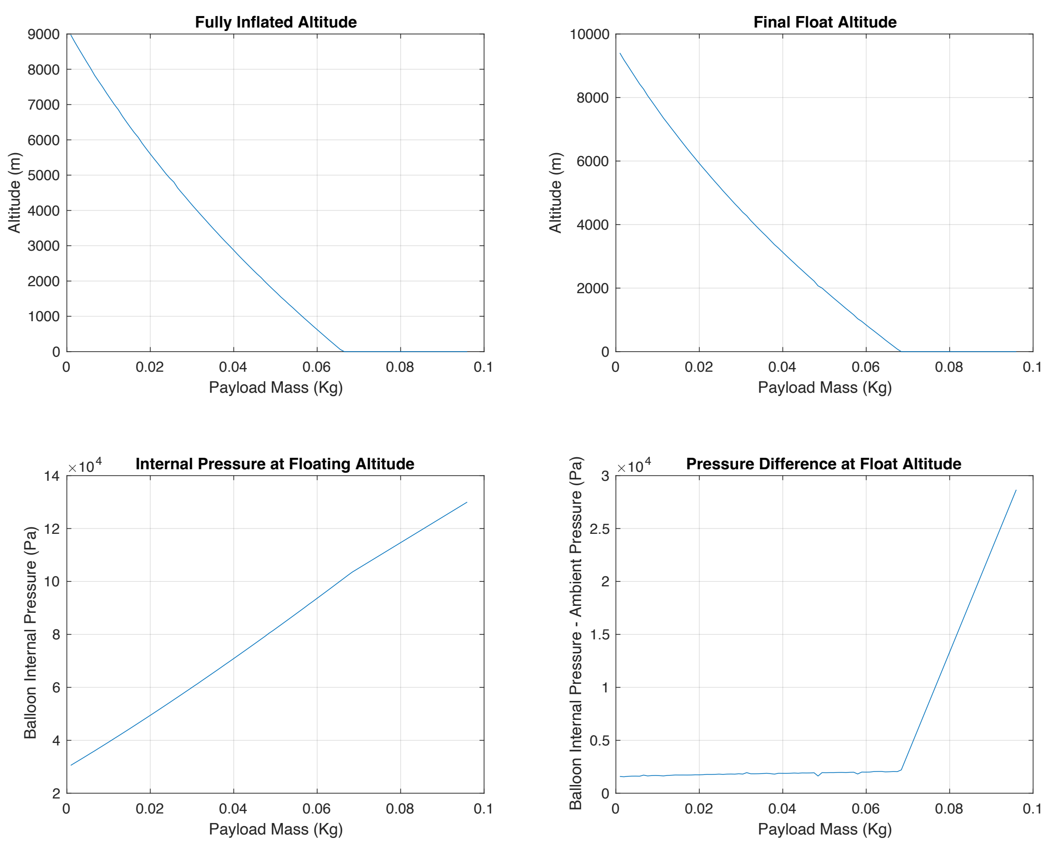

Figure 2. Example of 0.1 m^3 Helium Foil Balloon with Varying Payload Mass

Matlab Code 1 - b_calc_demo.m

```
%   -------------------------------------------------------------------------------------------------------------------
%   SCRIPT DESCRIPTION: Example script to show how to use b_calc and isa_search to compute
%                       key balloon performance metrics.
%
%   ---- b_calc() FUNCTION CALL SYNTAX ----
%   [h_i, h_f, p_f, delta_p_f] = b_calc(R_gas, D_launch_site, P_launch_site, T_launch_site, m_f, m_b, m_p, v_balloon)
%
%   ---- USER INPUTS ----
%   R_gas                         % Gas Constant in J/(Kg*K) (2077 for Helium, 4124 for Hydrogen)
%   D_launch_site                 % Atmospheric Density at launch site in Kg/m^3 (1.225 for SSL)
%   P_launch_site                 % Atmospheric Pressure at launch site in Pa (101325 for SSL)
%   T_launch_site                 % Atmospheric Temperature at launch site in Kelvin (288 for SSL)
%   m_f                           % Desired Free lift weight (Kg) (0.0024 Kg for example)
%   m_b                           % Mass of balloon material (Kg) (0.037 Kg for example)
%   m_p                           % Mass of Payload (Kg) (0.015 Kg for example)
%   v_balloon                     % Fully inflated balloon volume (m^3) (0.1m^3 or 0.25m^3 for example)
%
%   ---- OUTPUTS ----
%   h_i                           % Fully inflated balloon altitude (meters)
%   h_f                           % Final balloon float altitude (meters)
%   p_f                           % Balloon's internal Pressure at floating altitude (Pa)
%   delta_p_f                     % Pressure difference between balloon's internal pressure and ambient pressure (Pa)
%
%   -------------------------------------------------------------------------------------------------------------------

clear
clc

% ---- Plot Range & Fidelity ----
steps = 101;
min_mass = 0.005;
max_mass = 0.1;
mass = zeros(1,steps);

% ---- initialize return values ----
h_i = zeros(1,steps);
h_f = zeros(1,steps);
p_f = zeros(1,steps);
delta_p_f = zeros(1,steps);

% ---- Compute Balloon Volume ----
for n = 1:steps
   mass(1,n) = n*((max_mass - min_mass)/(steps-1));

   % --- Enter your modified values into this function call ---
   [h_i(1,n), h_f(1,n), p_f(1,n), delta_p_f(1,n)] = b_calc(2077, 1.225, 101325, 288, 0.0024, 0.037, mass(1,n), 0.1);
end
```

```
% ---- Plot Data ----
subplot(2,2,1)
plot(mass, h_i)
title('Fully Inflated Altitude')
xlabel('Payload Mass (Kg)')
ylabel('Altitude (m)')
grid on;

subplot(2,2,2)
plot(mass, h_f)
title('Final Float Altitude')
xlabel('Payload Mass (Kg)')
ylabel('Altitude (m)')
grid on;

subplot(2,2,3)
plot(mass, p_f)
title('Internal Pressure at Floating Altitude')
xlabel('Payload Mass (Kg)')
ylabel('Balloon Internal Pressure (Pa)')
grid on;

subplot(2,2,4)
plot(mass, delta_p_f)
title('Pressure Difference at Float Altitude')
xlabel('Payload Mass (Kg)')
ylabel('Balloon Internal Pressure - Ambient Pressure (Pa)')
grid on;
```

Matlab Code 2 - b_calc.m

```
function [h_i, h_f, p_f, delta_p_f] = b_calc(R_gas, D_launch_site, P_launch_site, T_launch_site, m_f, m_b, m_p, v_balloon)

%   ----------------------------------------------------------------------------------------------------------------------
%   FUNCTION DESCRIPTION: Compute key balloon performance metrics
%
%   ---- EXAMPLE ----
%   [h_i, h_f, p_f, delta_p_f] = b_calc(2077, 1.225, 101325, 288, 0.0024, 0.037, 0.015, 0.1);
%
%   ---- USER INPUTS ----
%   R_gas                         % Gas Constant in J/(Kg*K) (2077 for Helium, 4124 for Hydrogen)
%   D_launch_site                 % Atmospheric Density at launch site in Kg/m^3 (1.225 for SSL)
%   P_launch_site                 % Atmospheric Pressure at launch site in Pa (101325 for SSL)
%   T_launch_site                 % Atmospheric Temperature at launch site in Kelvin (288 for SSL)
%   m_f                           % Desired Free lift weight (Kg) (0.0024 Kg in example)
%   m_b                           % Mass of balloon material (Kg) (0.037 Kg in example)
%   m_p                           % Mass of Payload (Kg) (0.015 Kg in example)
%   v_balloon                     % Fully inflated balloon volume (m^3) (0.1 m^3 in example)
%
%   ---- OUTPUTS ----
%   h_i                           % Fully inflated balloon altitude (meters)
%   h_f                           % Final balloon float altitude (meters)
%   p_f                           % Balloon's internal Pressure at floating altitude (Pa)
%   delta_p_f                     % Pressure difference between balloon's internal pressure and ambient pressure (Pa)
%
%   ----------------------------------------------------------------------------------------------------------------------

% ---- STEP 1: Compute Helium required to meet Free Lift Weight Design Criteria ----
m_gas = (m_f+m_b+m_p)/((R_gas*T_launch_site*D_launch_site/P_launch_site)-1);

% ---- STEP 2: Compute Fully Inflated Altitude ----
D_lookup = (m_f+m_b+m_p+m_gas)/v_balloon;
h_i = isa_search(D_lookup);

% ---- STEP 3: Compute Float Altitude ----
D_lookup = (m_b+m_p+m_gas)/v_balloon;
h_f = isa_search(D_lookup);
[T_ambient, ~, P_ambient, ~] = atmosisa(h_f);
p_f = (m_gas*R_gas*T_ambient)/v_balloon;
delta_p_f = p_f - P_ambient;

end
```

Matlab Code 3 - isa_search.m

```
function [alt] = isa_search(rho_target)

%   ------------------------------------------------------------------------------------------------------------
%   FUNCTION DESCRIPTION: Reverse lookup of altitude from density in the Standard Atmosphere Table
%                         Matlab's ISA Table is only good up to 20,000 meters.
%
%   ---- EXAMPLE ----
%   [alt] = isa_search(1.0);
%
%   ---- USER INPUTS ----
%   rho_target                    % Fully inflated balloon volume (Kg/m^3)
%
%   ---- OUTPUTS ----
%   alt                           % Altitude that corresponds to an atmospheric density of rho_target (meters)
%
%   ------------------------------------------------------------------------------------------------------------

alt = 0;    % initialize return variable alt

% Check if Aerospace Toolbox Installed - which includes the atmosisa() function.
if (license('test','aerospace_toolbox') == 0)
    disp('You do not have the Aerospace Toolbox Installed');
    disp('You can still use the Standard Atmosphere Table by Hand');
    disp(['Look up the row that corresponds to a density of: ', num2str(rho_target), ' Kg/m^3']);
    disp('to find the corresponding altitude, ambient pressure and temperature.');
    return;
end

% Special Case - Alt <= 0 meters - Assume we are SSL or above
if (rho_target > 1.225)
    rho_target = 1.225;
    alt = 0;
    return;
end

% Special Case - Alt >= 20,000 meters
if (rho_target <= 0.0880) % Special case
    disp('ERROR: Atmospheric Density < 0.0880 Kg/m^3');
    disp('Matlab atmosisa() function does not support lookup above 20,000 meters');
    disp('Matlab R2023a allows for extended table up to 84,852 meters using atmosisa(rho_target,extended=true)');
    disp('You can still use the Standard Atmosphere Table by Hand');
    disp(['Look up the row that corresponds to a density of: ', num2str(rho_target), ' Kg/m^3']);
    disp('to find the corresponding altitude, ambient pressure and temperature.');
    alt = 20e3;
    return;
end
```

```
% Initialize Search Variables
alt = 10e3;
delta_alt = 10e3;
[~,~,~,rho] = atmosisa(alt);

while ((abs(rho-rho_target) > 0.0001))

    % Update rho based on alt guess
    [~,~,~,rho] = atmosisa(alt);

    % if not close enough guess new altitude
    delta_alt = (delta_alt/2);
    if (rho <= rho_target) % go down in altitude
        alt = alt - delta_alt;
    elseif (rho > rho_target) % go up in altitude
        alt = alt + delta_alt;
    end

end

end
```