A Computational Approach to Helium Balloon Assisted High Altitude Falls

Reaal Khalil
St. Hilda's College

May 21, 2015

1 Abstract

The aim of this project is to analyse the fall of a person from very high altitudes with the aid of a helium filled weather balloon to slow their descent. Descents both with and without a connected helium tank will be investigated. The results showed that the ground impact speeds in both cases are significantly less than the value had there been no balloon at all: 270kph. However the ground impact speed in the case with both a tank and a balloon (20kph) is significantly lower than that with only a balloon (140kph) due to the variation of atmospheric pressure with altitude.

2 Introduction

We will focus on finding the ground impact speeds in the two aforementioned configurations. The reason for studying both configurations is that carrying a pressurised tank would add a significant amount of weight which will partially counteract the forces of buoyancy and drag. Whereas the lack of a connected pressurised Helium tank will allow the balloon to shrink as atmospheric pressure increases during the descent. So our goal is to find which of these configurations is more efficient in slowing the fall. Before we attempt to find the ground impact speeds in the two cases, we must write a secondary code to find an experimental constant which will come up in Equation (7).

This paper will refer to an existing commercial weather balloon (Novalynx 400-8242[1]) and a high pressure gas cylinder (Hydra SP63[2]). The balloon has an uninflated radius of 0.54 metres and a burst radius of 3.4 metres, so we can safely keep it inflated at 3 metres. The gas cylinder can hold 50 litres ($0.05m^3$) of gas at a pressure of $2 \times 10^7 Pa$ and it has a mass of 50kg.

Firstly this paper shall list and explain all the physical laws and relations that will be used, then it will show how these mathematical expressions are used in the Matlab code. And finally, we will look at the results of running the code and analyse them.

[1]`http://novalynx.com/products/upper-air/400-82xx-weather-balloons/`

[2]`http://www.hydra.co.za/products/diving-solutions/cylinders-and-gas-banks.html`

3 Theory

3.1 Properties of the Earth's Atmosphere

At ground level, P_{atm} has an average value of 101 KPa and it drops exponentially as altitude increases, therefore in this problem the variation in atmospheric pressure (and density) can not be ignored. Temperature also varies with altitude; however, it decreases in the troposphere then it starts to increase again in the stratosphere. Thus its behaviour is difficult to model therefore instead of using theoretical models for pressure, air density and temperature, I will refer to a data table [1] of these measured values.

3.2 Properties of the Balloon

Balloons made from types of rubber that have a stress–strain relation of the Mooney–Rivlin[2] type are described by equation (1). Although there is experimental evidence [3] of hysteresis in rubber balloons when they are repeatedly inflated and deflated, we shall ignore this phenomenon since in this problem the balloon will only be inflated once.

$$\Delta P = 2\mu \frac{t_0}{r_0} \left(\left(\frac{r_0}{r} \right) - \left(\frac{r_0}{r} \right)^7 \right) \left(1 + \frac{1-\alpha}{\alpha} \left(\frac{r}{r_0} \right)^2 \right) \tag{1}$$

Where ΔP is the difference between the pressure inside the balloon and the atmospheric pressure, r is the radius of the balloon, r_0 and t_0 are the radius and the thickness of the balloon, respectively, before it is inflated. The parameter μ is called the shear modulus and has a typical value of $300kPa$ for rubber while the parameter α usually takes the value 10/11 [4].

Using the ideal gas law for the contents of the balloon (where P_{atm} and T_{atm} are respectively the atmospheric pressure and temperature at the balloon's altitude, R is the ideal gas constant, n is the number of moles of Helium in the balloon) we have:

$$\Delta P \;=\; P_{balloon} - P_{atm} \;=\; \frac{nRT_{atm}}{V_{balloon}} - P_{atm} \;=\; \frac{nRT_{atm}}{\frac{4}{3}\pi r^3} - P_{atm} \tag{2}$$

Which will give us a relation between r, P_{atm}, T_{atm} and n. Taking $P_0 = \frac{2\mu t_0}{r_0}$ and $k = \frac{1-\alpha}{\alpha}$ we have:

$$\frac{P_0 k}{r_0} r^8 + P_{atm} r^7 + P_0 r_0 r^6 - \frac{3nRT_{atm}}{4\pi} r^4 - P_0 k {r_0}^5 r^2 - P_0 {r_0}^7 = 0 \tag{3}$$

Using Matlab's built–in function `roots` this is easily solvable and it always results in one positive real value for r.

3.3 Forces on the Balloon

Gravitational force: We will assume the approximation $F_{grav} = mg$ for gravity holds at the altitudes of this problem, as this model fails by only 5% at an altitude of 100km.

Drag force: We will also assume the person in this problem is point-like, as the drag force on a weather balloon is much greater than that on a human body. Where C_D is the drag coefficient (around 0.5 for a sphere), A is the cross section area of the balloon, ρ_{atm} is the density of the atmosphere at the altitude of the balloon, and v is the speed of the descent, this force is given by:

$$F_D = \frac{1}{2} C_D A \rho_{atm} v^2 = \frac{1}{2} C_D \left(\pi r^2\right) \rho_{atm} v^2 \tag{4}$$

Buoyancy force: The main reason we use Helium is because it's lighter than air and thus experiences a decelerating buoyancy force given by:

$$F_B = g \rho_{atm} V_{balloon} = g \rho_{atm} \frac{4\pi}{3} r^3 \tag{5}$$

3.4 Inflation of the Balloon

The flow of gases slower than about 0.3 Mach speed can be approximated as incompressible flow, so we are justified in applying Bernoulli's equation:

$$\frac{1}{2} \rho v_t^2 + \rho g y_t + P_t = \frac{1}{2} \rho v_b^2 + \rho g y_b + P_b \tag{6}$$

Where the subscripts indicate on which side of the gas cylinder's valve the values are taken, t refers to the tank's side of the valve and b to the balloon's side. The difference in gravitational energy ($\rho g y_t - \rho g y_b$) is zero, and we'll take the speed of gas flow in the tank to be negligible. Thus using the fact that the rate of Helium atoms entering the balloon per unit time is proportional to the velocity of the gas ($\frac{dn}{dt} \propto v$) we get:

$$\frac{dn}{dt} = C \sqrt{P_{tank} - P_{balloon}} \tag{7}$$

We will find the constant of proportionality in the next section.

4 Code and Results

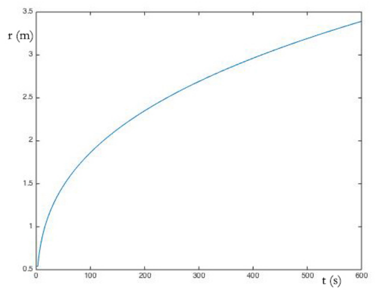
All of the time differential equations we've come across can be solved computationally using the Euler method depicted by equation (8).

$$x_{n+1} = x_n + \Delta t \frac{dx}{dt} \tag{8}$$

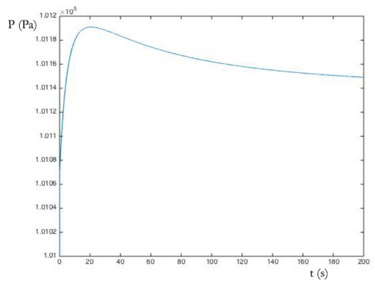
This involves a fair bit of errors as time increases, but this is more of a problem with rapidly changing solutions. In this problem, solutions will look more like exponential decays than oscillating functions so it's safe to use this method, decreasing step sizes (Δt) also greatly increases accuracy.

4.1 Finding the constant C

The constant in (7) depends on the inner size and shape of the valve, it is far easier and more accurate to use a measured value than to model gas flow through a constriction. To do this we need to employ a secondary program, this program will simulate a balloon inflating at ground level. The Matlab code for this is included in Appendix A. A fair assumption is that the balloon takes 10 minutes to fully inflate at ground level, using this information, we can vary the constant and run the code iteratively until the time taken to inflate the balloon fully is 10 minutes. The code works by first calculating $\frac{dn}{dt}$ (`dn(t)`) then using that to find the number of moles in the tank and in the balloon (`ntank`) and (`nballoon`) respectively. Next it finds the radius of the balloon (`rballoon`) and from that the volume of the balloon (`Vballoon`) and the new pressure in both the balloon (`Pballoon`) and in the tank (`Ptank`). Using these new values of pressure, we find the new value of $\frac{dn}{dt}$ and so on... The results we get are illustrated by Figure 1.



(a) Plot of balloon radius with time



(b) Plot of pressure inside the balloon with time

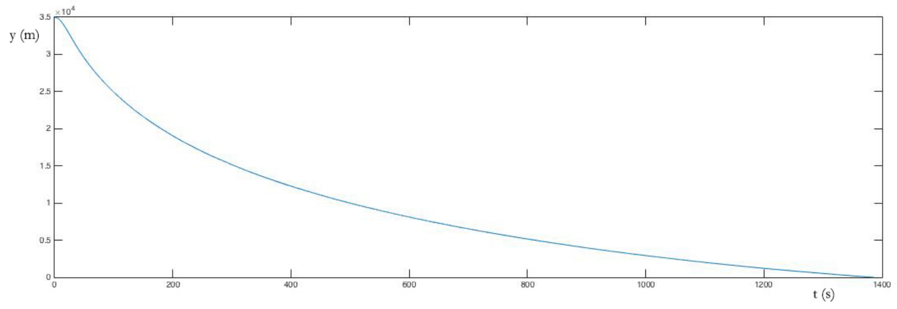
Figure 1: Inflating a balloon at ground level

After some tuning, the constant turns out to be: $C = 4 \times 10^{-8} m^{\frac{1}{2}} kg^{-\frac{1}{2}}$

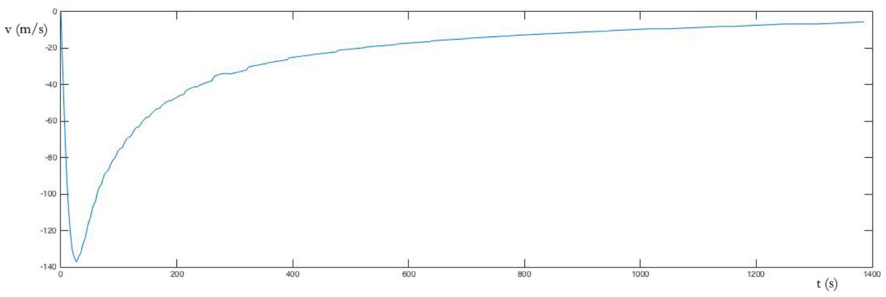
4.2 Falling with a balloon and a Helium tank

The full MATLAB code for a person falling with a weather balloon connected to a Helium tank is included in Appendix B. This code works similarly to the previous one but since the balloon is at high altitudes, the risk of bursting is greater, so we employ a ”smart valve” on the tank, which turns on only when the radius is less than 3 metres. This allows the balloon to maintain a maximum radius of 3 metres throughout the descent. Also, atmospheric pressure and temperature are fetched from a table of values included in Appendix B. The sum of forces on the balloon is used to find the acceleration, from that the velocity and altitude are found. Note that the mass of the tank is added to the mass of the person and balloon, this is in fact a significant addition.

The highest altitude a weather balloon can remain intact is about 45km so a choice of 35km would be a reasonable height to start from, since any higher and there wouldn't be enough pressure to have any effect. The results of running it with an initial altitude of 35km are shown in Figure 2.



(a) Plot of altitude against time



(b) Plot of speed against time

Figure 2: Falling with a balloon and a Helium tank

The main features of these plots are that the fall lasts 23 minutes before impact with the ground at a speed of 5.5m/s (19.8km/h) which is equivalent to an unprotected fall from 1.54 metres off the ground, which is very easily survivable. To get a quantitative idea of how dangerous such an impact can be, I will refer to a study by R. Cuerden [5]. The study showed that pedestrians involved in crashes with vehicles travelling at similar speeds have a 75% chance of surviving with minor injuries, a 23% chance of sustaining severe injuries and a 2% chance of death.

4.3 Falling with only a balloon

This section uses the same previous code, save for the pressurised tank and so the number of moles in the balloon is constant. The Matlab code for this section is included in Appendix C, we will run it from the same initial altitude of 35km.

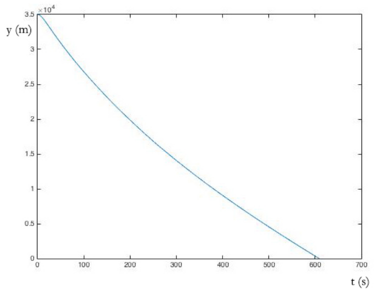


Figure 3: Plot of $y(t)$

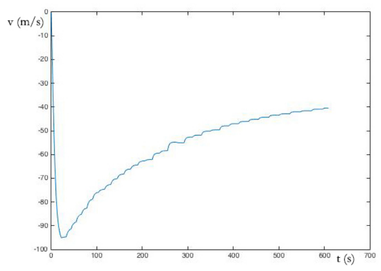


Figure 4: Plot of $\frac{dy}{dt}(t)$

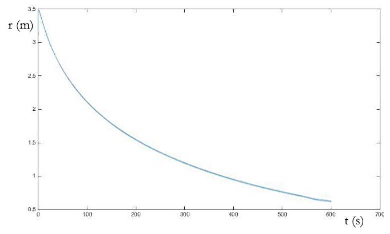


Figure 5: Plot of $r_{balloon}(t)$

Figure 6: Falling with only a balloon

This fall lasts 10 minutes before impact with the ground at a speed of 40m/s (144km/h). The reason the impact speed in this case is so much greater than that in the first case is because an inflated balloon at high altitudes (and thus high pressure) will deflate at lower altitudes. The decrease in radius decreases the forces of drag and buoyancy and results in a higher impact speed.

5 Conclusion

The results of this study showed that a Helium filled weather balloon can slow the descent of a skydiver to safe speeds. However, if the balloon is not connected to a pressurised tank, due to the increase in atmospheric pressure with decrease in altitude the radius of the balloon will drop significantly compromising the decelerating forces,

References

[1] Williams, J. *Standard Atmosphere Tables*, USA TODAY, 2005. [3]

[2] Atkins, J. E. and Rivlin, R. S. *Large Elastic Deformations of Isotropic Materials IX. The Deformation of Thin Shells* Davy Faraday Laboratory of the Royal Institution, 1951.

[3] Merritt, D. R. and Weinhaus, F. *The pressure curve for a rubber balloon*[4] University of Santa Clara, California, 1977.

[4] Müller, I. and Strehlow, P. *Rubber and rubber balloons: paradigms of thermodynamics*[5] Lecture Notes in Physics, Springer, 2004.

[5] Cuerden R. et al *Pedestrians and their Survivability at Different Impact Speeds*, Transport Research Laboratory [6]

[3] `http://usatoday30.usatoday.com/weather/wstdatmo.htm`

[4] `http://www.loreto.unican.es/Carpeta2007/00TorreonCartes2007/BalloonPressureC.pdf`

[5] `http://www.amazon.com/Rubber-Balloons-Paradigms-Thermodynamics-Lecture/dp/3540202447`

[6] `http://www-nrd.nhtsa.dot.gov/pdf/esv/esv20/07-0440-W.pdf`

Appendix A: Inflating a balloon at ground level

This code uses a function called `radiusmooneyrivlin` which is included in Appendix B

```
Vtank=0.4;
Pout = 101000;
R = 8.314;
T = 288;

Ptank(1) = 20000000;
ntank(1) = Ptank(1)*Vtank/R*T;
rballoon(1) = 0.54;
Vballoon(1) = 4*pi*rballoon(1).^3/3;
Pballoon(1) = Pout;
nballoon(1) = Vballoon(1)*Pballoon(1)/(R*T);

tmax=900
timediv=1/100;

for t=1:tmax/timediv;
    dn(t) = 40e-7*timediv*(Ptank(t)-Pballoon(t))^0.5;
    ntank(t+1) = ntank(t) - dn(t);
    nballoon(t+1) = nballoon(t) + dn(t);
    rballoon(t+1)=radiusmooneyrivlin(nballoon(t),Pout,T);
    Ptank(t+1)=ntank(t+1)*R*T/Vtank;
    Vballoon(t+1)=4*pi*rballoon(t+1)^3/3;
    Pballoon(t+1)=nballoon(t+1)*R*T/Vballoon(t+1);
end

figure
plot(timediv*(1:tmax/timediv),rballoon(1:tmax/timediv));
```

Appendix B: Falling with a balloon and a Helium tank

main.m

```
Vtank=0.05;
m=160;
g=9.8;
R = 8.314;

h(1)=35000;
T(1)=atmosphere(h(1),1);
Pout(1)=atmosphere(h(1),2);
rho(1)=atmosphere(h(1),3);

Ptank(1) = 20000000;
ntank(1) = Ptank(1)*Vtank/R*T(1);
rballoon(1) = 0.54;
Vballoon(1) = 4*pi*rballoon(1).^3/3;
Pballoon(1) = Pout(1);
nballoon(1) = Vballoon(1)*Pballoon(1)/(R*T(1));
v(1)=0;
a(1)=-g;

tmax=40000;
timediv=1/50;

for t=1:tmax/timediv;
    %inflate balloon
    if rballoon(t)<3 && Ptank(t)>=Pballoon(t)
        aa=t;
        dnbydt(t) = 40e-7*timediv*(Ptank(t)-Pballoon(t))^0.5;
    else
        dnbydt(t) = 0;
    end
    ntank(t+1) = ntank(t) - dnbydt(t);
    nballoon(t+1) = nballoon(t) + dnbydt(t);
    rballoon(t+1)=radius_mooneyrivlin(nballoon(t),Pout(t),T(t))
        ;
    Ptank(t+1)=ntank(t+1)*R*T(t)/Vtank;
    Vballoon(t+1)=4*pi*rballoon(t+1)^3/3;
    Pballoon(t+1)=nballoon(t+1)*R*T(t)/Vballoon(t+1);

    Fgrav = m*g;
    Fbuoyancy = rho(t)*Vballoon(t)*g;
    Fdrag = 0.5*rho(t)*v(t)^2*0.5*pi*rballoon(t).^2;
    a(t+1) = (Fbuoyancy+Fdrag-Fgrav)/m;
    v(t+1) = v(t)+a(t)*timediv;
    h(t+1) = h(t)+v(t)*timediv+.5*a(t)*timediv^2;

    T(t+1) = atmosphere(h(t+1),1);
    Pout(t+1) = atmosphere(h(t+1),2);
    rho(t+1)= atmosphere(h(t+1),3);
    if h(t+1)<=0
        tmax=t;
        break
    end
end
figure
plot(timediv*(1:tmax),v(1:tmax));
```

radiusmooneyrivlin.m

```
function r = radiusmooneyrivlin(nballoon,Pout,T)
    r0=0.54;
    t0=0.0002;
    mu = 300000;
    R = 8.314;
    p0 = 2*mu*t0/r0;
    k = 0.1;
    X = 3*nballoon*R/(4*pi);
    poly = [ (p0*k/r0) Pout (p0*r0) 0 (-X*T) 0 (-p0*k*r0^5) 0
       (-p0*r0^7) ];
    r = roots(poly);
    r = r((imag(r) == 0) & (r > 0));
end
```

atmosphere.m

```
function a = atmosphere(h,n)
    %n=1 temperature; n=2 pressure; n=3 density
    table = [288,101300,1.2; 281.5,90000,1.1; 275,80000,1;
       268.5,70000,0.91; 262,62000,0.82; 255.5,54000,0.74;
       249,47000,0.66; 242.5,41000,0.59; 236,36000,0.53;
       229.5,31000,0.47; 223,26000,0.41; 216.5,23000,0.36;
       216.5,19000,0.31; 216.5,17000,0.27; 216.5,14000,0.23;
       216.5,12000,0.19; 216.5,10000,0.17; 216.5,9000,0.14;
       216.5,7500,0.12; 216.5,6500,0.1; 216.5,5500,0.088;
       217.5,4700,0.075; 218.5,4000,0.064; 219.5,3400,0.054;
       220.5,2900,0.046; 221.5,2500,0.039; 222.5,2200,0.034;
       223.5,1800,0.029; 224.5,1600,0.025; 225.5,1400,0.021;
       226.5,1200,0.018; 227.5,1000,0.015; 228.5,870.0,0.013;
       231.3,750,0.011; 234.1,650,0.0096; 236.9,560,0.0082];
    if h <= 0
        h = 1;
    end
    A = table(round(h/1000)+1,:);
    a = A(n);
end
```

Appendix C: Falling with an inflated balloon

```
Vtank = 0.05;
m = 70;
g = 9.8;
R = 8.314;
h(1) = 35000;
T(1) = atmosphere(h(1),1);
Pout(1) = atmosphere(h(1),2);
rho(1) = atmosphere(h(1),3);

Ptank(1) = 20000000;
ntank(1) = Ptank(1)*Vtank/R*T(1);
rballoon(1) = 3.5;
Vballoon(1) = 4*pi*rballoon(1).^3/3;
Pballoon(1) = Pout(1);
nballoon = Vballoon(1)*Pballoon(1)/(R*T(1));
v(1) = 0;
a(1) = -g;
tmax = 1500;
timediv = 1/50;

for t=1:tmax/timediv;
    rballoon(t+1) = radius_mooneyrivlin(nballoon,Pout(t),T(t));
    Vballoon(t+1) = 4*pi*rballoon(t+1)^3/3;
    Pballoon(t+1) = nballoon*R*T(t)/Vballoon(t+1);
    Fgrav = m*g;
    rhoBalloon = Pballoon(t+1)*2/R/T(t)/1000;
    Fbuoyancy = rho(t)*Vballoon(t)*g;
    Fdrag = 0.5*rho(t)*v(t)^2*0.5*pi*rballoon(t).^2;
    a(t+1) = (Fbuoyancy+Fdrag-Fgrav)/m;
    v(t+1) = v(t)+a(t)*timediv;
    h(t+1) = h(t)+v(t)*timediv+.5*a(t)*timediv^2;
    T(t+1) = atmosphere(h(t+1),1);
    Pout(t+1) = atmosphere(h(t+1),2);
    rho(t+1) = atmosphere(h(t+1),3);
    if h(t+1) <= 0
        tmax = t;
        break
    end
end

figure
plot(timediv*(1:tmax),v(1:tmax));
```