Modeling a Parachute Jump

ARO (`@_art_of_electronics_`)

June 23, 2025

1 Introduction

This document presents the modeling of a parachute jump using Lagrangian mechanics and co-energy analysis. A Simulink block diagram is also provided for simulation purposes.

2 Parachute Jump Description

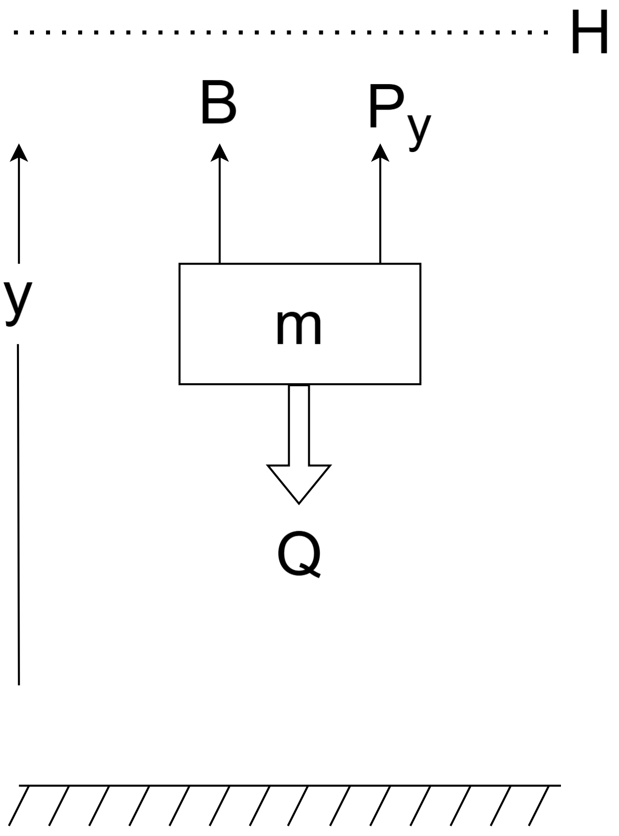

Figure 1: Overview of the parachute jump model

Presented model is a simplified version where 3 main forces are considered:

$$\begin{aligned} Q &- \text{weight;} \\ B &- \text{inertia;} \\ P_y &- \text{air resistance (drag);} \end{aligned}$$

3 Lagrangian Formulation

The kinetic and potential energies of the system are:

$$\begin{aligned} E_k &= \frac{1}{2} m\dot{y}^2 \\ E_p &= -m \int_0^y g(y)dy = -mg(y)y \end{aligned}$$

As can be seen, kinetic energy of the system is linked to inertia, and potential energy is linked to weight. Note, that potential energy is directed opposite to the direction of the y axis, so it must be written with negative sign. In this system has drag can be considered as a dissipative element:

$$D = \frac{1}{2} c \cdot A \int_0^{\dot{y}} \rho(y)\dot{y}^2 dy = \frac{1}{6} cA\rho(y)\dot{y}^3$$

Drag is linked to disspiation.

The Lagrangian of parachute jump is:

$$\mathcal{L}(x, \dot{x}) = E_k - E_p = \frac{1}{2} m\dot{y}^2 - (-mg(y)y) = \frac{1}{2} m\dot{y}^2 + mg(y)y$$

Using Lagrange's equation with a non-conservative force, the equation expands into the following form:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q} + \frac{\partial \mathcal{D}}{\partial \dot{q}} = 0$$

This leads to the following differential equation:

$$\frac{d}{dt}\left(\frac{\partial\left(\frac{1}{2}m\dot{y}^2 + mg(y)y\right)}{\dot{y}}\right) - \frac{\partial\left(\frac{1}{2}m\dot{y}^2 + mg(y)y\right)}{y} + \frac{\partial\left(\frac{1}{6}cA\rho(y)\dot{y}^3\right)}{\dot{y}} = 0$$

$$\frac{d}{dt}\left(\frac{\partial\left(\frac{1}{2}m\dot{y}^2 + 0\right)}{\dot{y}}\right) - \frac{\partial(0 + mg(y)y)}{y} + \frac{\partial\left(\frac{1}{6}cA\rho(y)\dot{y}^3\right)}{\dot{y}} = 0$$

$$\frac{d}{dt}\left(\frac{\partial\left(\frac{1}{2}m\dot{y}^2\right)}{\dot{y}}\right) - \frac{\partial(mg(y)y)}{y} + \frac{\partial\left(\frac{1}{6}cA\rho(y)\dot{y}^3\right)}{\dot{y}} = 0$$

$$\frac{d}{dt}\left(2 \cdot \frac{1}{2} m\dot{y}^{2-1}\right) - \left(mg(y)y^{1-1}\right) + \left(3 \cdot \frac{1}{6} cA\rho(y)\dot{y}^{3-1}\right) = 0$$

$$\frac{d}{dt}(m\dot{y}) - mg(y) + \frac{1}{2} cA\rho(y)\dot{y}^2 = 0$$

$$m\ddot{y} - mg(y) + \frac{1}{2}cA\rho(y)\dot{y}^2 = 0$$

Sorting yields the equation's final form:

$$m\ddot{y} + \frac{1}{2}cA\rho(y)\dot{y}^2 - mg(y) = 0$$

4 Co-energy and Energy Expressions

Inertia

The general inertia co-energy for mass is:

$$T = \int_0^v mv'\,dv' = \frac{1}{2}mv^2 = \frac{1}{2}m\dot{y}^2$$

Weight

The general potential co-energy stored in the free falling mass is:

$$U = m\int_y^0 g(y)\,dy' = -mg(y)y$$

Drag

The general drag co-energy can be written as:

$$D = k\int_0^t \rho(y)|v|v^2(\tau)\,d\tau = \frac{1}{3}k\rho(y)v^3 = \frac{1}{6}cA\rho(y)\dot{y}^3$$

Using the co-energy equation with a non-conservative force, we can write:

$$\frac{d}{dt}\left[\frac{\partial T}{\dot{y}}\right] - \frac{\partial T}{y} + \frac{\partial U}{y} + \frac{\partial D}{\dot{y}} = f_i$$

By substituting the coefficients into the equation, we obtain the following:

$$\frac{d}{dt}\left[\frac{\partial\left(\frac{1}{2}m\dot{y}^2\right)}{\dot{y}}\right] - \frac{\partial\left(\frac{1}{2}m\dot{y}^2\right)}{y} + \frac{\partial\left(-mg(y)y\right)}{y} + \frac{\partial\left(\frac{1}{6}cA\rho(y)\dot{y}^3\right)}{\dot{y}} = 0$$

$$\frac{d}{dt}\left[\frac{\partial\left(\frac{1}{2}m\dot{y}^2\right)}{\dot{y}}\right] - 0 + \frac{\partial\left(-mg(y)y\right)}{y} + \frac{\partial\left(\frac{1}{6}cA\rho(y)\dot{y}^3\right)}{\dot{y}} = 0$$

$$\frac{d}{dt}\left[m\dot{y}\right] - mg(y) + \frac{1}{2}cA\rho(y)\dot{y}^2 = 0$$

$$m\ddot{y} - mg(y) + \frac{1}{2}cA\rho(y)\dot{y}^2 = 0$$

Sorted and simplified the final equation is:

$$m\ddot{y} + \frac{1}{2}cA\rho(y)\dot{y}^2 - mg(y) = 0$$

5 Altitude-dependent Functions

In this model are two empirical functions that strongly depend on the change of altitude - $g(y)$ and $\rho(y)$.

5.1 Altitude-dependent Gravity

$$g(y) = g_0 - 3.086 \cdot (H - y) \cdot (10^{-5})$$

where:

$$\begin{aligned} &g_0 - \text{Reference level of gravity, } g_0 = 9.81\frac{m}{s^2} \\ &H - \text{Jump-out initial altitude } [m] \end{aligned}$$

5.2 Altitude-dependent Air Density

$$\begin{cases} \rho(y) = \rho_0 \cdot \frac{p(y)}{p_0} \\ p(y) = p_0 - 13 \cdot (H - y) \end{cases}$$

where:

$$\begin{aligned} &\rho_0 - \text{Reference level of air density, } \rho_0 = 1.2\frac{kg}{m^3} \\ &p_0 - \text{Reference level of atmospheric pressure, } p_0 = 1.01 \cdot 10^5\frac{N}{m^2} \\ &H - \text{Jump-out initial altitude } [m] \end{aligned}$$

Ultimately, the final form of equation is:

$$\rho_y = \rho_0 \cdot \frac{p_0 - 13 \cdot (H - y)}{p_0}$$

6 Simulink Model for Lagrangian and Co-energy Expression

In order to model the equation properly, we must translate the dynamic equation into a form understandable for Simulink:

$$m\ddot{y} + \frac{1}{2}cA\rho(y)\dot{y}^2 - mg(y) = 0$$

$$\ddot{y} = g(y) - \frac{1}{2m}cA\rho(y)\dot{y}^2$$

In the following, the Simulink model is presented.

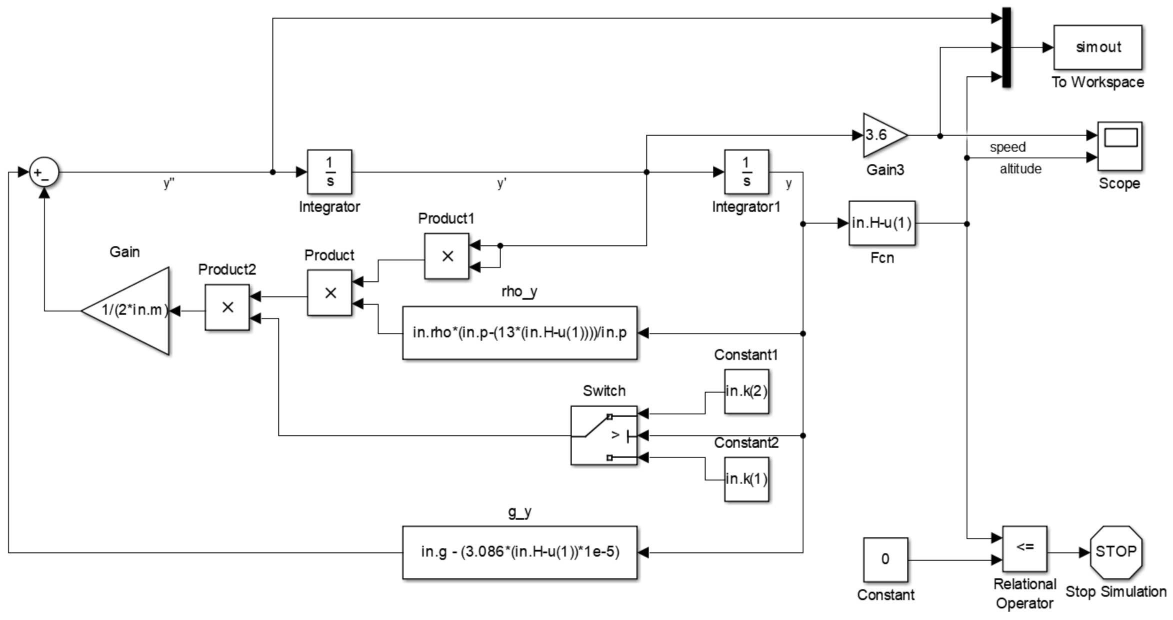

Figure 2: Simulink model of parachute jump

7 Matlab and Python script

For script-based system modeling, we must take a slightly different approach to translating mathematical equations into code. Let $y = y(1)$, $\dot{y}(t) = y(2)$, in that case the matrix of parachute jump equation will be written as:

$$\frac{d}{dt}\begin{bmatrix} y(1) \\ y(2) \end{bmatrix} = \begin{bmatrix} y(2) \\ g\_y - (c \cdot A \cdot rho\_y \cdot y(2) \cdot y(2))/(2 \cdot m) \end{bmatrix}$$

Now, the equation can be coded into Matlab:

```
g_y = g0 - 3.086 * (in.H - y(1)) * 1e-5;
rho_y = rho0 * ((p0 - 13 * (in.H - y(1))) / p0);
Dy = [y(2); g_y - c * A * (rho_y .* y(2) .* y(2) / (2 * in.m))];
```

Equivalent Python code can be written as:

```python
def parachutes(t, y):
    altitude = H - y[0]

    if altitude > h0:
        _c = c[0]
        _A = A[0]
    else:
        _c = c[1]
        _A = A[1]

    g_y = g0 - 3.086e-5 * altitude
    rho_y = rho0 * ((p0 - 13 * altitude) / p0)

    return y[1], g_y - (_c * _A * rho_y * y[1] ** 2) / (2 * m)
```

8 Simulation Results

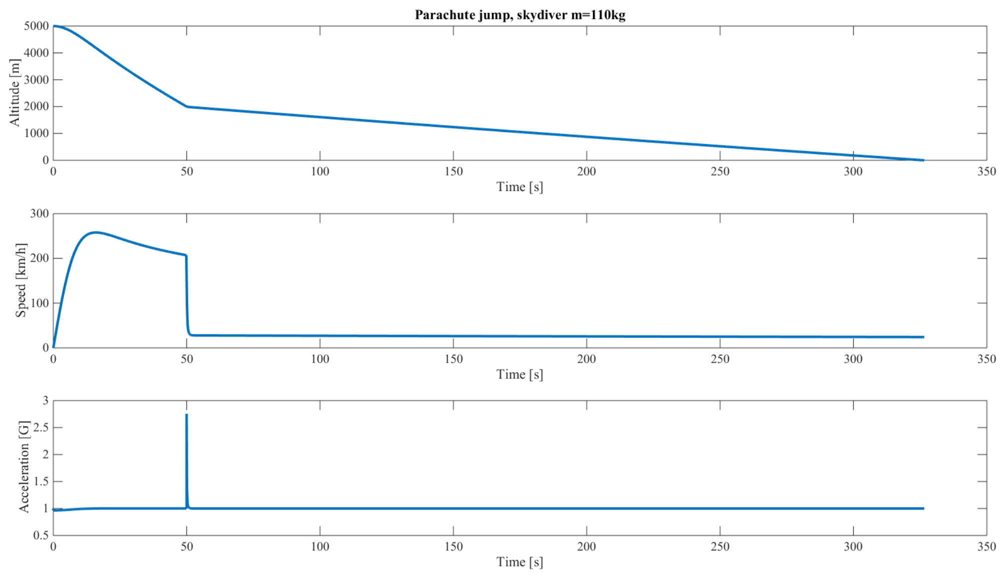

Figure 3: Result of simulation

9 Conclusion

This document demonstrates the Lagrangian and co-energy approach to modeling a parachute jump. It also provides a visual block diagram suitable for simulation in Simulink and basic code for scripting.