Basic Energy Calculations

March 6, 2018

1 Basic Energy Calculations

Jean-Christophe Perrin 2018 March 5 Monday

This notebook answers several fundamental questions about our jumper and energy required from it.

```
In [1]: import numpy as np
        import JumperUtilities
```

1.1 Jumper Properties

These are the basic properties of our robot from which we will calculate the energy required.

```
In [2]: mass = 0.1 # [kg] the mass of the robot

        height = 1 # [m] the height the robot is required to jump
        g = 9.8 # [m/s^2]
```

1.2 How much energy is associated with being one meter above ground?

This is easy to compute: mgh

```
In [3]: potentialEnergy = mass*g*height

        print 'Egrav = ', potentialEnergy, '[J]'

Egrav =  0.98 [J]
```

1.3 How much kinetic energy is needed at lift-off?

As we saw on Monday, we need somewhat higher kinetic energy at liftoff than mgh. There is air drag. Also, as pieces of the mechanism oscillate or rattle during flight, the associated kinetic energy is not usefully converted to upward motion. You can estimate air drag. The "rattling" effect is harder to estimate and might be where a simulation helps.

For a first pass, will estimate as a jump efficiency parameter.

```
In [4]: jumpEfficiency = 0.5 # [1] amount of stored energy
        kineticEnergy = potentialEnergy / jumpEfficiency

        print 'Ekinematic =', kineticEnergy, '[J]'
```

```
Ekinematic = 1.96 [J]
```

1.4 How much spring energy is needed to achieve the required kinetic energy at lift-off?

As we saw on Monday, only about $^1/_2$ of the stored potential energy in the springs, $0.5k(L_{max}^2 - L_{min}^2)$, got usefully converted to upward kinetic energy, $0.5mV_o^2$, at liftoff. The acceleration plot of the main body tells the story: it starts with a steady positive acceleration as the legs start to come together (good); but there is a large negative spike as the legs lock into straight position and the foot leaves the ground (not so good). This part involves complicated accelerations and collisions of various parts of the mechanism. It might be the part most useful to try to simulate in Working Model.

$$E = k\frac{L_{max}^2 - L_{min}^2}{2} \tag{1}$$

Rearranging we find:

$$\sqrt{\frac{2E}{k} + L_{min}^2} = L_{max} \tag{2}$$

```
In [5]: springEfficiency = 0.5 # [1] Kinetic_energy.
        k = 100 # []
        Lmin = 0   # [m] minimum extension of spring

        springEnergy = kineticEnergy / springEfficiency
        Lmax = np.sqrt(2*springEnergy/k + Lmin**2)

        print 'Espring =', springEnergy, '[J]'
        print 'Lmax = ', Lmax, '[m]'
```

```
Espring = 3.92 [J]
Lmax =  0.28 [m]
```

1.5 Will the gearmotor have enough torque to stretch or compress the springs to their maximum position?

$$F = kx \tag{3}$$

```
In [6]: bobbinDiam = 0.06 # [m]

        deltaX = Lmax - Lmin # [m]
        forceMotor = k * deltaX # [N]
        motorTorque = bobbinDiam * forceMotor # [Nm]

        print 'Torque =', motorTorque, '[Nm]'
```

```
Torque = 1.6800000000000002 [Nm]
```

1.6 How much energy will the motor consume in doing the compression and does the battery have plenty of energy to achieve it?

You can get enough information from motor data sheets, etc. to estimate this. It's likely you have far more than enough battery energy.

```
In [7]: motorEfficiency = 0.5
        gearEfficiency = 0.5
        energyIn = springEnergy/(motorEfficiency * gearEfficiency)

        print 'Estimated energy needed: ', energyIn, '[J]'
```

```
Estimated energy needed:  15.68 [J]
```