Project 2: How to Keep your Burrito Warm

First Draft

Alex Wenstrup and Robin Graham-Hayes

Question

Suppose you buy a chipotle burrito (or any other burrito) for takeout, and want it to be as warm as possible when you get home. If you lead a lonely life of luxury and have a vacant but heated passenger seat in your car, you may be inclined to turn on your heated seat and strap your burrito in for a toasty ride. If, however, you don't find yourself in that position, are you better off leaving your burrito in the initialy cooler, but insulated seat next to you, or on your lap which starts off warmer, but will be cooled by your bloodstream drawing the heat throughout your body?

In [7]:
```
# Configure Jupyter so figures appear in the notebook
%matplotlib inline

# Configure Jupyter to display the assigned value after an assignment
%config InteractiveShell.ast_node_interactivity='last_expr_or_assign'

# import functions from the modsim.py module
from modsim import *

# import image function to display images inline
from IPython.display import Image
```

Introduction

To answer this question, we will model two burritos simultaneously, one in somebody's lap and the other lying on a cushion, beginning when the burritos get to the car. We began by defining the initial temperatures of relevant surfaces, as well as all the model parameters in a System object.

In [8]:
```
#Define initial temperatures (in degrees C)

tBurrito=50
tWall=50

#from https://hypertextbook.com/facts/2001/AbantyFarzana.shtml
tLap=34
tCushion=22
```

Out[8]: 22

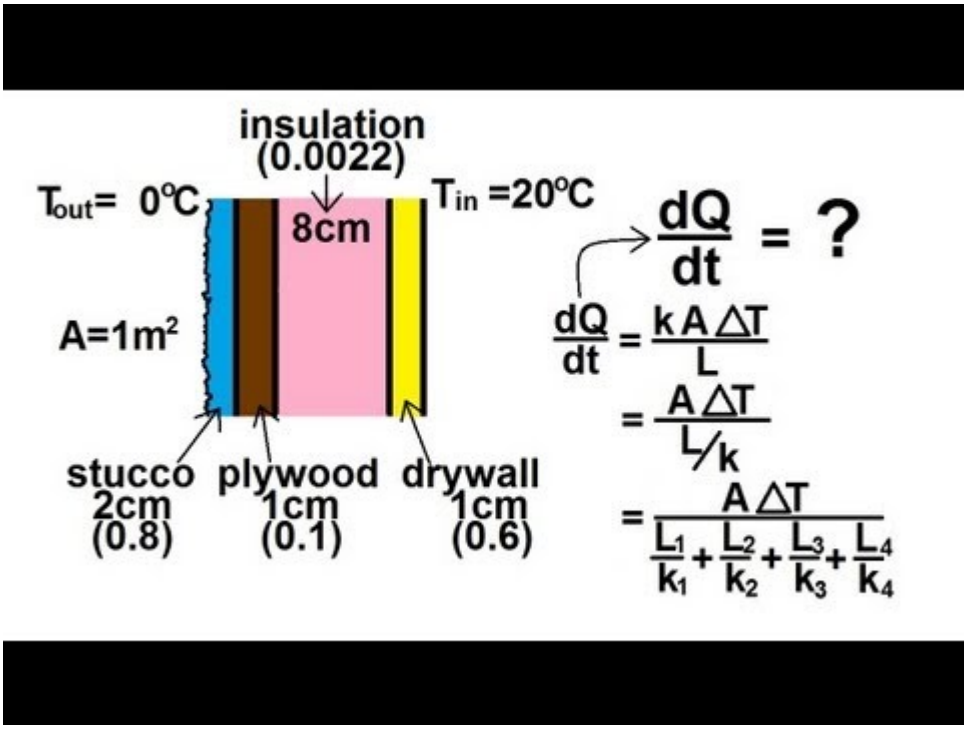
In [9]:
```
#Define State
state = State(tBurritoCush=tBurrito,
              tBurritoLap=tBurrito,
              tLap=tLap,
              tCushion=tCushion)

#Define system
system = System(init = state,
               k_burrito = 0.5, # W/(m * K)

               #k_cushion = 2x k_air
               k_cushion = 0.05,

               #k_lap = weighted average of k for human muscle, dermis and fat
               k_lap = .4,

               #Contact area of burrito and cooling surface, assumed constant
 between the lap and cushion
               A = 0.027,

               C_burrito = 3, #J/(g * C)
               C_cushion = 2,
               C_wall = 4,
               C_lap = 4)
```

Out[9]:

	values
init	tBurritoCush 50 tBurritoLap 50 tLap ...
k_burrito	0.5
k_cushion	0.05
k_lap	0.4
A	0.027
C_burrito	3
C_cushion	2
C_wall	4
C_lap	4

In [10]:
```
Image("multilayerconduction.jpg")
#This image demonstrates where our DE's came from
```

Out[10]:



Thermal Model

During the course of the project, we tried about half a dozen different stock and flow models and systems of diferential equations to model the heat trasfer from the burrito to its surroundings. Eventually, we decided to model heat as moving from the center of the burrito to the center of its surroundings, according the equation described above (where the multiple layers represtent the burrito and either the cushion or the lap).

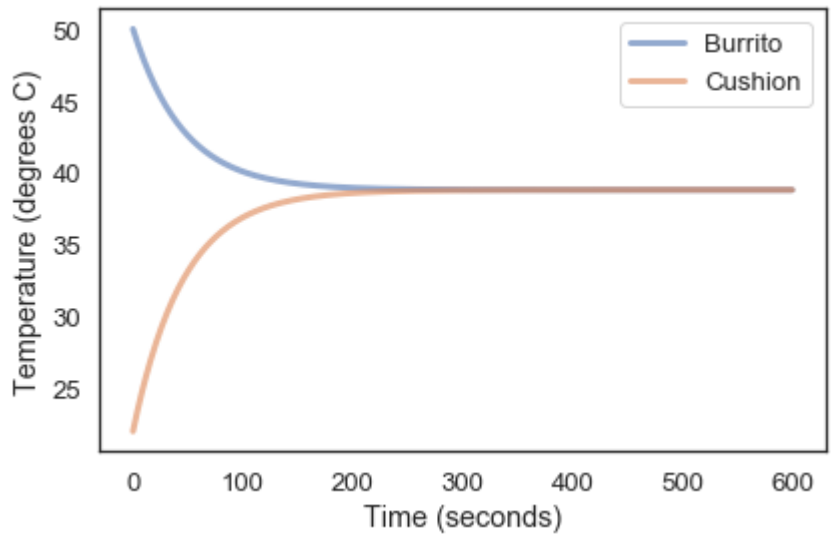
In [11]:
```
def update_Cush(state, t, system):

    """Updates states for the burrito on the cushion for a timestep of 1 secon
d

    state: state object of temperatures before update
    t: time
    system: all model parameters

    returns: new State object
    """

    new_state = state

    #Convert temperature to energy
    energy_B1 = state.tBurritoCush * system.C_burrito #Joules / gram
    energy_Cush = state.tCushion * system.C_cushion #Joules / gram

    #Calculate changes in energy
    energy_dB1 =  system.A * (state.tCushion - state.tBurritoCush ) / ((0.023
/ system.k_burrito ) + (0.05/system.k_cushion ))
    energy_dCushion = -1 * energy_dB1

    #Convert energy changes back to temperature
    dB1 = energy_dB1 / system.C_burrito #Degrees Celcius / gram
    dCushion = energy_dCushion / system.C_cushion #Degrees Celcius / gram

    #Update state
    new_state.tBurritoCush += dB1
    new_state.tCushion += dCushion

    return new_state
```

Note that the above function does not account for the masses of either the heat donor or reciever. It functions as if we treated the heat donor (burrito) and heat reviever (lap/cushion) as having equal masses, an assumption that a) doesn't drastically change how our model behaves and b) we felt comfortable making given the volumes we were dealing with.

In [12]:
```
def run_sim(system, t, update_func):

    """Runs simulation over t timesteps

    system: all model parameters
    t: time
    update_func: function called during the simulation

    returns: TimeFram object
    """

    frame = TimeFrame(columns=system.init.index)
    frame.row[0] = system.init

    for i in linrange(0, t):
        frame.row[i+1] = update_func(frame.row[i], i, system)

    return frame
```

In [13]:
```
def plot_2_results(a, a_label, b, b_label):
    """Plot the results of our model

    a: first dataset to plot
    a_label: label for a
    b: second dataset to plot
    b_label: label for b
    """
    plot(a, label=a_label)
    plot(b, label=b_label)
    decorate(xlabel='Time (seconds)',
             ylabel='Temperature (degrees C)')
```

In [14]:
```
res = run_sim(system,600,update_Cush);
plot_2_results(res.tBurritoCush, 'Burrito', res.tCushion, 'Cushion')
```



After running our update function for 10 minutes (and changing it a number of times), it became clear that no matter what we set our parameters to, the burrito and cushion reached equilibrium in under 10 minutes. Our model ingores the effect of convection (because the burrito is served in a paper bag which reduces convection significantly), so our model states that the burrito reaches its final temperature of about 39 degrees in just over 3 minutes.

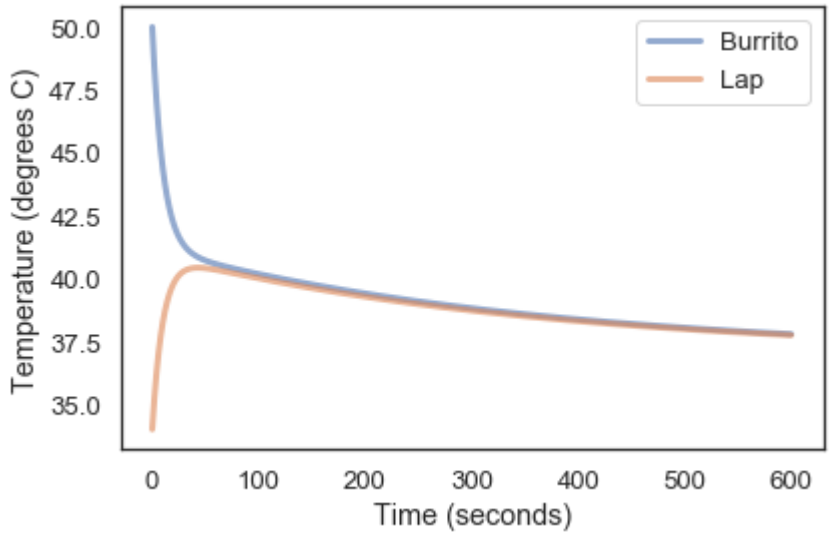
While this is clearly wrong, we feel comfortable ignoring the effect of air because the difference in temperatures between the two cases in question is small enough that air shouldn't affect one burrito much more or less than it affects the other.

The Tricky Burrito

While a burrito sitting on an insulating seat cushion can be modeled fairly simply, placing it on somebody's lap adds another level of dificulty. In addition to the behavior of the cushion, the human lap responds to homeostasis, always adjusting to stay at 34 degrees celcius (a few degrees colder than core body temp). Due to both time and educational constraints, we modeled homeostatis with Newton's law of cooling with r = 0.005, which gave us a curve that looked close to what we expected. Note that this number is not backed up by any scientific papers, as all of our k values are.

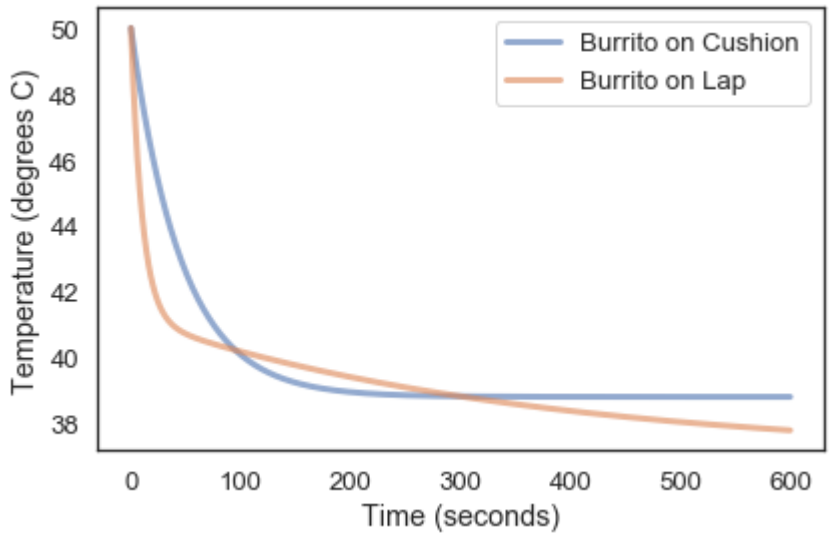
In [15]:
```
def update_Lap(state, t, system):

    """Updates states for the burrito on the lap for a timestep of 1 second

    state: state object of temperatures before update
    t: time
    system: all model parameters

    returns: new State object
    """
    new_state = state

    #Convert temperature to energy
    energy_B1 = state.tBurritoLap * system.C_burrito #Joules / gram
    energy_Lap = state.tLap * system.C_lap #Joules / gram

    #Calculate changes in energy
    energy_dB1 =  system.A * (state.tLap - state.tBurritoLap) / ((0.023 / syst
em.k_burrito ) + (0.05/system.k_lap ))
    energy_dLap = -1 * energy_dB1

    #Convert energy changes back to temperature
    dB1 = energy_dB1 / system.C_burrito
    dLap= energy_dLap / system.C_lap  + (0.005*(37 - state.tLap))

    #0.005 and 37 were hard coded in becasue at this point
    #it was easier to change those values in the update function than
    #in the system object. Also, our core DE's were changing often
    #enough that our actual parameters were changing (not just
    #their values), which made using our system object properly
    #a challenge

    #Update state
    new_state.tBurritoLap += dB1
    new_state.tLap += dLap

    return new_state
```

In [16]:
```
res = run_sim(system,600,update_Lap);
plot_2_results(res.tBurritoLap, 'Burrito', res.tLap, 'Lap')
```



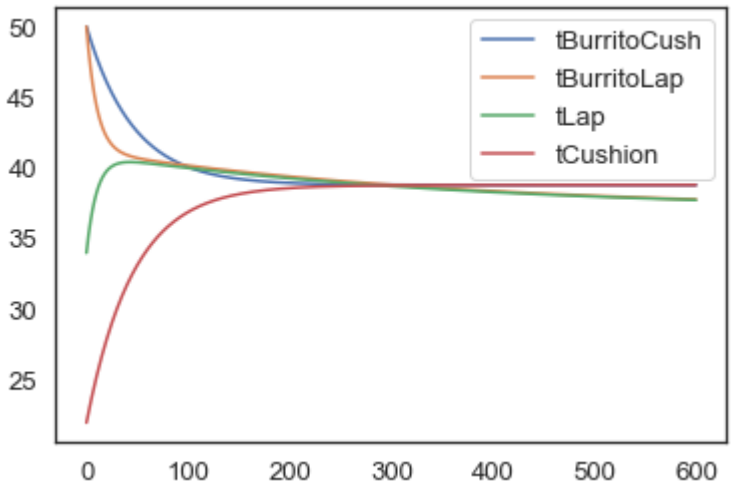
In [17]:
```
def update_burrito(state, t, system):
    """An update function that creates results for both the burrito stored in t
he lap and on the cushion

    state: state object of temperatures before update
    t: time
    system: all model parameters

    returns: new State object
    """
    new_state = update_Lap(state, t, system)
    new_state = update_Cush(new_state, t, system)
    return new_state
```

In [18]:
```
res = run_sim(system,600,update_burrito);
plot_2_results(res.tBurritoCush, 'Burrito on Cushion', res.tBurritoLap, 'Burri
to on Lap')
```



In [19]:
```
res.plot() #plot the graph of the combined burrito models
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b2e0c3e48>
```



Interpretation

Shown in the graph above, leaving the burrito on your lap is only favorable for trips between 2 and 5 minutes. For the vast majority of drives, leaving the burrito on the cushion next to you will keep it warmer. That said, our model was inexact, and given how close our results are (only about 1 degree off after 10 minutes), I don't think our model can confidently answer our initial question without more information.

Sources

- https://hypertextbook.com/facts/2001/AbantyFarzana.shtml (https://hypertextbook.com/facts/2001/AbantyFarzana.shtml)
- https://www.sciencedirect.com/science/article/pii/S026087740600392X (https://www.sciencedirect.com/science/article/pii/S026087740600392X)
- https://users.ece.utexas.edu/~valvano/research/Thermal.pdf (https://users.ece.utexas.edu/~valvano/research/Thermal.pdf)