Modeling the Blending of Different Types of Chocolate

Jaclyn Breier, Kathleen Flavin, Adam Mallette, Andrew Mikes

May 4, 2017

University of Notre Dame

Dept. of Chemical and Biomolecular Engineering, CBE 30338

Dr. Kantor

1. Problem Statement

This project aims to explore the chocolate production process, specifically concentrating on the process of blending the different ingredients while controlling the inlet stream flow rates and tank temperature. As a globally popular product, chocolate is produced through a multitude of manufacturing processes. Each process requires several raw materials and varying degrees of refinement. A successful process begins with harvesting, fermentation, and drying of cocoa beans to produce cocoa liquor. This cocoa liquor can be further refined into cocoa butter or other cocoa solids. Once these materials are obtained, they must be efficiently blended with milk, sugar, and vanilla to yield chocolate product. Emulsifiers are also added to prevent the contents of the chocolate from separating out.

The blending process proceeds depending on the type of chocolate desired. Each type of chocolate requires different ratios of materials and processing conditions to create the predetermined type. This project models the blending of these ingredients into a liquid form with three different grade changes: from dark chocolate to milk chocolate, milk chocolate to white chocolate, and white chocolate back to dark chocolate. For each of these changes, the inlet and temperature setpoints are adjusted, the system is controlled until steady state, and the setpoint and disturbance response are recorded. Using this data, it will be possible to determine the most sustainable method of using chocolate ingredients to lessen material waste.

The blending step is an imperative step in chocolate production because it prepares the chocolate solution for the remaining process steps: conching and tempering. During conching, the liquid chocolate is melted and passed through grinders to improve the consistency of the chocolate. Finally, the chocolate is tempered through heat treatment to obtain the proper physical characteristics. These ideal physical characteristics include resistance to melting, resistance to crumbling, and firmer texture. After this process is complete, the chocolate is ready to be packaged and shipped[1].

The technology challenge is to model this blending process accurately while maintaining the predetermined setpoint values of tank temperature and material ratios for each chocolate type. Since different types of chocolate are being produced, the process increases in complication with the variation in processing conditions for each type. The material composition acts as the main variation in each of the chocolate types; each type must be generated by a specific recipe that details the material ratios to be blended together. Furthermore, these different material constitutions result in differences of optimal mixing temperatures [2]. These conditions are imperative to control in order to create a quality chocolate product. This project will concentrate on using PID control to maintain the specific material ratios and optimal mixing temperatures for white, milk, and dark chocolate. In doing this, it is possible to determine the most sustainable method to blend various chocolate products.

Overall, the main project goal is to create a model system capable of accurately controlling the chocolate blending process with each grade change in chocolate type. To attain this goal it is necessary to implement 3 PID controllers, on the cocoa liquor inlet stream, the cocoa butter inlet stream, and tank temperature. The data from these PID controllers is then used to plot the setpoint and disturbance responses in each chocolate type grade change.

---

[1]:"The Production of Chocolate." The World Atlas of Chocolate. N.p., n.d. Web. 20 Apr. 2017. <https://www.sfu.ca/geog351fall03/groups-webpages/gp8/prod/prod.html>.

[2]: Williams, Pam, and Hallberg, Vincent. "Tempering Chocolate and Why." Lesson-Tempering Chocolate. Wilmor Publishing Corp, n.d. Web. 27 Apr. 2017 <https://www.ecolechocolat.com/en/chocolate-tempering.html>.

2. Theoretical Development

During the blending phase of chocolate production, various ingredients are combined in a tank to create the homogeneous mixture that eventually becomes either white, milk, or dark chocolate. The project describes this process by utilizing a continuous stirred tank reactor (CSTR) model and PID control. Initially the system is set to produce one specific type of chocolate and allowed to reach a pseudo-steady state with the help of 3 PID controllers. Once this is reached, the setpoint conditions are changed to produce a new chocolate type. As the blending tank transitions into producing a new chocolate type, the model records and plots the setpoint and disturbance responses. This process is repeated multiple times using different chocolate type transitions.

The type of chocolate produced is dictated by the composition of its various ingredients. Whether the product is white, milk, or dark chocolate depends on the various amounts of cocoa liquor, cocoa butter, sugar, milk, and vanilla with in the tank. Since it melts at about 33°C, the cocoa liquor is pumped in from a temperature controlled tank of 38°C. Along the same vein, the cocoa butter melts to its liquid form in a temperature controlled tank of above 34°C before entering the blending tank. When the butter enters the tank, it's inlet rate is controlled by a PID controller, responding to any deviation from the desired setpoint in the outlet stream. The milk solids and sugar are added at constant flow rates from their respective containers and dissolve in solution. Finally, the vanilla inlet flow rate is held constant, flowing in through a pipe. **Table 1** below gives the range of composition for each raw material that is used to make each type of chocolate. These values were estimated from the values found in the various sources displayed in **Appendix 1**.

Table 1: Chocolate Grade Changes

	White Chocolate (A)	Milk Chocolate (B)	Dark Chocolate (C)
Cocoa Butter	23-25%	15-18%	15-35%
Cocoa Liquor	0%	10-12%	25-35%
Milk Soilds	20-25%	13-15%	0%
Sugar	45-55%	55-62%	30-50%
Vanilla	~0.5%	~0.5%	~0.5%
Temperature	37-43°C	40-45°C	46-48°C

In order to achieve optimal blending, the project aims to control the material composition of the product by manipulating the flow of cocoa butter and cocoa liquor into the tank using feedback control. The key assumptions made include perfect mixing, constant volume of fluid in the tank, complete melting of the liquor and butter streams before blending, and constant streams of sugar, milk and vanilla into the tank. This assumption is reasonable because it is assumed that the supplier of each of these ingredients provided quality products with clearly articulated properties. However the same cannot be assumed for the cocoa liquor and cocoa butter because these ingredients are produced in the same facility where they are blended and therefore will have greater fluctuation in properties.

In terms of the temperature of the contents of the tank, it is assumed that the temperature restrictions account for the acceptable range of viscosities at which the chocolate can flow. These ranges are: for dark chocolate, 46-48°C, for milk chocolate, 40-45°C, and for white chocolate, 37-43°C [3]. Additionally since chocolate is a shear-thinning fluid, the problem of the viscosity becoming too high at any point in the process is not a concern, assuming that the temperature remains high enough throughout the process of pumping the fluid such that the chocolate does not solidify to any significant extent. The melting point of chocolate is assumed to be 32°C. Therefore in tuning the system, it is essential that the tank temperature never drops below this value.

As previously stated, the system is modelled as a CSTR with 3 PID controllers. Five inlet streams represent the cocoa liquor, cocoa butter, milk, sugar, and vanilla inputs to the system. The outlet stream represents the chocolate product. The control scheme utilizes feedback control by measuring the cocoa content in the product stream, using a method of spectroscopy (for example, photoacoustic spectroscopy [4]), and then adjusting the flow rates of the two varying input streams using separate PID controllers to bring the system to a setpoint. Additionally, there is a level controller that keeps the volume of the tank constant by manipulating the flow rate of the product stream. Finally, a heating coil is used to maintain the temperature of the contents of the tank at a specific temperature, accounting for disturbances. This control scheme is demonstrated by **Figure 1**.

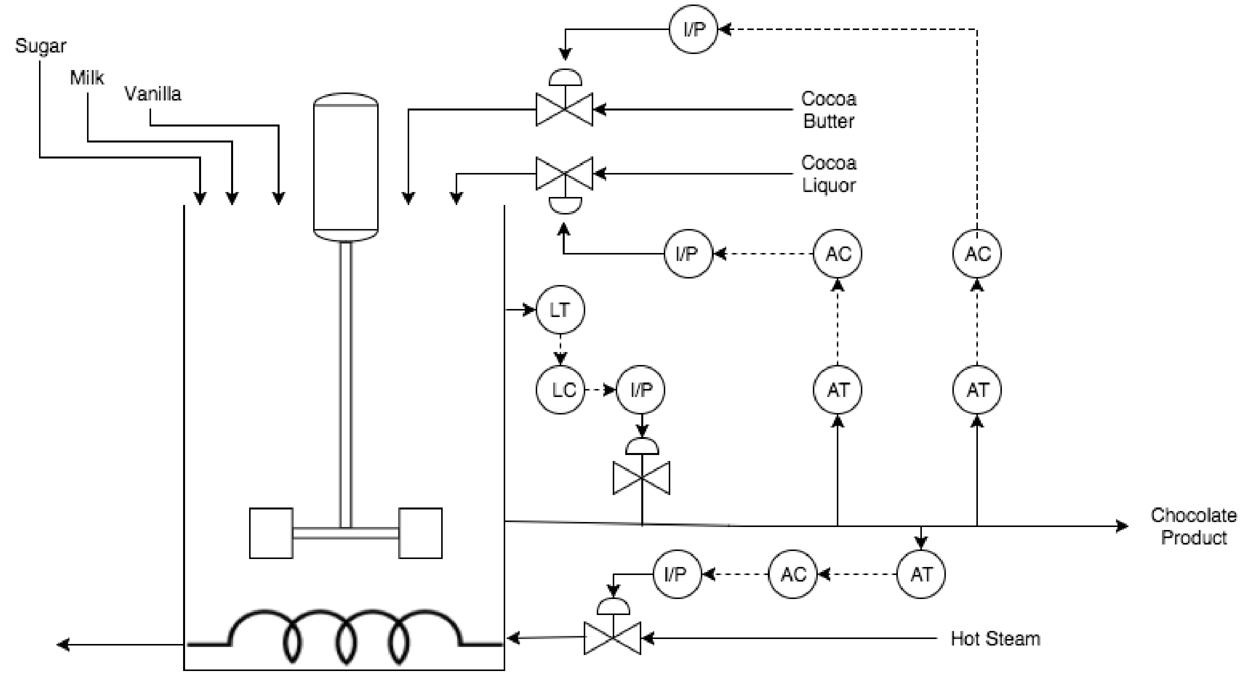


Figure 1: Blender Control Scheme

In order to accurately quantify the dynamics of the Chocolate Blending System, multiple mass balances and an energy balance are used to model the inlets and outlets of the system. The following variables were used in these equations:

$$\begin{aligned} Cocoa\ Liquor &= C \\ Cocoa\ Butter &= B \\ Sugar &= S \\ Milk &= K \\ Vanilla &= V_n \end{aligned}$$

First, the average density of the system was found using the following equation, where ρ_i is density and w_i is a mass fraction of component i

$$\rho_{avg} = \frac{1}{\frac{w_C}{\rho_C} + \frac{w_B}{\rho_B} + \frac{w_S}{\rho_S} + \frac{w_K}{\rho_K} + \frac{w_{V_n}}{\rho_{V_n}}}$$

Using this information, the volumetric flowrate of product out of the tank was modelled with the next equation, where q is a volumetric flow rate and m is mass flow rate

$$q_{out} = \frac{m_{out}}{\rho_{avg}}$$

Next, the following mass balances were determined to understand the mass flow into and out of the tank of each compositon. V is volume.

$$\frac{d(w_C)}{dt} = \frac{m_C}{\rho_{avg}V} - \frac{w_C q_{out}}{V}$$
$$\frac{d(w_B)}{dt} = \frac{m_B}{\rho_{avg}V} - \frac{w_B q_{out}}{V}$$
$$\frac{d(w_S)}{dt} = \frac{m_S}{\rho_{avg}V} - \frac{w_S q_{out}}{V}$$
$$\frac{d(w_K)}{dt} = \frac{m_K}{\rho_{avg}V} - \frac{w_K q_{out}}{V}$$
$$\frac{d(w_{V_n})}{dt} = \frac{m_{V_n}}{\rho_{avg}V} - \frac{w_{V_n} q_{out}}{V}$$

Finally, an energy balance was used to determine the heat flow in the tank after the inlet streams were added to the tank at various temperatures. This energy balance is demonstrated in the following equation, where T_{iC} is the temperature of the cocoa liquor input stream, T_{iB} is the temperature of the cocoa butter input stream, and T_{RT} is room temperature. C_{Pi} is the heat capacity of component i.

$$\frac{d(T)}{dt} = (m_C C_{PC}(T_{iC} - T_{tank}) + m_B C_{PB}(T_{iB} - T_{tank}))+\ldots$$
$$\ldots+(m_S C_{PS} + m_K C_{PK} + m_{V_n} C_{PV_n})(T_{RT} - T_{tank}) + \frac{m_{Steam} C_{PSteam}(T_{Steam} - T_{tank})}{V\rho_{product} C_{Pproduct}}$$

As previously stated, the project aims to use this model to analyze the system response to disturbance variables and determine optimal tuning conditions for different product properties. The project will plot the response to any disturbances in such a way that the fluctuation in steady state response is minimized.

The project must account for temperature variation as both the inlet temperature of cocoa liquor and the setpoint temperature of the contents in the tank can change. A third PID controller is needed to maintain the temperature of the mixing tank at a constant value. This is implemented with feedback control measuring the temperature of the outlet stream and sending control signals to the hot water inlet of the heating coil.

---

[3]: Townsend, Adam K. The Fluid Dynamics of Chocolate Fountains. Publication. Dept. of Mathematics, U College London, 14 July 2012. Online. <http://adamtownsend.com/wp-content/uploads/2012/06/Chocolate-Fountains-Synopsis.pdf>.

[4]:Nieburg, Oliver. "New Method to Measure Chocolate's Cocoa Content." ConfectioneryNews.com. N.p., 10 June 2013. Web. 30 Apr. 2017. <http://www.confectionerynews.com/R-D/New-method-to-measure-chocolate-s-cocoa-content>.

3. Primary Results

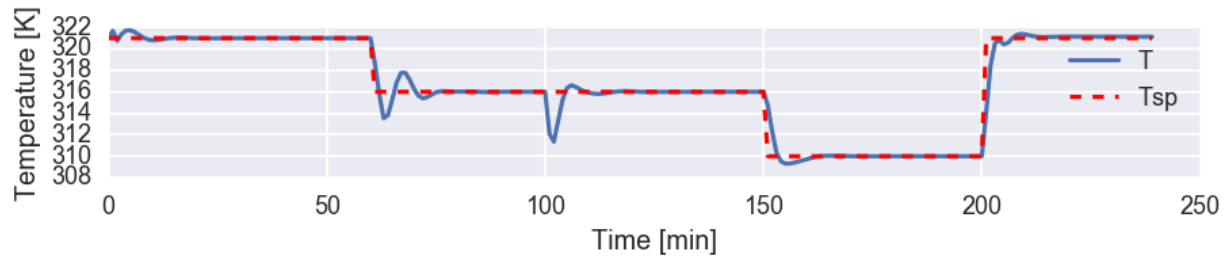
The project determines that the chocolate blending process can be modelled and controlled using 3 PID controllers. The ideal control values were optimized to be as follows:

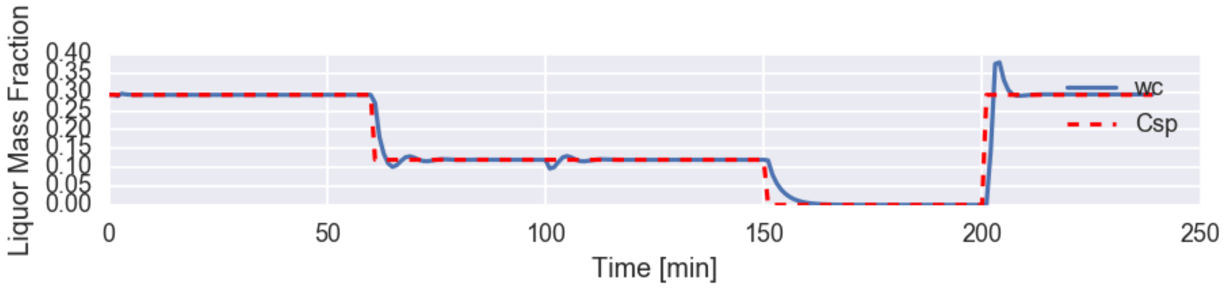
for the temperature PID, $K_p = 10$, $K_i = 35$, and $K_d = 0.5$

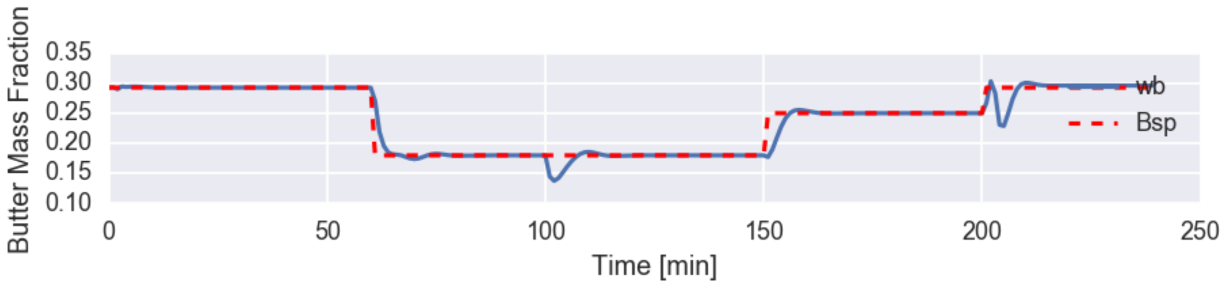
for the cocoa liquor PID, $K_p = 500$, $K_i = 3000$, and $K_d = 50$

for the cocoa butter PID, $K_p = 300$, $K_i = 1000$, and $K_d = 50$

These controllers are able to accurately track the setpoint as seen in the following graphs. These graphs show the real value of the measured parameter as well as setpoint value as a function of time. There is a setpoint change at 60 minutes,at 150 minutes, and at 200 minutes. These represent a transition from the production of dark chocolate to milk chocolate, from milk chocolate to white chocolate, and from white chocolate back to dark chocolate. The system is also forced to react to a disturbance variable in the form of an excess flow of sugar into the tank at 100 minutes of elapsed time.

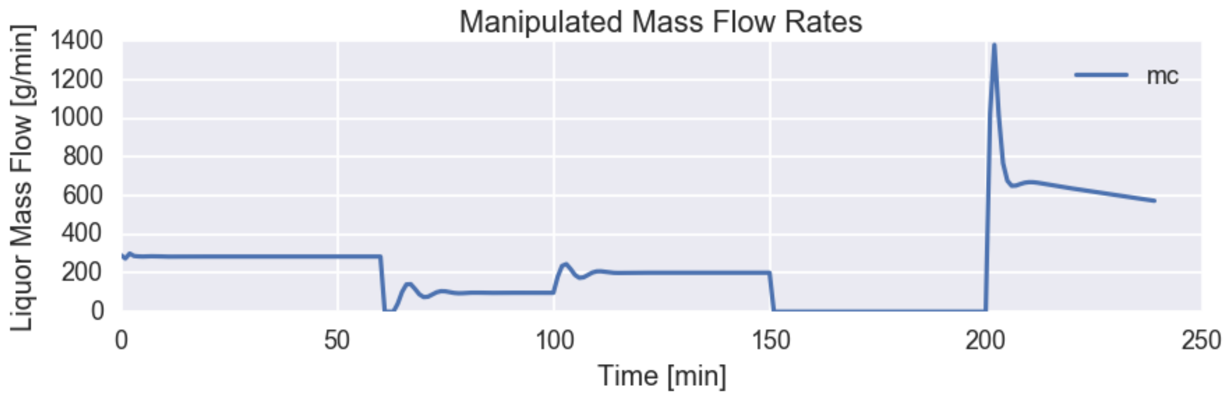


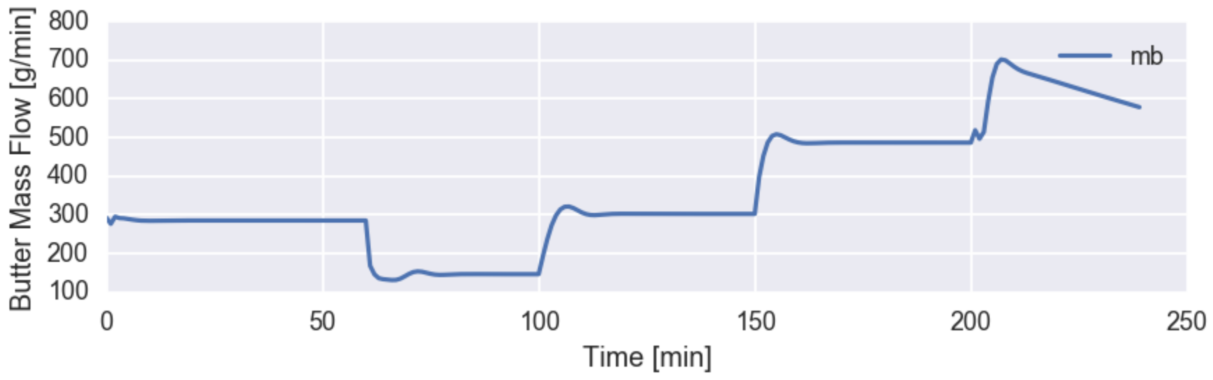


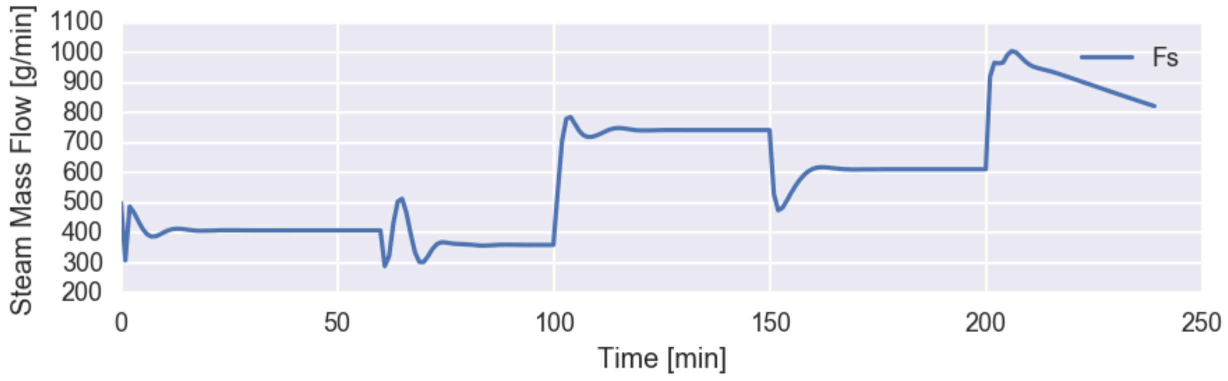


Secondary Results

The following graphs show how the manipulated variables of this system changed as a function of time. These manipulated variables are the mass flows of cocoa liquor, cocoa butter, and steam. These control the mass fractions of the cocoa liquor and cocoa butter and also control the tank temperature.







These results display that the model was able to successfully maintain stability between grade changes of chocolate type. This is evidenced by the low degree of overshoot and limited oscillations during each grade change. This model creates a sustainable grade change system that uses the ingredients already in a blending tank to produce a new type of chocolate without having to empty and clean the tank between each transition. Reasonably wide ranges of acceptable limits exist for composition ratios in each chocolate type; therefore as long as the concentration of each ingredient is near its setpoint, the chocolate will be acceptable to sell. With the control scheme above, the down-time of production is minimal because the controllers are able to adjust the inlet stream flow rates of cocoa liquor and cocoa butter to a value near the new desired setpoint within only a few minutes. This minimizes the waste of chocolate ingredients by limiting the time that the CSTR is producing chocolate that does not fit under the category of white, milk, or dark chocolate.

The primary issue in the response as displayed above occurs in the transition from producing white chocolate to dark chocolate. However, since there is such a wide range of acceptability for cocoa butter and cocoa liquor, the delay in quality chocolate production is only a few minutes. This occurs as a result of the interrelationships between cocoa butter concentration and cocoa liquor concentration. This showcases a factor that is very difficult for the PID controllers to handle; in transitioning from white chocolate to dark chocolate the cocoa liquor flow rate must drastically increase. In doing so, the cocoa butter concentration drops rapidly. However, the PID controller quickly responds to this change.

Finally, slight ringing occurs as a response to both disturbances and setpoint changes, but they produce negligible effects on the quality of the chocolate produced. It is assumed that the stream controllers can physically handle these rates of change.

4. Executable Element

The following code simulates the blending of the different grade changes while plotting the setpoint and disturbance responses. The following changes and disturbances are implemented in the system as it is run in the executed code:

At $t = 60\ min$, the process setpoints change to end producing dark chocolate and begin producing milk chocolate.

At $t = 100\ min$, a significant amount of excess sugar is dumped into the tank accidentally, which is done to model a disturbance.

At $t = 150\ min$, the process setpoints change to end producing milk chocolate and begin producing white chocolate.

At $t = 200\ min$, the process setpoints change to end producing white chocolate and begin producing dark chocolate.

In [13]:
```
%matplotlib inline
!pip install seaborn
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import odeint
from scipy.interpolate import interp1d
import seaborn as sns
sns.set_context('talk')
```

```
Requirement already satisfied (use --upgrade to upgrade): seaborn in c:\users\a
ndrew\anaconda3\lib\site-packages
```

```
You are using pip version 8.1.2, however version 9.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' com
mand.
```

In [14]:
```
# mass fractions for components in final blend (currently set for dark chocolate)
wc = .2925     # liquor
wb = .2925     # butter
ws = .4        # sugar
wk = 0         # milk
wv = .005      # vanilla
wp = .595      # cocoa in final product

# Densities of each component
c1b = .3333      # g/cm^3 liquor (bad value, used cocoa powder)
c2b = .92143     # g/cm^3 butter
c3b = 1.59       # g/cm^3 sugar
c4b = 1.033      # g/cm^3 milk
c5b = .88        # g/cm^3 vanilla
c6b = (wc*c1b+wb*c2b+ws*c3b+wk*c4b+wv*c5b)    # g/cm^3 product (weighted average)

# heat capacities of each component
cp1 = 1.87     # J/(g*K) liquor
cp2 = 1.78     # J/(g*K) butter weighted average of components
cp3 = 1.33     # J/(g*K) sugar
cp4 = 3.93     # J/(g*K) milk
cp5 = 2.5      # J/(g*K) vanilla
cp6 = 1.432    # J/(g*K) product
cps = 1.871    # J/(g*K) @ 325 K steam

# Initial Temperatures (aribtitrarily set)
Tti = 321         # K initial tank temperature (Dark Chocolate)
Tc = 308          # K
Tb = 311          # K
Tr = 293          # K room temperature
Tsi =  356.67     # K  initial steam temperature

# Steady State Flow Rates (mass/time)
mout = 1000         # g/min (arbitrarily set) total flow rate into and out of the
mc =wc*mout         # g/min liquor
mb = wb*mout        # g/min butter
ms = ws*mout        # g/min sugar
mk = wk*mout        # g/min milk
mv = wv*mout        # g/min vanilla
Fs = 500            # g/min steam

# Tank Parameters (arbitrarily set)
r = 8.921            # cm tank radius
h= 20                # cm liquid height
V = np.pi*(r**2)*h   # cm^3 volume

# time variable for use in the ode's
t = np.linspace(0,240,500)

# defines the differential equations that model the system.
def deriv(X,t):
    wc,wb,ws,wk,wv,T = X         # mass fractions and temperature
    pavg = 1/((wc/c1b)+(wb/c2b)+(ws/c3b)+(wk/c4b)+(wv/c5b))    # average density
    qout = mout/pavg                  # volumetric flow rate out of the tank
    dwC = mc/(pavg*V) - wc*qout/V     # liquor  differential equation
    dwB = mb/(pavg*V) - wb*qout/V     # butter  diff eq
    dwS = ms/(pavg*V) - ws*qout/V     # sugar   diff eq
    dwK = mk/(pavg*V) - wk*qout/V     # milk    diff eq
    dwVn = mv/(pavg*V) - wv*qout/V    # vanilla diff eq
    dwT = (mc*cp1*(Tc-T)+(mb*cp2*(Tb-T))+(ms*cp3+mk*cp4+mv*cp5)*(Tr-T)
           +Fs*cps*(Tsi-T))/(V*c6b*cp6) #temp diff eq
    return [dwC,dwB,dwS,dwK,dwVn,dwT]
```

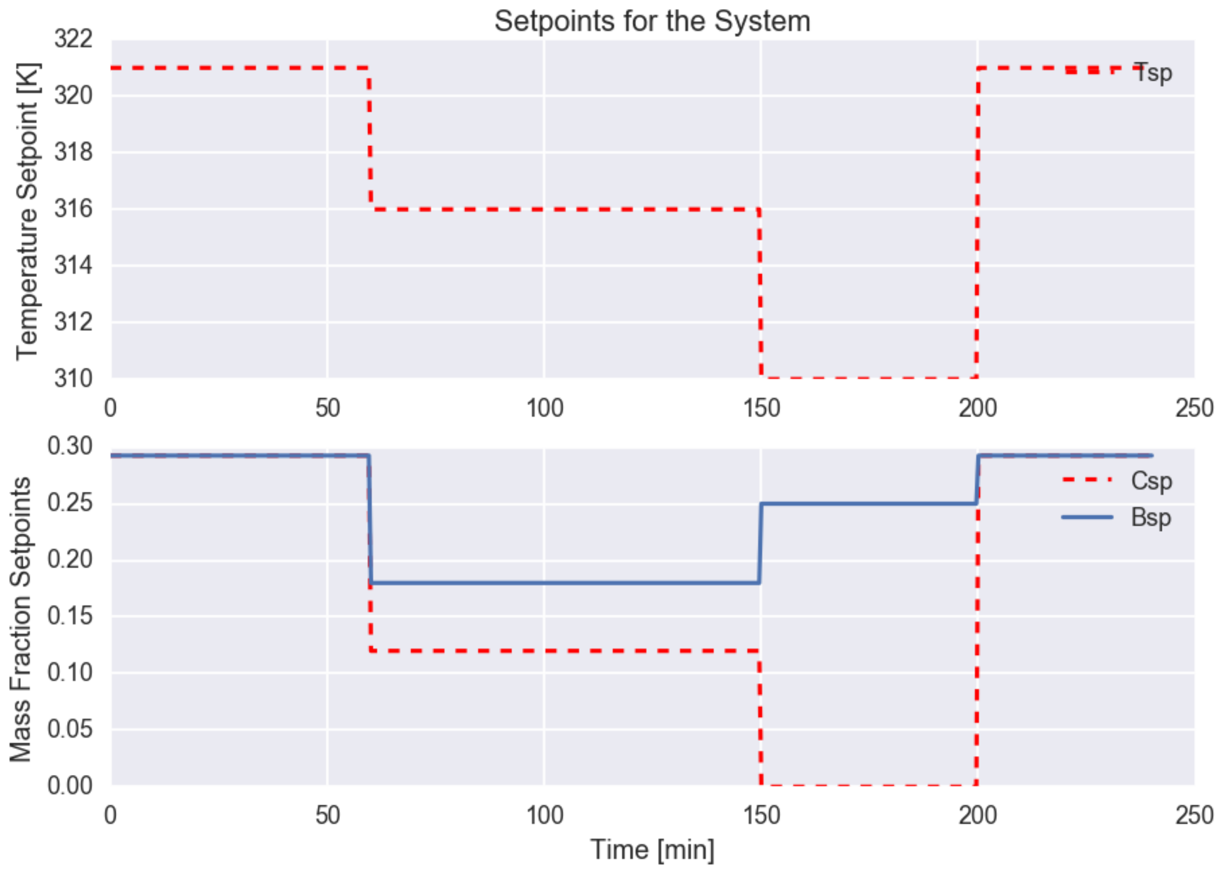
PID Control

In [15]:
```
import matplotlib.pyplot as plt
import numpy as np

class PID:
    """ An implementation of a PID control class for use in process control simul
    """
    def __init__(self, name=None, SP=None, Kp=0.2, Ki=0, Kd=0, beta=1, gamma=0, M
        self.name = name
        self.SP = SP
        self.Kp = Kp
        self.Ki = Ki
        self.Kd = Kd
        self.beta = beta
        self.gamma = gamma
        self.MVrange = MVrange
        self.DirectAction = DirectAction
        self._mode = 'inAuto'
        self._log = []
        self._errorP0 = 0
        self._errorD0 = 0
        self._errorD1 = 0
        self._lastT = 0
        self._currT = 0

    def auto(self,t=None,SP=None):
        """Change to automatic control mode.
        """
        self.SP = SP
        self._mode = 'inAuto'

    def manual(self,t):
        """Change to manual control mode. In manual mode, the setpoint tracks the
        variable to provide bumpless transfer on return to auto.
        """
        self._lastT = t
        self._mode = 'inManual'

    def _logger(self,t,SP,PV,MV):
        self._log.append([t,SP,PV,MV])

    def plot(self):
        """Create historical plot of SP,PV, and MV using the controller's interna
        """
        dlog = np.asarray(self._log).T
        t,SP,PV,MV = dlog
        plt.subplot(2,1,1)
        plt.plot(t,PV,t,SP)
        plt.title('Process Variable')
        plt.xlabel('Time')
        plt.legend(['PV','SP'])
        plt.subplot(2,1,2)
        plt.plot(t,MV)
        plt.title('Manipulated Variable')
        plt.xlabel('Time')
        plt.tight_layout()
    @property
    def beta(self):
        """beta is the setpoint weighting for proportional control where the prop
        is given by error_proportional = beta*SP - PV. The default value is one.
        """
        return self._beta

    @beta.setter
    def beta(self,beta):
        self._beta = max(0.0,min(1.0,beta))

    @property
    def DirectAction(self):
        """DirectAction is a logical variable setting the direction of the contro
        value means the controller output MV should increase for PV > SP. If Fals
        is reverse acting, and ouput MV will increase for SP > PV. IFf the steady
        process gain is positive then a control will be reverse acting.

        The default value is False.
        """
        return self._DirectAction

    @DirectAction.setter
    def DirectAction(self,DirectAction):
        if DirectAction:
            self._DirectAction = True
            self._action = +1.0
        else:
            self._DirectAction = False
            self._action = -1.0

    @property
    def gamma(self):
        """gamma is the setpoint weighting for derivative control where the deriv
        is given by gamma*SP - PV.  The default value is zero.
        """
        return self._gamma

    @gamma.setter
    def gamma(self,gamma):
        self._gamma = max(0.0,min(1.0,gamma))

    @property
    def Kp(self):
        """Kp is the proportional control gain.
        """
        return self._Kp

    @Kp.setter
    def Kp(self,Kp):
        self._Kp = Kp

    @property
    def Ki(self):
        """Ki is the integral control gain.
        """
        return self._Ki
    @Ki.setter
    def Ki(self,Ki):
        self._Ki = Ki

    @property
    def Kd(self):
        """Kd is the derivative control gain.
        """
        return self._Kd

    @Kd.setter
    def Kd(self,Kd):
        self._Kd = Kd

    @property
    def MV(self):
        """MV is the manipulated (or PID outpout) variable.
        """
        return self._MV

    @MV.setter
    def MV(self,MV):
        self._MV = max(self._MVmin,min(self._MVmax,MV))

    @property
    def MVrange(self):
        """range is a tuple specifying the minimum and maximum controller output.
        Default value is (0,100).
        """
        return (self._MVmin,self._MVmax)

    @MVrange.setter
    def MVrange(self,MVrange):
        self._MVmin = MVrange[0]
        self._MVmax = MVrange[1]

    @property
    def SP(self):
        """SP is the setpoint for the measured process variable.
        """
        return self._SP

    @SP.setter
    def SP(self,SP):
        self._SP = SP

    @property
    def PV(self):
        """PV is the measured process (or control) variable.
        """
        return self._PV

    @PV.setter
    def PV(self,PV):
        self._PV = PV
    def update(self,t,SP,PV,MV):
        self.SP = SP
        self.PV = PV
        self.MV = MV
        if t > self._lastT:
            dt = t - self._lastT
            self._lastT = t
            if self._mode=='inManual':
                self.SP = PV
            self._errorP1 = self._errorP0
            self._errorP0 = self.beta*self.SP - self.PV
            self._errorI0 = self.SP - self.PV
            self._errorD2 = self._errorD1
            self._errorD1 = self._errorD0
            self._errorD0 = self.gamma*self.SP - self.PV
            if self._mode=='inAuto':
                self._deltaMV = self.Kp*(self._errorP0 - self._errorP1) \
                    + self.Ki*dt*self._errorI0 \
                    + self.Kd*(self._errorD0 - 2*self._errorD1 + self._errorD2)/d
                self.MV -= self._action*self._deltaMV

        self._logger(t,self.SP,self.PV,self.MV)
        return self.MV
```

In [16]:
```
tbreak = [0,60, 60.00001,65,70,75,80,85,90,95,100,110,120,130,140,150, 150.00001,
          155,160,165,170,175,180,185,190,200,200.00001,210,220,230,240]   # time
Tbreak = [321,321, 316,316,316,316,316,316,316,316,316,316,316,316,316,316,
          310,310,310,310,310,310,310,310,310,310,321,321,321,321,321]     # temp
wcbreak= [.2925,.2925, .12,.12,.12,.12,.12,.12,.12,.12,.12,.12,.12,.12,.12,.12,
          0,0,0,0,0,0,0,0,0,0,.2925,.2925,.2925,.2925,.2925]             # liquor
wbbreak= [.2925,.2925, .18,.18,.18,.18,.18,.18,.18,.18,.18,.18,.18,.18,.18,.18,
         .25,.25,.25,.25,.25,.25,.25,.25,.25,.25,.2925,.2925,.2925,.2925,.2925]


Tsp = interp1d(tbreak,Tbreak,kind='linear')        # makes the setpoints into plo
Csp = interp1d(tbreak,wcbreak,kind='linear')
Bsp = interp1d(tbreak,wbbreak,kind='linear')
print(len(tbreak))
t = np.linspace(0,240,500)                       # plots the setpoints

plt.subplot(2,1,1)
plt.plot(t,Tsp(t),'r--')
plt.title('Setpoints for the System')
plt.ylabel('Temperature Setpoint [K]')
plt.legend(['Tsp'])

plt.subplot(2,1,2)
plt.plot(t,Csp(t),'r--')
plt.plot(t,Bsp(t))
plt.ylabel('Mass Fraction Setpoints')
plt.xlabel('Time [min]')
plt.legend(['Csp','Bsp'])
```

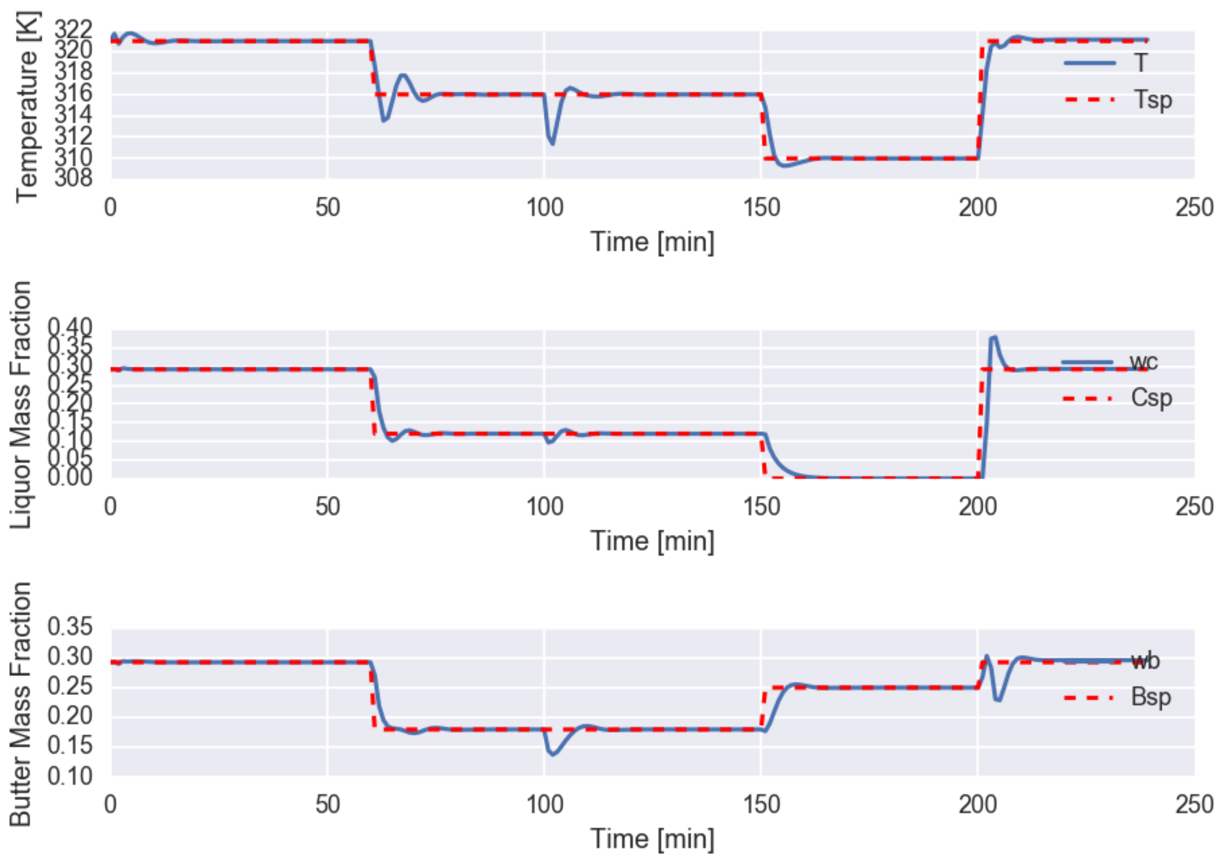
```
31
```

Out[16]:
```
<matplotlib.legend.Legend at 0x2a5c214ff28>
```

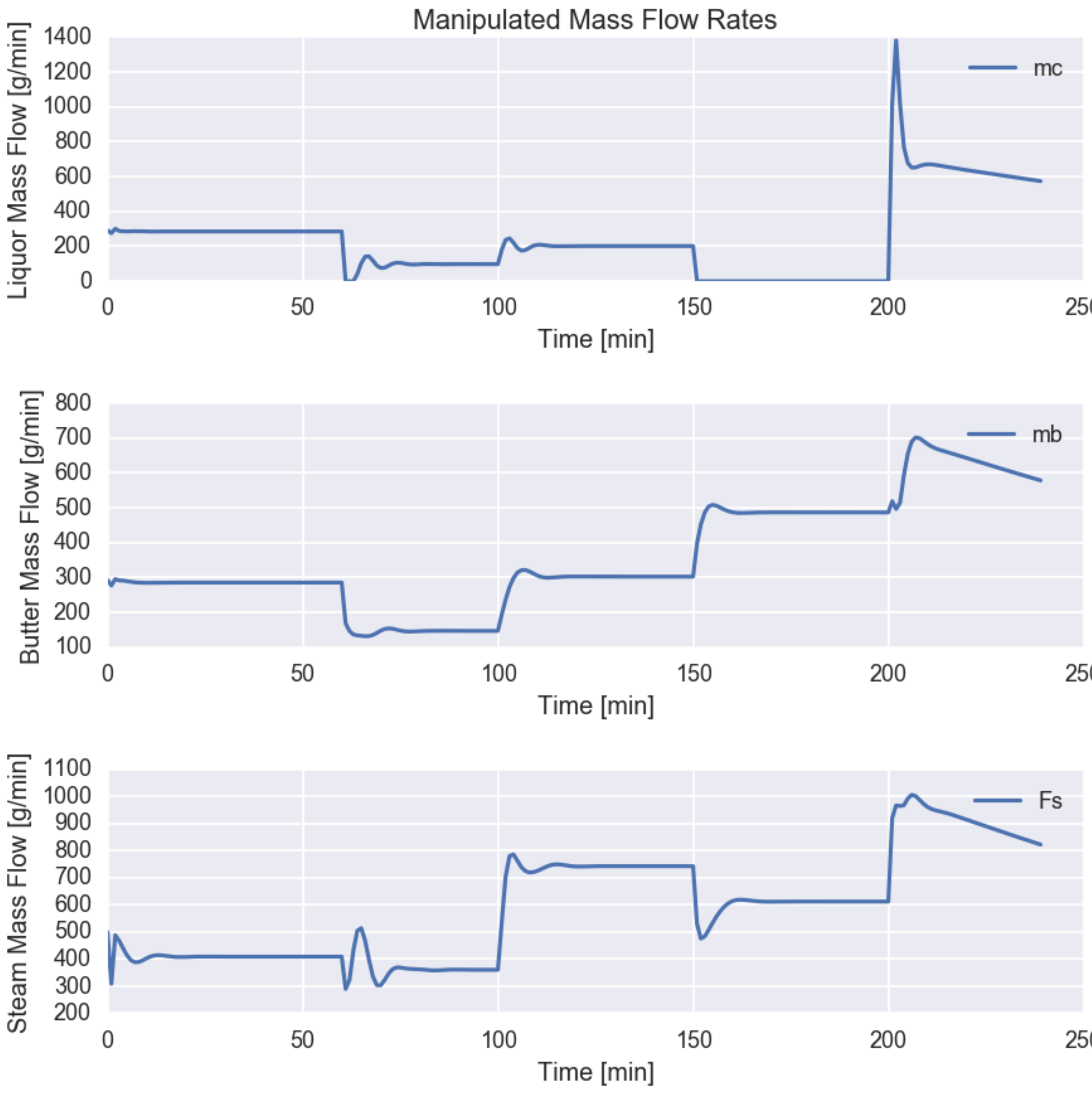

Setpoints for the System
Temperature Setpoint [K]
322
320
318
316
314
312
310
Tsp
0
50
100
150
200
250
Mass Fraction Setpoints
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Csp
Bsp
Time [min]

In [17]:
```
dt = 1       # sample time

log = []                             # initialize data log
I = 0                                # initialize error sum
IC = [.2925,.2925,.4,0,.005,321]     #initial conditions
wc,wb,ws,wk,wv,T = IC                # initial condition on states, sets initial va
Fs = 500                              # initial value of steam mass flow
mc = wc*mout              # initial value of liquor mass flow
mb = wb*mout              # initial value of butter mass flow

# temperature PID controller
tPID = PID(Kp = 10, Ki=35, Kd=.5, MVrange=(0,10000))

# liquor PID controller
cPID = PID(Kp = 500, Ki = 3000, Kd = 50, MVrange=(0,10000),beta=1)

# butter PID controller
bPID = PID(Kp =300, Ki =1000, Kd = 50, MVrange = (0,10000))

for t in np.arange(0,240,dt):
    if t >= 60:               # adjust the initial conditions for the change in s
        ws,wk,wv = .55, .145, .005
        ms,mk,mv = ws*mout,wk*mout,wv*mout
    if t == 100:              # models the introduction of a disturbance into the
        ms = 2*mout
    if t >= 150:              # adjust the initial conditions for the change in s
        ws,wk,wv = .5, .245, .005
        ms,mk,mv = ws*mout,wk*mout,wv*mout
    if t>=200:
        ws,wk,wv = .4,0,.005
        ms,mk,mv = ws*mout, wk*mout, wv*mout
    mout = mc+mb+ms+mk+mv     # adjust the flow out of the tank to maintain const
    tPID.SP = Tsp(t)
    cPID.SP = Csp(t)
    bPID.SP = Bsp(t)
    Fs = tPID.update(t,Tsp(t),T,Fs)       # adjust mass flow of steam to control
    mc = cPID.update(t,Csp(t),wc,mc)      # adjust input mass flow of liquor to c
    mb = bPID.update(t,Bsp(t),wb,mb)      # adjust input mass flow of butter to c
    log.append([t,T,Fs,wc,mc,wb,mb])      # stores the values of time, temperatur
    wc,wb,ws,wk,wv,T = odeint(deriv,[wc,wb,ws,wk,wv,T],[t,t+dt])[-1]     # calcul

t,T,Fs,wc,mc,wb,mb = np.asarray(log).T

fig=plt.figure(figsize=(5,5), dpi= 80, facecolor='w', edgecolor='k')

plt.figure()

plt.subplot(3,1,1)
plt.plot(t,T,t,Tsp(t),'r--')
plt.legend(['T','Tsp'])
plt.xlabel('Time [min]')
plt.ylabel('Temperature [K]')


plt.subplot(3,1,2)
plt.plot(t,wc,t,Csp(t),'r--')
plt.legend(['wc','Csp'])
plt.xlabel('Time [min]')
plt.ylabel('Liquor Mass Fraction')

plt.subplot(3,1,3)
plt.plot(t,wb,t,Bsp(t),'r--')
plt.legend(['wb','Bsp'])
plt.xlabel('Time [min]')
plt.ylabel('Butter Mass Fraction')

plt.subplots_adjust(hspace=1)
```

```
<matplotlib.figure.Figure at 0x2a5c1553048>
```



```
In [18]: fig=plt.figure(figsize=(10,10), dpi= 100, facecolor='w', edgecolor='k')

         plt.subplot(3,1,1)
         plt.plot(t,mc)
         plt.legend(['mc'])
         plt.xlabel('Time [min]')
         plt.ylabel('Liquor Mass Flow [g/min]')
         plt.title('Manipulated Mass Flow Rates')

         plt.subplot(3,1,2)
         plt.plot(t, mb)
         plt.legend(['mb'])
         plt.xlabel('Time [min]')
         plt.ylabel('Butter Mass Flow [g/min]')

         plt.subplot(3,1,3)
         plt.plot(t, Fs)
         plt.legend(['Fs'])
         plt.xlabel('Time [min]')
         plt.ylabel('Steam Mass Flow [g/min]')

         plt.subplots_adjust(hspace=.5)
```



5. Conclusions and Summary

The project demonstrates that the chocolate blending system can be accurately modeled and successfully controlled using 3 PID controllers for the temperature and the inlet streams of cocoa butter and cocoa liquor. This is evidenced by the fact that the model reacts well to both setpoint changes and disturbances, quickly reaching a steady state for temperature and composition. At all points throughout the process, the tank temperature remains within the acceptable range such that the chocolate product is molten but not burning. Additionally, the product has acceptable composition for the desired type of chocolate at all times with the exception of a few-minute time delay after a setpoint change. Finally, it is concluded that the system response follows the setpoint well at all places, and the points in time at which the response does not perfectly match the setpoint do not correspond to any states that could be problematic.

6. Appendix 1

The following is a list of sources used for the various physical properties of the substances used in the control scheme. This includes the sources for chocolate compositions, heat capacities, densities, etc.

CFR - Code of Federal Regulations Title 21." Accessdata.fda.gov. Food & Drug Administration, 21 Sept. 2016. Web. 15 Apr. 2017. <https://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=163>.

"Cocoa Butter and Cocoa Solids." TheChocolateLife. N.p., n.d. Web. 30 Apr. 2017. <https://www.thechocolatelife.com/community/forums/tasting-notes/1244/cocoa-butter-and-cocoa-solids>.

"Cocoa Butter Density." Aqua-Calc. N.p., n.d. Web. 30 Apr. 2017. <http://www.aqua-calc.com/page/density-table/substance/oil-coma-and-blank-cocoa-blank-butter>.

Davis, Mike, and Sara Bir. "Chocolate and Vanilla." OregonLive.com. N.p., 02 Dec. 2008. Web. 30 Apr. 2017. <http://www.oregonlive.com/foodday/index.ssf/2008/12/chocolate_and_vanilla.html>.

Depypere, Elisa. "White, Milk And Dark Chocolate." Belgian Smaak. N.p., 14 May 2014. Web. 30 Apr. 2017. <http://www.belgiansmaak.com/white-milk-and-dark-chocolate/>.

Jayes, Wayne. "Specific Heat Capacity." Sugar - Specific Heat Capacity. N.p., n.d. Web. 30 Apr. 2017. <http://www.sugartech.com/heatcapacity/index.php>.

"Liquids and Fluids - Specific Heats." The Engineering ToolBox. N.p., n.d. Web. 30 Apr. 2017. <http://www.engineeringtoolbox.com/specific-heat-fluids-d_151.html>.

Wolke, Robert L. "Chocolate By the Numbers." The Washington Post. WP Company, 9 June 2004. Web. 30 Apr. 2017. <http://www.washingtonpost.com/wp-dyn/articles/A24276-2004Jun8.html>.