Dynamic Optimization and Product Scheduling

Submitted to

Dr. Hedengren

Brigham Young University

April 15, 2016

Andrew Fonda

Luke Wilding

Matthew Johnson

Trevor Blanchard

Abstract

Production scheduling and process control have been two separate sciences with a similar goal, to optimize economic gain. Very few efforts have been made to integrate the two, but significant economic benefits would arise from doing so. The goal of this paper is to integrate the control of a continuously stirred tank reactor (CSTR) with a simple 72 hour optimized schedule to maximize profit while meeting a required demand schedule.

The CSTR is controlled by changes in cooling temperature to meet certain product purity concentrations. Transitions between product grades are imposed by changing the coolant temperature of the reactor. The transition times from the CSTR problem are used in the scheduling problem to give a cost of changing the product being produced.

The scheduling problem takes a demand schedule and profit margins in order to make production decisions. These production decisions decide what temperature set point should be used with the CSTR, and when the set point should be made. The optimization problem is run over a 72 hour period to simulate daily demand constraints. The scheduling problem successfully handles different demand schedules and prices for each product in order to give the required cooling temperature set point changes to maximize profit.

Introduction and Literature Review

The project sought first to understand the limited research done on the topic by recreating the benchmark continuous stirred tank reactor (CSTR) prototype discussed in a review article by Michael Baldea and Liro Harjunkoski entitled “Integrated production scheduling and process control: A systematic review.” The CSTR in this benchmark problem can make three different products: low-grade P1 corresponding to 80% conversion, medium-grade P2 (85% conversion), and premium P3 (94% conversion) [1]. Our work sets out to recreate the CSTR model from Baldea’s paper and to incorporate a scheduling model using APMonitor programming language in Python. The scheduling model will seek to maximize profits by determining when different product purities should be produced in order to minimize transition time (between purities) and storage time. Transition times for changes in the temperature set point were found by modeling the CSTR problem described previously. Constants relating to the kinetics and heat transfer of the CSTR were changed to be more realistic, which helped the problem to converge to reasonable solutions. No estimation was required for either model because all variables are known or will be manipulated by the optimizer.

Methods and Model Design

The project is broken into three pieces: creating a reactor model, creating a scheduling model, and interfacing the scheduling and reactor models. The following equations describe the dynamics of the CSTR. Equation 1 is the material balance for the reactor while equation 2 is the energy balance that describes the temperature dependence of the reaction. A benchmark CSTR problem provided values for all the parameters seen in the equations below except for reactor concentration (C_A), reactor temperature (T), and cooling jacket temperature (T_c).

$$\frac{dC_A}{dt} = \frac{q}{V}(C_{A0} - C_A) - k_0 e^{-E_A/RT}\ C_A \quad (1)$$

$$\frac{dT}{dt} = \frac{q}{V}(T_0 - T) - \frac{1}{\rho C_p} k_0 e^{-E_A/RT}\ C_A \Delta H - \frac{UA}{V\rho C_p}(T - Tc) \quad (2)$$

The only manipulated variable is T_c. The table below identifies the remaining CSTR parameters.

Table 1: Equation variables

Variable	Definition	Variable	Definition
V	Volume of Reactor	**UA**	Overall Heat Transfer Coefficient x Transfer Area
EA	Activation Energy	**delta H**	Change in enthalpy
R	Ideal Gas Constant	**q**	Inlet flow rate
k_0	Pre-exponential Factor	**CA**	Concentration of A
ρ	Reaction Mixture Density	**CA_0**	Initial Concentration of A
C_p	Reaction Mixture Heat Capacity	**T**	Reactor Temperature

Equation 3 below shows the objective function used in the scheduling problem where profit (J) is maximized. In the benchmark CSTR problem there are three different product types that differ in purity, hence the summation terms. The first term is the revenue based on the amount of product made. The second term describes the cost of product storage. The third term assures that only one product is made in a given time slot. The fourth term is the cost of raw materials for the process.

$$J = \frac{1}{T_m}\left[\sum_{i=1}^{3} \pi_i \omega_i - \sum_{i=1}^{3} c_{storage,i} \omega_i \sum_{s=1}^{3} z_{i,s}(T_m - t_s^f) - q c_{rm} T_m\right] \quad (3)$$

There are two key variables for integrating our CSTR and scheduling models. The scheduling model provides the production schedule which is based off demand. The production schedule tells the reactor how long to produce a specific product and when to change to another. The reactor model will provide transition time which is the time it takes to transition from producing one product to another. A shorter transition time will have greater power to follow an optimized schedule. The two models will work together to maximize profit by determining a production schedule that will best meet demand. Figure 1 shows how the two problems can be integrated.

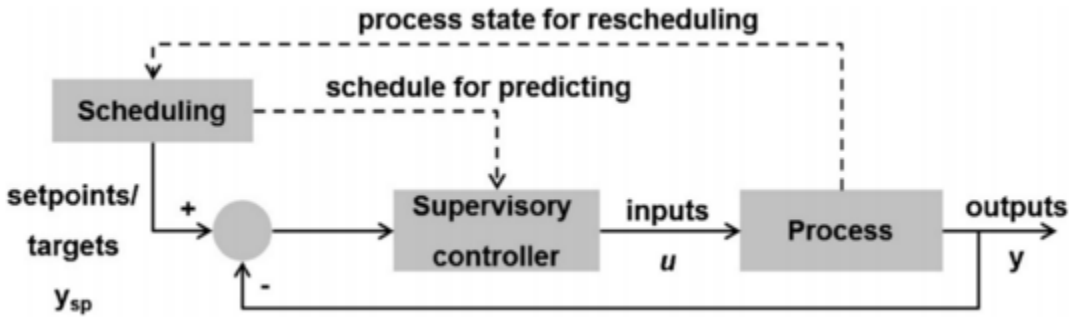

Figure 1: Integration between scheduling and process control

Table 2 shows the model parameters that were used to simulate the CSTR. These are different from the benchmark problem in order to be more realistic. The source code for this problem is also included in the appendix.

Table 2: CSTR parameters

Parameter	Value
T_c (Cooling Temperature)	280 K (Initial Condition)
Q (Cooling water flow)	100 m^3/sec
V (Volume of CSTR)	100 m^3
Rho (Density of Mixture)	1000 kg/m^3
CP (Heat Capacity of Mixture)	0.239 J/kg-K
mdelH (Heat of Reaction)	5x104 J/mol
Ea/R	8750
k0	7.2x104
UA (Heat Transfer)	5x104 W/K
Feed Concentration	1 mol/m^3
Feed Temperature	350 K

Nonlinear model predictive control was used in building the CSTR model. The CSTR model has been tuned to handle the different set point changes. For the low grade product (80% purity) the temperature set point is 371 K. For the mid-grade product (85% purity) the temperature set point is 377 K. For the premium product (90% purity), the temperature set point is 382 K. The figure below shows the controllability of the reactor. The temperature is gradually stepped up until the desired concentration is produced. Then the temperature set points are changed to produce the various purities. Adequate control over close to a 24 hour period is achieved. The controller can handle any set point change to match production specifications without causing a runaway reaction.

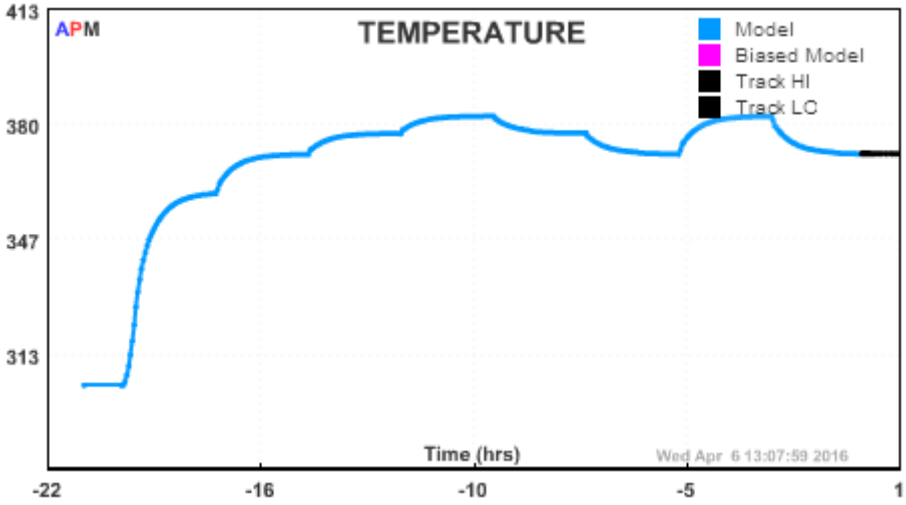

Figure 2: CSTR temperature profile with set point changes

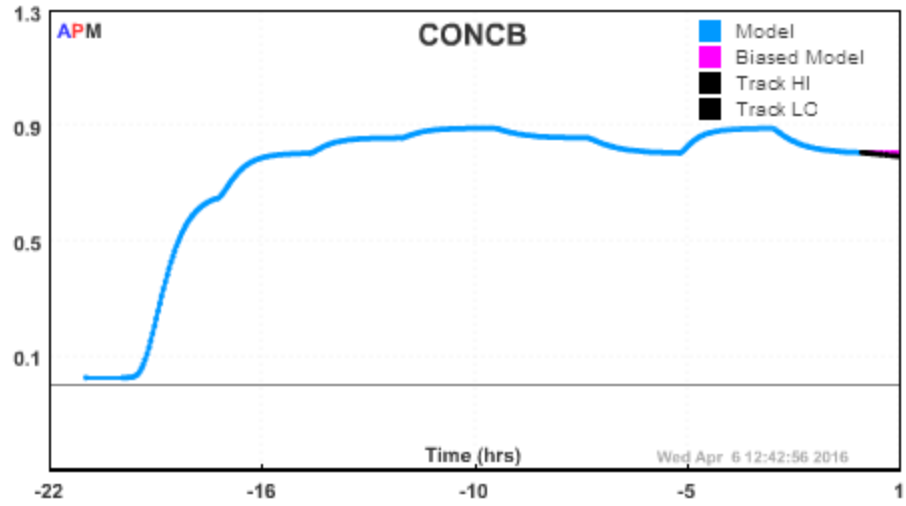

Figure 3: CSTR outlet concentration profile with set point changes

Sensitivity Analysis on CSTR Model

Figure 4 below shows a sensitivity analysis for the CSTR model. It can be seen that the manipulated variable, the cooling jacket temperature (Tc), has no effect on the objective function. It does have a minor positive effect on the concentration of the desired product (b) in the reactor. It has a major positive contribution to the reactor temperature which makes sense due to the fact that the cooling jacket temperature directly impacts the reactor temperature. The cooling jacket temperature also secondarily impacts the product concentration by manipulating the reactor temperature. It is important to note that the concentration of reactant (a) in the reactor is inversely affected by cooling temperature with the same magnitude as the concentration of product (b).

APMonitor		Objective	CV(1)	CV(2)	SV(1)
	Sensitivities	Function	p(7).n(3).concb	p(7).n(3).temperature	p(7).n(3).concentration
MV(1)	p(1).n(1).cooling_temp	0.00	9.612E-03	1.09174	-9.612E-03

Figure 4: AP Monitor sensitivity analysis

Simulation with Scheduler

The purpose of the scheduler is to take a product discharge schedule and turn it into an optimal schedule for CSTR operation. The scheduler is optimized using a model that adjusts the times that each product is produced in the CSTR based on a schedule of trucks coming to pick up a particular product. These trucks come from different companies, some that have big contracts and others that are just "spot sales". The amount and spec of the product that a particular truck demands depends on the needs of the company making the purchase, and is ordered at least one week in advance. For this project, it is assumed that the time of arrival of every truck, and the amount and spec that they are loading is known 7 days before the arrival of the truck. Hence, the schedule optimizer plans the operation of the CSTR for 7 days, hour by hour. This trucking schedule is inputted into a .csv file with 168 time points that represent the hours for 7 weeks. A 24 hour period of this .csv file is sampled as table 1.

Table 3: 24hr sample of the trucking schedule from the .csv file. Demand values are totals with units of m^3. [1] is 80% purity, [2] is 85% purity, and [3] is 90% purity.

Time (hrs)	Demand[1]	Demand[2]	Demand[3]
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	5	0	0
7	5	0	0
8	5	0	0
9	5	0	0
10	5	0	0
11	5	0	0
12	5	0	0
13	5	0	0
14	5	0	0
15	5	0	0
16	5	0	0
17	5	0	0
18	5	0	0
19	5	0	0
20	5	14	0
21	5	14	0
22	5	14	0
23	5	14	0
24	5	14	5

For the simulator, this .csv file tells the model when each product is to be produced.

The objective function for the scheduler maximizes a profit equation. The foundation of this equation comes from “Integrated production scheduling and process control: A systematic review”, but has been modified slightly to fit the present optimization problem. The following equations outline the objective of our model which is written for and solved by APMonitor. The equations are kept in APMonitor language format.

$$\text{Profit} = w[1:3] * \text{price}[1:3] - A[1:3] * \text{cstorage}[1:3] - q * CR * \text{time} \quad (4)$$

Where time has units of hours, q is the flow rate through the CSTR set at a constant 1 m^3 per hour, CR is the cost of raw materials set at \$20/m^3 reactant, cstorage is the cost to store the product which changes for each spec, A is the amount of each product stored, and w is the amount of each product produced. The storage, A, was calculated simply as the amount produced minus the amount sold, as seen in equation 2.

$$A[1:3] = w[1:3] - \text{sale}[1:3] \quad (5)$$

Equation 3 shows how the amount produced, w, was calculated in the model.

$$\$w[1:3] = \text{intb}[1:3] * q \quad (6)$$

Where \$ is the derivative with respect to time, and intb is an integer value that represents the three product specs. Figure 5 below is a sample simulation of a simple product schedule.

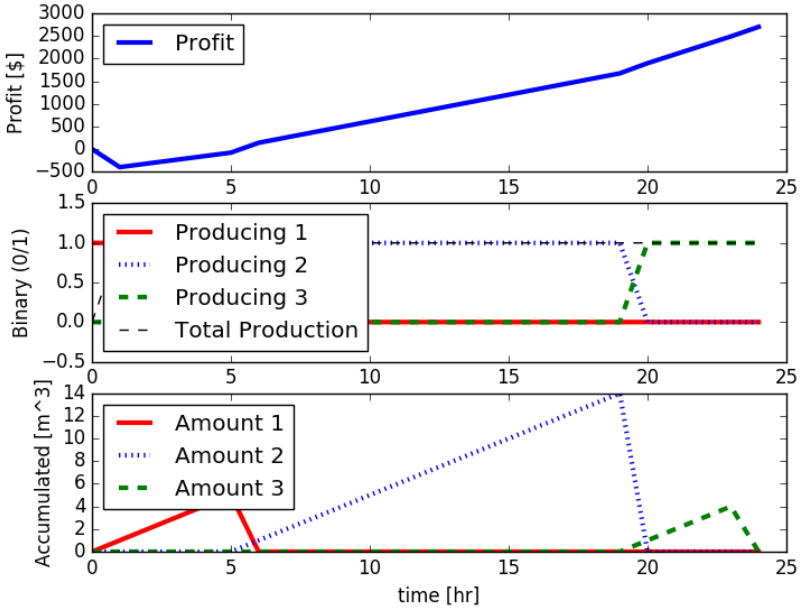

Figure 5: Scheduling simulation

The products can only be made one at a time, as shown in the second sub plot. The code takes a binary input where a 'one' means that product is being produced while a 'zero' means it is not being produced. The three binary values have to be equal to one at all times in order to ensure that only one product is being produced at a time. For this example, there was a demand of 5 for product 1 that needed to be met at hour 5. The third plot shows that the accumulated product was 5 at hour 5 then it went to 0 by hour 6. This means the product was sold and taken out of storage so the accumulated product goes to 0. Product 2 has a demand of 14 that needed to be met by hour 20. At hour 20, all of the product is sold and the accumulation goes to 0.

Sensitivity Analysis on Scheduling Model

When a sensitivity analysis is performed on the scheduling model, APMonitor shows that the manipulated variables (intb1, intb2, and intb3) have no effect on the objective function. The demand schedule controls when each of the products is made. As can be seen in the third subplot in Figure 5, each of the different product purities is made in the time leading up to when each is demanded to be shipped out. The optimizer is more dynamic and can choose when to produce a certain product in order to maximize profitability.

Optimization with Scheduler

The optimizer defines intb[1:3] from Equation 6 as manipulated variables and sets the objective function in the model to maximize Equation 4. The optimizer sees the demand schedule in the .csv file and then determines how much total of each product spec needs to be made by each hour, and outputs the CSTR's optimal (most profitable) schedule for spec production and storage. There are no controlled variables, only the objective function to maximize profit at the last time point (72nd hour).

Below is an example of an optimized 3-day schedule. The first figure shows an optimized solution without accounting for transition time. The top plot is the profit which totals to $9500 after three days of operating. The middle plot shows the 'intb' values which indicate which product is being produced. The bottom plot shows the storage levels of each product which is a cost in the profit equation. It can be seen that this solution changes between products a lot. This isn't a very precise model because of the lack of transition time in the model. To account for transition time a dcost constraint was used to discourage changing between products grades frequently. In the scheduling model transition times were all assumed as the same and do not account for a change from the low-grade to the premium product, which would have a greater transition time. Figure 6 below shows the scheduling solution with the dcost function to account for transition time.

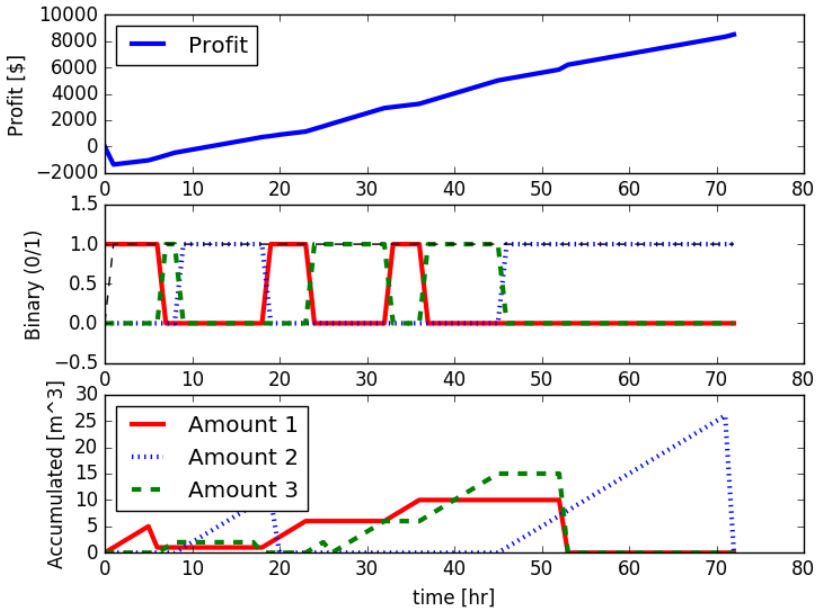

Figure 6: Optimization results for a 72 hr schedule without dcost for transition times.

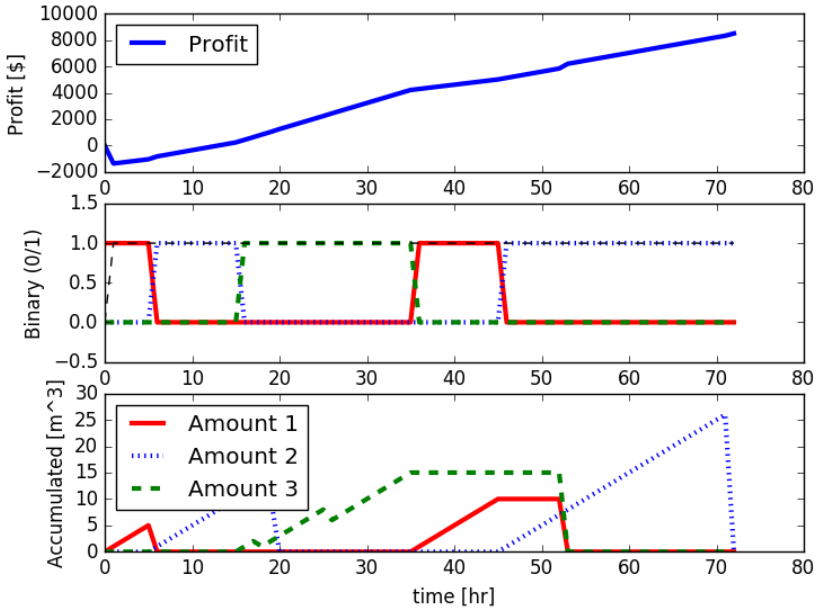

Figure 7: Optimization results for a 72 hr schedule with dcost for transition times.

As can be seen by comparing Figures 6 and 7, the dcost functions on the manipulated variables intb[1:3] reduce the number of transitions from seven to four. The profit was also reduced to $8500. This is because the cost of transition times is now somewhat accounted for. Even though this isn't an exact method for implementing transition times found from the CSTR model it is believed to be adequate for the scope of this project.

The shipment schedule must be set up in a way so the output of product does not exceed the maximum amount of product that can be produced. The maximum is based on the fixed flow rate of $1m^3/hr$. If the demand on the .csv file exceeds this maximum, the optimizer will be infeasible. To avoid this issue, an excel spreadsheet was created to aid in making the schedule so as to not overload the production. This spreadsheet basically takes the .csv file setup from Table 3 and adds two columns: "Total Product Sold" and "Maximum Product Available". The maximum product available multiplies the constant flow rate by the time and subtracts the total product sold. In making the schedule, it is imperative to keep the 'maximum product available' column above zero. To see this spreadsheet, refer to the Appendix.

Future Work

The CSTR and scheduling models are separate, but they are compatible. The scheduling model optimizes the production of the three product purities based on the demand schedule that it is given. The optimized production schedule is then used to specify at which temperatures the CSTR should operate at certain times. The temperature set points are manually changed in the CSTR model and the controller is able to manipulate the cooling water temperatures to control the reactor temperature and corresponding product compositions. Further improvements can be made to incorporate the scheduling and reactor models into one model file so that the scheduler and the controller can communicate with each other. This removes the human "middle man" that has to change the set points in the CSTR model. If these models were able to communicate, significant time would be saved and process optimization would be facilitated even further.

In its current state, the scheduling model uses a dcost constraint to represent (simulate) the transition time between the production of the different product purities. While this gives a decent representation of how the process behaves, it would be more accurate if true transition time were used in the scheduling model. Step tests performed with the CSTR mode have yielded transition times of 2 hours for changes between 80 and 85 percent purity and 85 to 90 percent purity. For changes between 80 and 90 percent purity, transition times of 3 hours have been observed. Knowing these numbers is only the first step. It is more difficult to implement them into the scheduling model in an if-then set up.

Appendix

CSTR Model File

```
Model
  Parameters
    ! PID set point
    sp = 320

    ! Manipulated Variables
    Cooling_Temp = 280    ! Temperature of cooling jacket (K)

    ! Parameters
    Flow = 100      ! Volumetric Flowrate (m^3/sec)
    V = 100      ! Volume of CSTR (m^3)
    rho = 1000   ! Density of A-B Mixture (kg/m^3)
    Cp = .239    ! Heat capacity of A-B Mixture (J/kg-K)
    mdelH = 5e4  ! Heat of reaction for A->B (J/mol)
                 ! E - Activation energy in the
                 !  Arrhenius Equation (J/mol)
                 ! R - Universal Gas Constant
                 !  = 8.31451 J/mol-K
                 ! EoverR = E/R
    EoverR = 8750
    k0 = 7.2e10 ! Pre-exponential factor (1/sec)
                 ! U - Overall Heat Transfer
                 !  Coefficient (W/m^2-K)
                 ! A - Area - this value is specific
                 !  for the U calculation (m^2)
                 ! UA = U * A
    UA = 5e4
    Feed_Conc = 1      ! Feed Concentration (mol/m^3)
    Feed_Temp = 350     ! Feed Temperature (K)
  End Parameters

  Variables
    ! Differential States
    Concentration = 0.9, >0, <=1     ! Concentration of A in CSTR (mol/m^3)
    ConcB = 0, >=0, <=1              ! Concentration of B in CSTR (mol/m^3)
    Temperature  = 360 , >0, <=2000   ! Temperature in CSTR (K)
  End Variables

  Intermediates
    k = k0*exp(-EoverR/Temperature)
    rate = k * Concentration >= 0
  End Intermediates
```

CSTR Python File

```
# Import
from apm import *

# Select server
s = 'http://byu.apmonitor.com'

###################################
# set up nonlinear MPC
###################################
c = 'nmpc'
# Clear previous application
apm(s,c,'clear all')

# load model variables and equations
apm_load(s,c,'nmpc.apm')

# load data
csv_load(s,c,'nmpc.csv')

#  APM Variable Classification
apm_info(s,c,'MV','Cooling_Temp')
apm_info
apm_info(s,c,'SV','Concentration')
apm_info(s,c,'CV','Temperature')
apm_info(s,c,'CV','ConcB')

# Options
apm_option(s,c,'nlc.nodes',3)
apm_option(s,c,'nlc.hist_hor',200)
apm_option(s,c,'nlc.web_plot_freq',3)
apm_option(s,c,'nlc.mv_type',0)
apm_option(s,c,'nlc.max_iter',200)
apm_option(s,c,'nlc.hist_hor',1000)
# Bounds
apm_option(s,c,'Cooling_Temp.lower',250)
apm_option(s,c,'Cooling_Temp.upper',350)
# Turn on parameters to control
apm_option(s,c,'Cooling_Temp.status',1)
apm_option(s,c,'Cooling_Temp.dmax',10)
apm_option(s,c,'Cooling_Temp.fstatus',0)
```

```
apm_option(s,c,'Temperature.fstatus',0)
apm_option(s,c,'Temperature.tau',0.5)
apm_option(s,c,'Temperature.tr_init',2)
apm_option(s,c,'Temperature.wsphi',10.0)
apm_option(s,c,'Temperature.wsplo',10.0)

apm_option(s,c,'Concentration.status',1)
apm_option(s,c,'Concentration.fstatus',0)


apm_option(s,c,'ConcB.status',1)
apm_option(s,c,'ConcB.fstatus',0)


# initialize with steady state simulation
apm_option(s,c,'nlc.imode',1)
solver_output = apm(s,c,'solve')
# switch back to dynamic optimization
apm_option(s,c,'nlc.imode',6)

# initialize values
time = 0.0
dt = 0.05
Tsp = 360
delta = 0.1

cycles = 401
rows = cycles
cols = 3
nrows = cycles
ncols = 3

import numpy as np
#g = np.empty((rows,cols))
#for j in range(nrows):
    #for k in range(ncols):
        #s[j][k]= np.nan

for isim in range(cycles):
```

```
    # Print data for import into Excel
    print('{0:2f} {1:2f} {2:2f} {3:2f}'.format(time,Tc,T,Ca))

    # Increment time
    time = time + dt

    if (isim==1):
        # Open Web Viewers
        url = apm_web(s,c)

import matplotlib.pyplot as plt

#plt.figure(1)

#plt.subplot(2,1,1)
#plt.plot(s[:,0],s[:,1],'b-')
#plt.xlabel('Time')
#plt.ylabel('Concentration B mol/L')

#plt.subplot(2,1,2)
#plt.plot(s[:,0],s[:,2],'r-')
#plt.xlabel('Time')
#plt.ylabel('Temperature (K)')

#plt.show()
```

Schedule Model File

```
File apopt.opt
  minlp_gap_tol 0
End File


Constants
 CR = 20 !$/m^3 Cost of raw materials
 q = 1 !m^3/h Production Rate
 Cao = 1 !mol/L
End Constants

Parameters
 p[1]=1
 p[2]=2
 p[3]=3

 !demand[1]=5 !m^3/day needs to be 10
 !demand[2]=14 !m^3/day needs to be 30
 !demand[3]=5 !m^3/day needs to be 10

 price[1]=100 !$/m^3
 price[2]=120 !$/m^3
 price[3]=200 !$/m^3

 cstorage[1] = 20 !$/m^3/hr
 cstorage[2] = 2 !$/m^3/hr
 cstorage[3] = 3 !$/m^3/hr

!Tf[1:3] = 0  !start time of the slot s where product i is made
!Ts[1:3] = 0  !finish time of the slot s where product i is made
!transtime = .5      !transition time
 final[1:3] = 0
 intb[2:3] = 0 <=1, >=0
 intb[1] = 1 <=1, >=0
 fin
 Tm = 72 !cycle time 24 hr
End Parameters

Variables
 Profit = 0 !Profit
 J[1:3] = 0 !individual product profit
 w[1:3] = 0 !Quantity of product i manufactured, m^3/hr
 c = 0
 A[1:3] = 0 , >= 0
End Variables

Equations
 Maximize Profit * fin
 A[1:3] = w[1:3] - final[1:3] ! divide demand by 24 to get into hours
 c = intb[1] + intb[2] + intb[3]
 J[1:3] = w[1:3] * price[1:3] - A[1:3] * cstorage[1:3]
 Profit = J[1] + J[2] + J[3] - q * CR * Tm
 intb[1] + intb[2] + intb[3] <= 1
 $w[1] = intb[1] * q
 $w[2] = intb[2] * q
 $w[3] = intb[3] * q
 !fin * (w[1:3] - demand[1:3]) >= 0
End Equations
```

Schedule Python File

```
# Import
from apm import *

# Select server
s = 'http://byu.apmonitor.com'

#################################
# set up schedule simulator
#################################
a = 'projschedule'
# Clear previous application
apm(s,a,'clear all')
# Load model file
apm_load(s,a,'scheduling.apm')
# Load time points for future predictions
csv_load(s,a,'schedule2.csv')

apm_option(s,a,'nlc.imode',6)
for i in range(3):
    apm_info(s,a,'MV', 'intb[' + str(i+1) + ']')
    apm_option(s,a,'intb[' + str(i+1) + '].status',1)

apm_option(s,a,'intb[1].dcost',100)
apm_option(s,a,'intb[2].dcost',100)
apm_option(s,a,'intb[3].dcost',100)
apm_option(s,a,'nlc.sensitivity',1)
# Solve
output = apm(s,a,'solve');
#print(output)
apm_web(s,a)

y = apm_sol(s,a)

import matplotlib.pyplot as plt
plt.figure(1)
plt.subplot(3,1,1)
plt.plot(y['time'], y['profit'],'b-',linewidth=3)
plt.legend(['Profit'],loc=2)
plt.xlabel('time [hr]')
plt.ylabel('Profit [$]')

ax=plt.subplot(3,1,2)
plt.plot(y['time'], y['intb[1]'],'r-',linewidth=3)
plt.plot(y['time'], y['intb[2]'],'b:',linewidth=3)
plt.plot(y['time'], y['intb[3]'],'g--',linewidth=3)
plt.plot(y['time'], y['c'],'k--',linewidth=1)
#plt.legend(['Producing 1','Producing 2','Producing 3','Total Production'],loc=2)
plt.xlabel('time [hr]')
plt.ylabel('Binary (0/1)')
ax.set_ylim([-0.5,1.5])

plt.subplot(3,1,3)
plt.plot(y['time'], y['a[1]'],'r-',linewidth=3)
plt.plot(y['time'], y['a[2]'],'b:',linewidth=3)
plt.plot(y['time'], y['a[3]'],'g--',linewidth=3)
plt.legend(['Amount 1','Amount 2','Amount 3'],loc=2)
plt.xlabel('time [hr]')
plt.ylabel('Accumulated [m^3]')
plt.show()
```

Schedule Spreadsheet

time (hrs)	sale[1]	sale[2]	sale[3]	fin	total sold (m^3)	max available (m^3)
0	0	0	0	0	0	0
1	0	0	0	0	0	1
2	0	0	0	0	0	2
3	0	0	0	0	0	3
4	0	0	0	0	0	4
5	0	0	0	0	0	5
6	0	0	0	0	0	6
7	5	0	0	0	5	0
8	5	0	0	0	5	3
9	5	0	0	0	5	4
10	5	0	0	0	5	5
11	5	0	0	0	5	6
12	5	0	0	0	5	7
13	5	0	0	0	5	8
14	5	0	0	0	5	9
15	5	0	0	0	5	10
16	5	0	0	0	5	11
17	5	0	0	0	5	12
18	5	0	0	0	5	13
19	5	0	0	0	5	14
20	5	12	0	0	17	1
21	5	12	0	0	17	4
22	5	12	0	0	17	5
23	5	12	0	0	17	6
24	5	12	5	0	22	0
25	5	12	5	0	22	3
26	5	12	5	0	22	4
27	5	12	5	0	22	5
28	5	12	5	0	22	6
29	5	12	5	0	22	7
30	5	12	5	0	22	8
31	5	12	5	0	22	9
32	5	12	5	0	22	10
33	5	12	5	0	22	11
34	5	12	5	0	22	12
35	5	12	5	0	22	13
36	5	12	5	0	22	14
37	5	12	5	0	22	15
38	5	12	5	0	22	16
39	5	12	5	0	22	17

40	5	12	5	0	22	18
41	5	12	5	0	22	19
42	5	12	5	0	22	20
43	5	12	5	0	22	21
44	5	12	5	0	22	22
45	5	12	5	0	22	23
46	5	12	5	0	22	24
47	5	12	5	0	22	25
48	5	12	5	0	22	26
49	5	12	5	0	22	27
50	5	12	5	0	22	28
51	5	12	5	0	22	29
52	5	12	5	0	22	30
53	5	12	5	0	22	31
54	5	12	5	0	22	32
55	5	12	5	0	22	33
56	5	12	5	0	22	34
57	5	12	5	0	22	35
58	5	12	5	0	22	36
59	5	12	5	0	22	37
60	5	12	5	0	22	38
61	5	12	5	0	22	39
62	20	12	20	0	52	8
63	20	12	20	0	52	11
64	20	12	20	0	52	12
65	20	12	20	0	52	13
66	20	12	20	0	52	14
67	20	12	20	0	52	15
68	20	12	20	0	52	16
69	20	12	20	0	52	17
70	30	12	20	0	62	6
71	20	12	20	0	52	17
72	20	12	20	0	52	20