Process model formulation and solution, 3E4

Computer software tutorial - Tutorial 2

Kevin Dunn, dunnkg@mcmaster.ca **September 2010**

Tutorial objectives

- Some questions to help you feel comfortable deriving model equations for actual chemical engineering systems.
- Brute force solving of equation systems. The rest of the course will focus on better ways to solve these equations.
- Interpreting source code written on paper.

Recap of tutorial rules

- Tutorials can be done in groups of two - please take advantage of this to learn with each other.
- Tutorials must be handed in at the start of class on Wednesday. No electronic submissions - thanks!

Question 1 [2]

Note: Parts 1 and 2 of this question were from the 2006 final exam (slightly modified). It was worth 10% of the 3 hour exam

Consider a mixing tank with fluid volume V where fluids A and B with densities ρ_A and ρ_b, specific heat capacities $C_{p,A}$, $C_{p,B}$ and temperatures T_A and T_B are the inlet streams at volumetric flow rates F_A and F_B. The outlet stream is at a flow rate $F_A + F_B$. The specific heat capacity and density of the outlet streams is given by $C_p = (C_{p,A}F_A + C_{p,B}F_B)/F$ and $\rho = (\rho_A F_A + \rho_B F_B)/F$. The fluid also looses heat from the tank at a rate $q = k_T(T - T_{\text{wall}})$ where T_{wall} is the constant tank wall temperature, k_T is a constant and T denotes the current fluid temperature.

1. Using 3-step modelling approach shown in class, derive a dynamical balance describing the time-dependent exit stream temperature.
2. Can the steady state exit stream temperature be higher than both T_A and T_B? Explain.
3. Calculate, *by-hand*, the steady-state exit temperature, using that
   - V = 10 m^3
   - ρ_A = 1200 kg/m^3 and ρ_b = 950 kg/m^3
   - $C_{p,A}$ = 2440 J/(kg.K) and $C_{p,B}$ = 3950 J/(kg.K)
   - T_A = 320 K and T_B = 350 K and T_{wall} = 300K
   - $F_A = F_B$ = 0.05 m^3/s
   - k_T = 200 W/(m^2.K) $\times$ 24 m^2 = 4800 W/K

Question 2 [2]

Note: You don't need to write any code for this question.

Consider the reaction

$$P_2I_4 + n\,P_4 + p\,H_2O \longrightarrow 4\,PH_4I + q\,H_3PO_4$$

where n, p and q denote the stoichiometric coefficients for P_4, H_2O and H_3PO_4 respectively.

1. Derive the equations necessary to solve for n, p, and q by equating atoms of P, H, and O on the reactant and product sides.
2. In the next section of the course we will use Guass Elimination to solve these equations. For now though, let's describe a brute force approach. First, complete the two lines of this MATLAB function:

```
function total_error = equation_error( n, p, q )

% Given the values of n, p, and q, calculate the error of each balance equation.
% Returns the sum of squares of the errors.

error_1 = _________________  % from the P-balance
error_2 = 2*p - 3*q - 16;   % from the H-balance
error_3 = _________________  % from the O-balance

total_error = (error_1)^2 + (error_2)^2 + (error_3)^2;

end  % end of function
```

   or complete this Python function:

```
def equation_error(n, p, q ):
    """
    Given the values of n, p, and q, calculate the error of each of the 3 equations.
    Returns the sum of squares of the errors.
    """
    error_1 = _________________
    error_2 = 2*p - 3*q - 16
    error_3 = _________________

    return error_1**2 + error_2**2 + error_3**2
```

3. Since we known that n, p, and q must be positive, we can construct a set of 3 nested for-loops, as shown below in MATLAB and Python. Describe in plain English what the code does.

   In MATLAB:

```
smallest = 0.0;
largest = 14.9;
step_size = 0.1;
vector = smallest : step_size : largest;

% How many elements in each vector?
num = length(vector);

errors = zeros(num, num, num);

index_n = 0;
index_p = 0;
```

```
index_q = 0;
for n = smallest : step_size : largest
    index_n = index_n + 1;
    for p = smallest : step_size : largest
        index_p = index_p + 1;
        for q = smallest : step_size : largest
            index_q = index_q + 1;

            % Calculate the error at this value of n, p and q:
            errors(index_n, index_p, index_q) = equation_error(n, p, q);

        end
        index_q = 0;
    end
    index_p = 0;
end
[min_error, min_index] = min(errors(:))
[index_n, index_p, index_q] = ind2sub([num, num, num], min_index);
disp(['Solution at ', num2str([vector(index_n), vector(index_p), vector(index_q)])])
```

In Python:

```
import numpy as np

smallest = 0.0
largest = 15.0
step_size = 0.1
vector = np.arange(smallest, largest, step_size)

# How many elements in each vector?
num = len(vector)

errors = np.zeros( (num, num, num) )

for index_n, n in enumerate(vector):
    for index_p, p in enumerate(vector):
        for index_q, q in enumerate(vector):

            # Calculate the error at this value of n, p and q:
            errors[index_n, index_p, index_q] = equation_error(n, p, q)

# Which combination had the smallest error?
min_index = np.argmin(errors)
index_n, index_p, index_q = np.unravel_index(min_index, (num, num, num))
n, p, q = vector[index_n], vector[index_p], vector[index_q]
print(n, p, q)
```

4. How many times will the function `equation_error` be called?
5. What will this function output be if $(n, p, q) = (1.0, 9.2, 2.5)$?

Bonus question [0.5]

Using the code given in question 2, report what the `min_index` variable is and what are the values of n, p, and q which give minimum error to the set of equations. How long did it take to find the solution of this simple linear equation system?
