Process System Engineering #3

#03150796 Amane Suzuki

October 20, 2015

1 Abstract

The purpose of this problem set is to estimate power consumption considering that stirring energy go into reactors and produce the heat.

2 Algorithm

There is a major change in `reactor()` function. The main structure is,

```
def reactor(gamma_in, gamma_out, prev_energy)
  calc_heat_transfer_rate   #=> h
  calc_dimensionless_number #=> Re, Pr, Nu
  calc_power_consumption    #=> new_energy

  if (new_energy - prev_energy).abs < accuracy
    return results
  else
    return reactor(gamma_in, gamma_out, new_energy)
  end
end
```

In `reactor()` function, we use Newton's method (Figure.1).

Given that stirring energy does not go into reactor, we can calculate power consumption of each reactor (based on the method which is stated in previous report).

Power consumption, however, actually go into reactor and produce the heat.

Therefore, while accuracy is inadequate, `reactor()` pass power consumption to `reactor()` repeatedly and modify it.

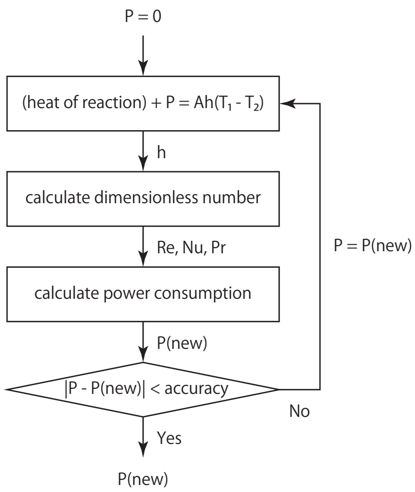

Figure 1: Routine in reactor() function

3 Result

(1) $N = 1, \gamma_0 = 0.05$

Use listing 1,

```
$ ruby plant.rb
input n[-], gamma_0[wt%], T2[K]
1
0.05
258
Conditions:
(N, gamma_0, T2) = (1, 0.05, 258)
Reactor Size:
V = 514.706 [m3]
D = 7.959 [m]
H = 10.346 [m]
Results:
#1
Re = 7299.855
n = 0.334 [rps]
P = 38267.204 [W]
Total:
Ptot = 38267.204 [W]
```

volume	514.706 m^3
diameter	7.959 m
height	10.346 m
total power consumption	38.267 kW

Table 1: Result in $(N, \gamma_0, T_2) = (1, 0.05, 258)$

(2) $N = 3, \gamma_0 = 0.04$

Use listing 1,

```
$ ruby plant.rb
input n[-], gamma_0[wt%], T2[K]
3
0.04
258
Conditions:
(N, gamma_0, T2) = (3, 0.04, 258)
Reactor Size:
V = 153.215 [m3]
D = 5.314 [m]
H = 6.908 [m]
Results:
#1
Re = 11429.325
n = 0.121 [rps]
P = 213.882 [W]
#2
Re = 3632.928
n = 0.153 [rps]
P = 594.829 [W]
#3
Re = 1636.309
n = 0.136 [rps]
P = 522.565 [W]
Total:
Ptot = 1331.275 [W]
```

volume	$151.215\,\mathrm{m}^3$
diameter	$5.314\,\mathrm{m}$
height	$6.908\,\mathrm{m}$
total power consumption	$1.331\,\mathrm{kW}$

Table 2: Result in $(N, \gamma_0, T_2) = (3, 0.04, 258)$

(3)show table

```
$ ruby plant-advanced.rb
593.502 126.602 54.813 31.533 20.908
2729.104 578.714 250.365 143.986 95.456
7790.098 1628.786 703.4 404.237 267.889
18586.917 3770.145 1622.235 930.917 616.453
41466.439 7879.11 3366.358 1926.365 1273.802
```

	N				
γ_0	1	2	3	4	5
0.01	593.502	126.602	54.813	31.533	20.908
0.02	2729.104	578.714	250.365	143.986	95.456
0.03	7790.098	1628.786	703.4	404.237	267.889
0.04	18586.917	3770.145	1622.235	930.917	616.453
0.05	41466.439	7879.11	3366.358	1926.365	1273.802

Table 3: Power Consumption P[kW] in each condition

4 Source Program

Listing 1: plant.rb

```
include Math

ACCURACY = 0.001 # when stop calculating

GP = (63.0*1000/24) # product flow rate [kg h-1]
ML = 50.0 # polymer length
TAU = 5.0 # residence time when n=1
RHO = 850.0 # density [kg m-3]
ALPHA = 1.3 # height/diameter of reactor
HP = (72.8*1000) # heat of polymerization [J mol-1]
CV = 0.6 # conversion
T1 = (273+50) # temperature in reactor [K]
TC = 0.128 # thermal conductivity [W m-1 K-1]
CP = (1.68*1000) # specific heat of toluene [J kg-1 K-1]
M = 54.0 # molecular weight of butadiene [g mol-1]

class Plant
  def initialize(n, feed, coolant)
    @n = n # number of reactors [-]
    @feed = feed # feed [wt%]
    @rate = GP/(RHO*CV*feed) # feed speed [m3 h-1]
    @k = (1.0/(1-CV)-1)/TAU # reaction constant [h-1]
    @tau = (1.0/@k)*((1-CV)**(-1.0/@n)-1) # residence time when n
        !=1
    @volume = @rate*@tau # [m3]
    @diameter = (@volume/(ALPHA*2*PI))**(1/3.0)*2 # [m]
    @height = @diameter*ALPHA # [m]
    @surface = @diameter*PI*@height # [m2]
    @coolant = coolant # T2 [K]
    @data = Array.new(n) # data of each reactor
  end

  def show()
    # conditions
    puts "Conditions:"
    puts "(N, gamma_0, T2) = (#{@n}, #{@feed}, #{@coolant})"

    # reactor size
    puts "Reactor Size:"
    puts "V = #{@volume.round(3)} [m3]"
    puts "D = #{@diameter.round(3)} [m]"
    puts "H = #{@height.round(3)} [m]"

    # result of each reactor
    puts "Results:"
    for n in 0...@n
      puts "##{n+1}"
      puts "Re = #{@data[n][:re].round(3)}"
      puts "n = #{@data[n][:revolution].round(3)} [rps]"
      puts "P = #{@data[n][:power].round(3)} [W]"
```

```ruby
    end

    # total power consumption
    total_power = 0
    @data.each do |data|
      total_power += data[:power]
    end
    puts "Total:"
    puts "Ptot = #{total_power.round(3)} [W]"
  end

  def calc()
    prop = (1-CV)**(1.0/@n) # proportional constant [-]
    for n in 0...@n
      @data[n] = reactor(@feed*(prop**(n)),@feed*(prop**(n+1)), 0)
    end
  end

  def reactor(gamma_in, gamma_out, stir_energy)
    # heat transfer rate
    heat_of_reaction = HP*(@rate*RHO*(gamma_in-gamma_out)/3.6)/M
    heat = heat_of_reaction + stir_energy
    h = heat/@surface/(T1-@coolant)

    # viscosity [Pa s]
    viscosity = ((ML)**1.7)*((1-(gamma_out/@feed))**2.5)*exp(21.0*
        @feed)*1e-3

    # dimensionless numbers
    pr = viscosity*CP/TC
    nu = h*@diameter/TC
    re = (2*nu/pr**(1/3.0))**1.5

    # power consumption
    revolution = re*viscosity/RHO/(@diameter/2)**2
    np = 14.6*re**(-0.28)
    power = np*RHO*(revolution**3)*(@diameter/2)**5

    if (power - stir_energy).abs < ACCURACY
      return {
        re: re,
        revolution: revolution,
        power: power
      }
    else
      return reactor(gamma_in, gamma_out, power)
    end
  end
end

puts "input n[-], gamma_0[wt%], T2[K]"
n = gets.to_i
feed = gets.to_f
coolant = gets.to_i
```

```

plant = Plant.new(n, feed, coolant)
plant.calc()
plant.show()
```