# Report
# Project 1 - February 16th 2021

## ME 249 – Machine Learning Tools for Modeling Energy Transport and Conversion Processes

Lance MIRANDA
Alexis RUIZ

# Task 1

a) As expected, the data printed matches the array in the appendix.

```
ydata = [[44.1, 32.5, 0.098, 1.79, 5.5], [47.4, 33.2, 0.098, 1.79, 5.5], [49.4, 34.2, 0.098, 1.79, 5.5], [59.2, 34.8,
0.098, 1.79, 5.5], [67.8, 36.3, 0.098, 1.79, 5.5], [73.6, 37.3, 0.098, 1.79, 5.5], [76.3, 37.8, 0.098, 1.79, 5.5], [8
5.3, 39.2, 0.098, 1.79, 5.5], [96.5, 39.3, 0.098, 1.79, 5.5], [111.0, 42.3, 0.098, 1.79, 5.5], [124.0, 43.5, 0.098,
1.79, 5.5], [136.2, 45.4, 0.098, 1.79, 5.5], [143.5, 46.7, 0.098, 1.79, 5.5], [154.6, 47.9, 0.098, 1.79, 5.5], [163.
1, 48.6, 0.098, 1.79, 5.5], [172.8, 50.9, 0.098, 1.79, 5.5], [184.2, 51.7, 0.098, 1.79, 5.5], [203.7, 56.4, 0.098, 1.
79, 5.5], [36.7, 30.2, 9.8, 1.79, 5.5], [55.1, 34.1, 9.8, 1.79, 5.5], [67.5, 35.3, 9.8, 1.79, 5.5], [78.0, 37.8, 9.8,
1.79, 5.5], [92.0, 38.1, 9.8, 1.79, 5.5], [120.0, 44.1, 9.8, 1.79, 5.5], [134.3, 46.9, 9.8, 1.79, 5.5], [150.3, 48.5,
9.8, 1.79, 5.5], [167.0, 49.2, 9.8, 1.79, 5.5], [184.0, 52.7, 9.8, 1.79, 5.5], [196.5, 53.1, 9.8, 1.79, 5.5], [42.4,
29.7, 19.6, 1.79, 5.5], [48.7, 31.0, 19.6, 1.79, 5.5], [54.5, 31.2, 19.6, 1.79, 5.5], [70.8, 32.4, 19.6, 1.79, 5.5],
[73.7, 31.4, 19.6, 1.79, 5.5], [81.8, 32.5, 19.6, 1.79, 5.5], [91.9, 36.3, 19.6, 1.79, 5.5], [103.9, 36.3, 19.6, 1.7
9, 5.5], [119.1, 37.2, 19.6, 1.79, 5.5], [133.7, 38.4, 19.6, 1.79, 5.5], [139.9, 39.7, 19.6, 1.79, 5.5], [148.3, 40.
9, 19.6, 1.79, 5.5], [157.0, 41.6, 19.6, 1.79, 5.5], [169.1, 43.9, 19.6, 1.79, 5.5], [179.2, 45.0, 19.6, 1.79, 5.5],
[205.0, 47.9, 19.6, 1.79, 5.5]]

Out[1]: '>>>>> end CodeP1.1 '
```

**Figure 1** - *First output: Storage of Heat flux, superheat, gravity…*

b) Thanks to the data stored in ydata, we can plot (log-log) the heat flux vs. wall superheat for $g$ = 0.098 *and* 9.81 m/s$^2$. The code used for this plot is available in the appendix.

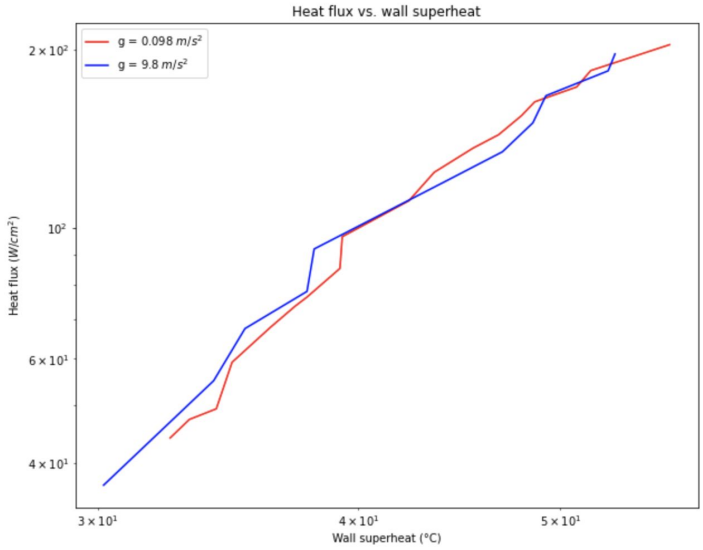


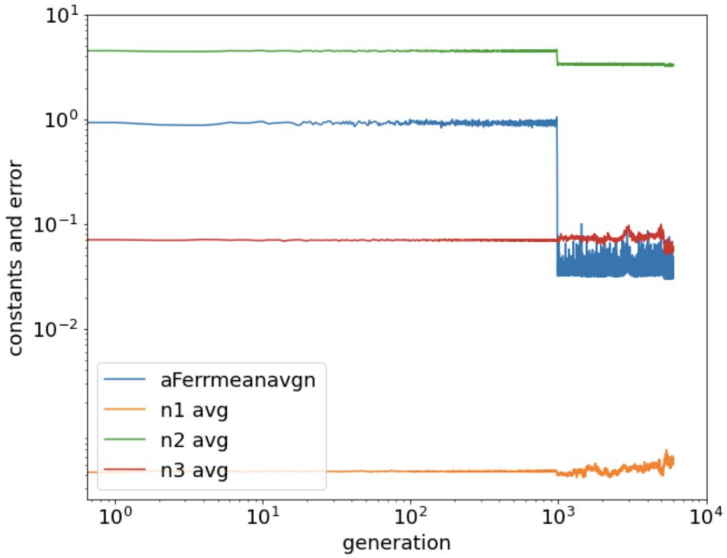

**Figure 2** - *Evolution of heat flux in function of gravity and wall superheat*

We observed that the heat flux strongly depends on the gravity and wall superheat. It varies from 40 to 200 W/cm$^2$ when the wall superheat only increases by 20 °C.

# Task 2

When running the code without modifying it, we obtain the following results (**figure 3**).
We observe a minimum error of 0.0299 which implies an acceptable fit for this 3-constant model.

```
ENDING: pop. avg n1-n3,aFerrmean: 0.0005446895455748129 3.3348413931449277 0.05679812460987663 0.047717825919730374
MINUMUM:  avg n1-n3,aFerrmeanMin: 0.000612193260978041 3.2556931263223454 0.0616613698608559 0.029953073098913582
TIME AVG:    avg n1-n3,aFerrmean: 0.0004681234558595183 5 3.376565083053762 0.07284932946961302 0.18143838275104598
```



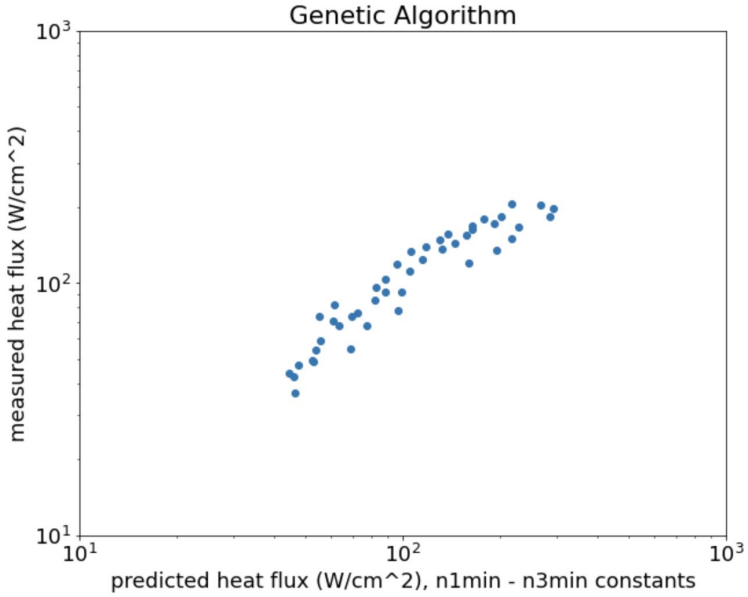




**Figure 3 -** *Output from code P1.1 & P1.2*

Now, we run the same code but we modify the initial guesses and see how the minimum error evolves (**Figure 4**).

We observe that small changes in initial guesses for the 3 constants might have a huge impact on the error (Going from 0.03 to more than 1). It is sensitive when changing all the initial guesses.
Same when strongly changing $n_2$ from 3.4 to 5, the minimum error evolves a lot and reaches 1.18 instead of the 0.03 previously.
Same when strongly changing $n_1$ from 0.0002 to 0.06. However, when slightly changing only one constant, it remains relatively robust and the minimum error does not change substantially. Example when changing $n_1$ from 0.0002 to 0.0012, the error only reaches 0.0399.
When changing $n_3$ from 0.09 to 0.14, we observe a minimum error of 0.05 instead of 0.029.

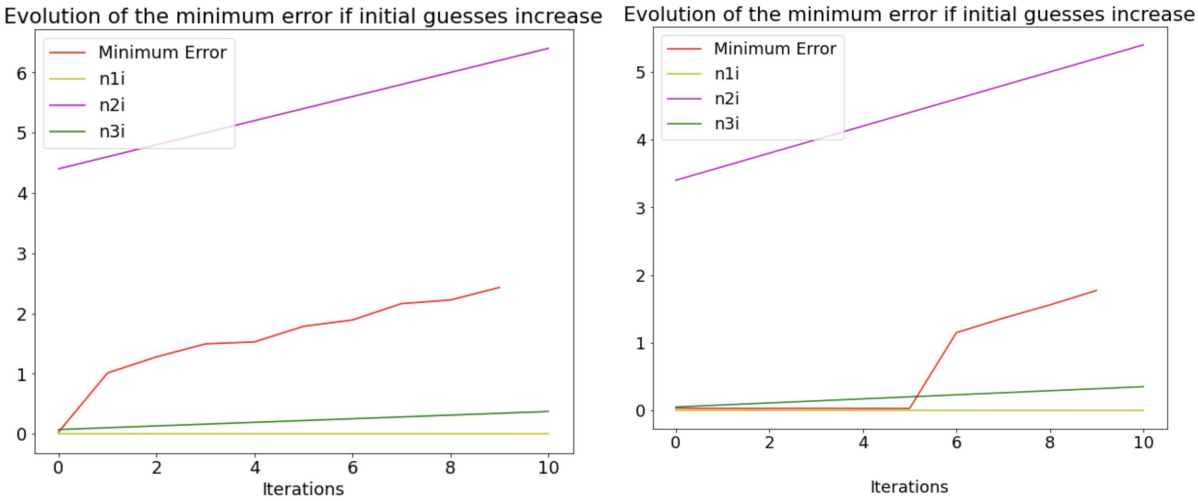



**Figure 4 -** *Evolution of the minimum error in function of initial guesses*

The figure 4 shows how the minimum error evolves quickly when changing the initial guesses - without adding new generations.

Additionally, we observed the effects of changing the different variables independently.
We notice that by increasing $n_1$, there is little effect on the overall error. However, it appears that $n_2$ and $n_3$ both have a significant effect on the error, with $n_2$ appearing to have the largest overall effect.

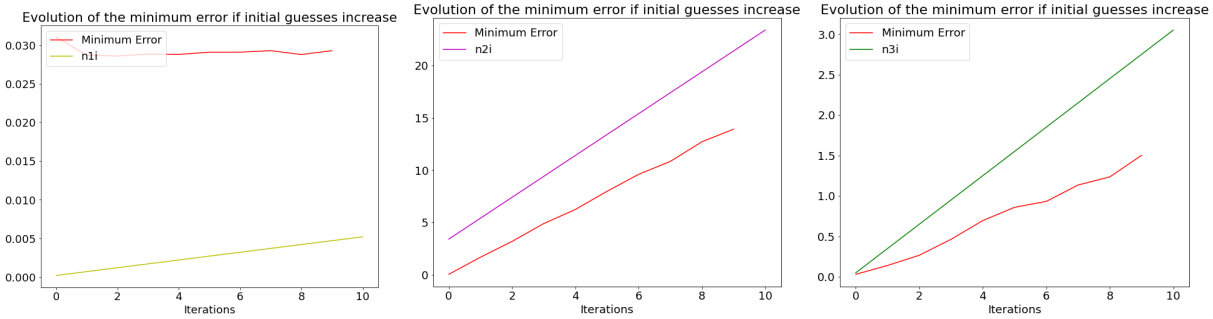



**Figure 5 -** *Effects of changing the initial guesses independently*

# Task 3

We modified the lines to take into account the new model with five constants instead of 3. We have the following changes in the code (figure 4):

```
for i in range(ND):

    Ferr[i] = n[i][0]*lydata[i][0] + math.log(n[i][1]) + n[i][2]*lydata[i][1]
    Ferr[i] = Ferr[i] + n[i][3]*math.log( ydata[i][2] + n[i][4]*9.81*ydata[i][3])
    + n[i][5]*lydata[i][4]

    aFerr[i] = abs(Ferr[i])/abs(lydata[i][0])  #- absolute fractional error
```

```
for i in range(ND):  ######CHANGE
    Ferravgn[i] = -1.*lydata[i][0] + math.log(n1avg[k]) + n2avg[k]*lydata[i][1]
    Ferravgn[i] = Ferravgn[i] + n3avg[k]*math.log( ydata[i][2] + n4avg[k]*9.81*ydata[i][3])
    + n5avg[k]*lydata[i][4]

    #aFerravgn[i] = abs(Ferr[i])/abs(lydata[i][0])
    aFerravgn[i] = abs(Ferravgn[i])/abs(lydata[i][0])
```

```
for i in range(ND):
    qpppred[i] = n1min*(ydata[i][1]**n2min) * ((ydata[i][2] + n4min*9.81*ydata[i][3])**n3min)*ydata[i][4]**n5min
    qppdata[i] = ydata[i][0]
```

**Figure 5 -** *Modifications for the 5 constant model*

After running the code with the modifications made (figure 4), we play with the initial guesses to find an acceptable error. We slightly increased the number of generations from 6000 to 6500 and took the following initial guesses:

$$n_{0i} = -1.0$$
$$n_{1i} = 0.000476$$
$$n_{2i} = 3.28$$
$$n_{3i} = 0.2849$$
$$n_{4i} = 1.174$$
$$n_{5i} = 0.287$$

We obtain the following minimum constants that minimize our error (less than 0.03: here for instance, we obtained `aFerrmeanavgnMin = 0.0294)`. The final constants from this run are outlined below in Table 1:

| | |
|---|---|
| **$n_1$** | **0.000589** |
| **$n_2$** | **2.961** |
| **$n_3$** | **0.345** |
| **$n_4$** | **1.26** |
| **$n_5$** | **0.255** |

**Table 1** - *Constants obtained after running five constant model*

The output from this model is the following figure (**figure 6**).

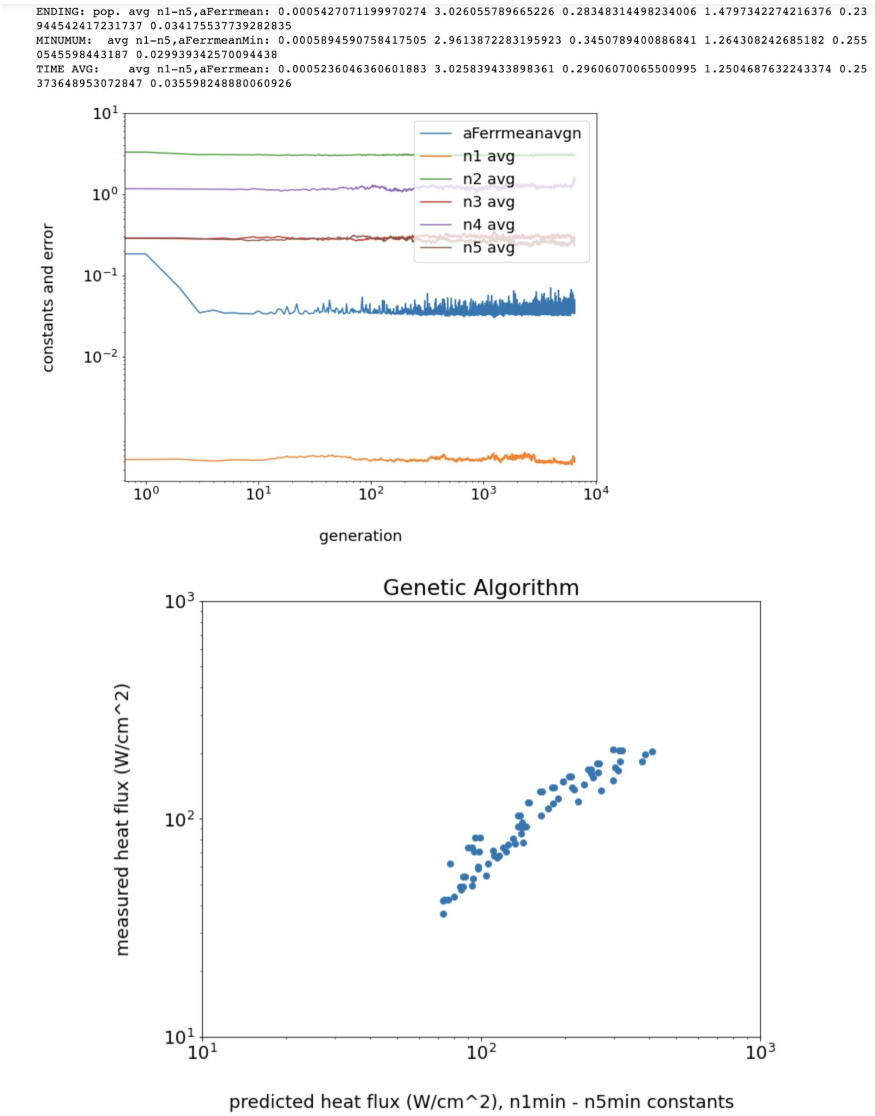



**Figure 6 -** *Output from the 5 constant model*

The model is also sensitive to the initial guesses. When modifying the constant $n_2$ from 3.28 to 4.28, the minimum error strongly changes from 0.03 to greater than 1.
However, some changes for other variables do not have the same influence on the output from the model. For instance, when changing the constant $n_4$ from 1.174 to 2.974, the error only reaches 0.0367 which is almost an acceptable error.

Thanks to the equation (5):

$$q'' = n_1 (T_w - T_{sat})^{n_2} (g + n_4 g_{en} \gamma)^{n_3} P^{n_5} \quad (5)$$

We can express $q''/ (T_w - T_{sat})^{n2}$ as a function of g, γ and P to plot it.
In fact, we have:

$$\frac{q''}{(T_w - T_{sat})^{n_2}} = n_1 (g + n_4 g_{en} \gamma)^{n_3} P^{n_5}$$

Thanks to the model, we know the minimum values for the constants $n_i$. We obtain the following plot on Python (code attached and in the appendix).

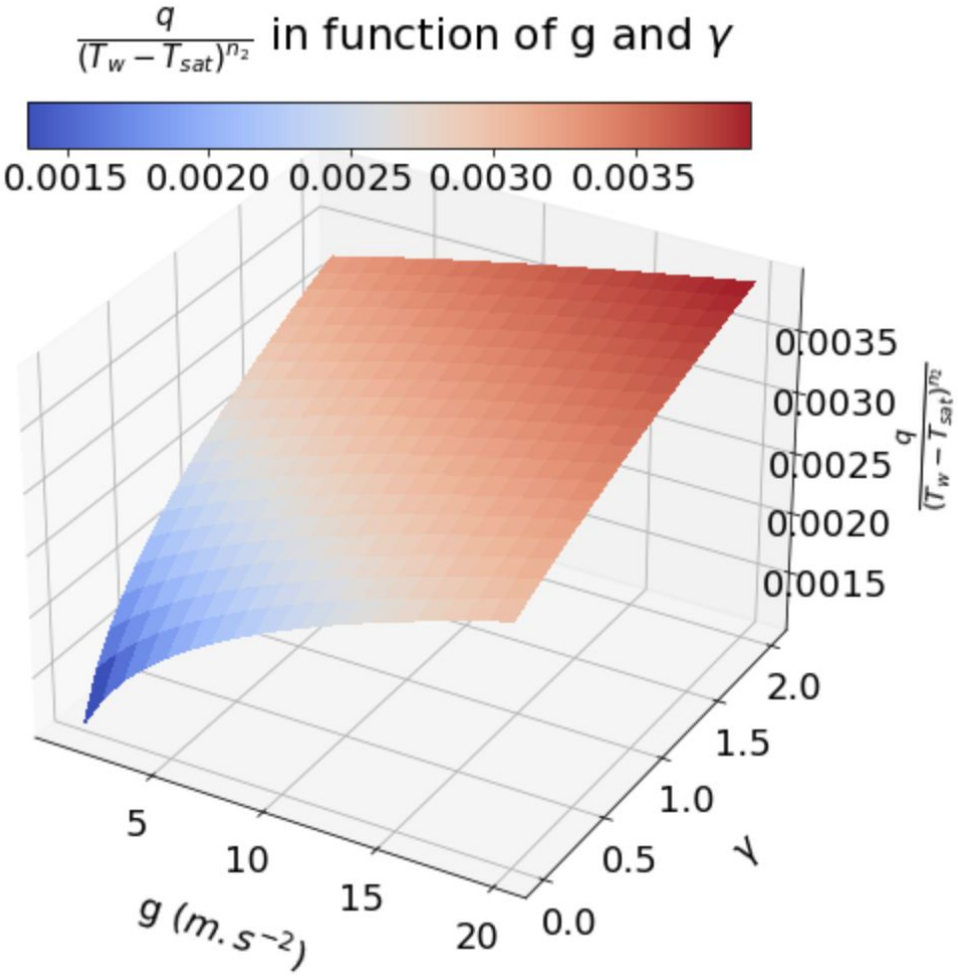



**Figure 7** - *Plot of $q''/ (T_w - T_{sat})^{n2}$ after having trained the model*

# Task 4

In the code attached, we can observe the function used to obtain the dimensionless variables for the three different values for P. In the appendix, there is the table where we stored our new values for those parameters.

Thanks to the equation 14, we can define a new error function such as:

$$Q_s = n_1 \mathrm{Ja}_s^{n_2} \left( \frac{g}{g_{en}} + n_3 \gamma \right)^{n_4} \mathrm{Pr}_l^{-n_5} \qquad (14)$$

We can therefore obtain:

$$ln(Q_S) = ln(n_1) + n_2 ln(Ja_S) + n_4 ln(\frac{g}{g_{en}} + n_3 \gamma) - n_5 Pr_l$$

We can define an error function $f_{err,i}$ similar as in task 3 for each data point with $n_0 = -1$ :

$$f_{err,i} = n_0 ln(Q_{S_{data,i}}) + ln(n_1) + n_2 ln(Ja_{S_{data,i}}) + n_4 ln(\frac{g_{data,i}}{g_{en}} + n_3 \gamma_{data,i}) - n_5 Pr_{l_{data,i}}$$

By summing on all the data points $N_D$ , we can also define the total error function $F_{err}$ which is the sum of the fractional absolute value of the error for each data point in the population:

$$F_{err} = \sum_{i=1}^{N_D} \frac{|f_{err,i}|}{|ln(Q_{s,data,i})|} = \sum_{i=1}^{N_D} \frac{\left| n_0 ln(Q_{S_{data,i}}) + ln(n_1) + n_2 ln(Ja_{S_{data,i}}) + n_4 ln(\frac{g_{data,i}}{g_{en}} + n_3 \gamma_{data,i}) - n_5 Pr_{l_{data,i}} \right|}{|ln(Q_{S,data,i})|}$$

Now, we can run a similar code to task 3 with this new error function and new initial guesses. We have a model with dimensionless parameters.

# Task 5

Thanks to the modifications made in task 4 for the error function, we can modify our code and run it with the five dimensionless parameters.

The modifications to the code (in comparison to task 4) are the following ones (**figure 8**). In summary:

- Used *ynewdata* (and *lynewdata*) instead of *ydata* (and *lydata*). (Array with our dimensionless parameters)
- Modified the 3 error functions (equations taken from task 4)
- Changed to plot $Q_{S,pred}$ and $Q_{S,data}$ instead of q''

```
ynewdata = ydimensionless(ydata) #Changes the array to the dimensionless array

''' need deepcopy to create an array of the same size as ydata,
#   since this array is a list(rows) of lists (column entries)  '''
lynewdata = deepcopy(ynewdata)  # create array to store ln of data values


# j is column, i is row downward for ydata[i][j] - both start at zero
# so it is:  ydata[row][column]
#now store log values for data
for j in range(DI):
    for i in range(ND):
        lynewdata[i][j]=math.log(ynewdata[i][j]+0.000000000010)
```

```
for i in range(ND):

    Ferr[i] = n[i][0]*lynewdata[i][0] + math.log(n[i][1]) + n[i][2]*lynewdata[i][1]
    Ferr[i] = Ferr[i] + n[i][4]*math.log(ynewdata[i][2] + n[i][3]*ynewdata[i][3] ) - n[i][5]*lynewdata[i][4]

    aFerr[i] = abs(Ferr[i])/abs(lynewdata[i][0])  #- absolute fractional error
```

```
for i in range(ND):
    Ferravgn[i] = -1.*lynewdata[i][0] + math.log(n1avg[k]) + n2avg[k]*lynewdata[i][1]
    Ferravgn[i] = Ferravgn[i] + n4avg[k]*math.log( ynewdata[i][2] + n3avg[k]*ynewdata[i][3] ) - n5avg[k]*lynewda

    #aFerravgn[i] = abs(Ferr[i])/abs(lydata[i][0])
    aFerravgn[i] = abs(Ferravgn[i])/abs(lynewdata[i][0])
```

```
#initialize values
Qpppred =  [[0.0]]
Qppdata =  [[0.0]]
for i in range(ND-1):
    Qpppred.append([0.0])
    Qppdata.append([0.0])
#calculate predicted and data values to plot
for i in range(ND):
    Qpppred[i] = n1min*(ynewdata[i][1]**n2min) * ((ynewdata[i][2] +n3min*ynewdata[i][3])**n4min)*ynewdata[i][4]*
    Qppdata[i] = ynewdata[i][0]
```

**Figure 8 -** *Modifications used for the dimensionless model*

Same as in task 4, we play with the initial guesses to find the best fit for our model. This time, we took:

$n_{0i} = -1.0$
$n_{1i} = 1.1$
$n_{2i} = 2.056$
$n_{3i} = 1.3$
$n_{4i} = 0.43$
$n_{5i} = 2.2$

After modifying the code with the new error function defined in the previous tasks, we found the following minimum constants (**Table 2**).

| | |
|---|---|
| **$n_1$** | **0.895** |
| **$n_2$** | **2.96** |
| **$n_3$** | **1.16** |
| **$n_4$** | **0.434** |
| **$n_5$** | **2.09** |

**Table 2** - *Constants obtained after running five constant with dimensionless model*

The output from the model is the following figure (**figure 9**):

```
ENDING: pop. avg n1-n5,aFerrmean: 1.044297445563221 2.861752929608925 1.1693713281239704 0.45298579518284854 2.049621
1646321516 0.04191930536341741
MINUMUM:  avg n1-n5,aFerrmeanMin: 0.8945246034965256 2.9578950825055634 1.1644876751759323 0.43443436218450415 2.0903
04621985502 0.03586273234989824
TIME AVG:    avg n1-n5,aFerrmean: 1.015549964638987 2.8838837795588765 1.159156407714235 0.4315598738977792 2.0789760
406192666 0.04152631871334516
```

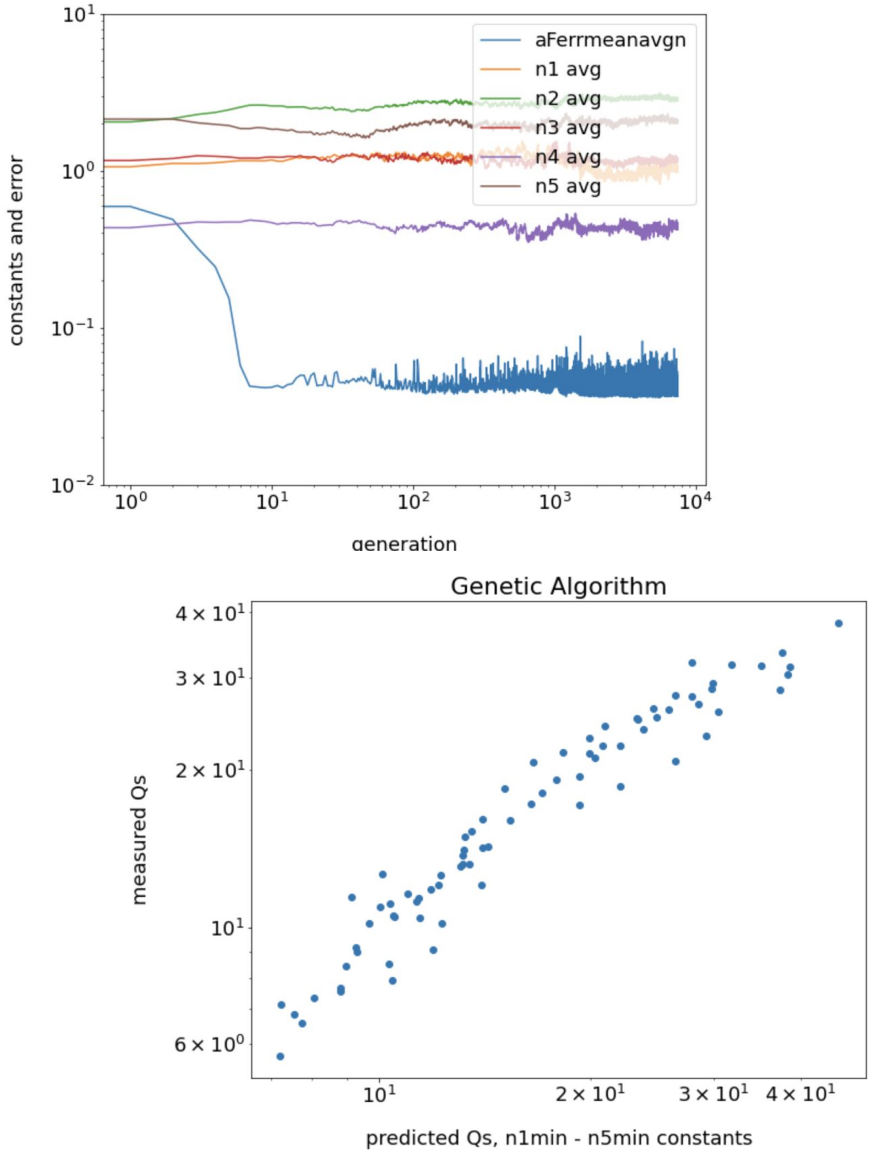



**Figure 9 -** *Output from the 5-constant dimensionless model*

We rearrange the equation (14) in order to solve it for $QsPr_l^{n5}/Ja_S^{n2}$ in function of $g/g_{en}$ and $\gamma$

$$Q_s = n_1 \mathrm{Ja}_s^{n_2} \left( \frac{g}{g_{en}} + n_3 \gamma \right)^{n_4} \mathrm{Pr}_l^{-n_5} \qquad (14)$$

From the previous equation, we have therefore:

$$\frac{Q_S}{J_{a_S}^{n_2} P_{r_l}^{-n_5}} = n_1 \left(\frac{g}{g_{en}} + n_3 \gamma\right)^{n_4}$$

And thus:

$$\frac{Q_S P_{r_l}^{n_5}}{J_{a_S}^{n_2}} = n_1 \left(\frac{g}{g_{en}} + n_3 \gamma\right)^{n_4}$$

We can plot for different values of $g/g_{en}$ and $\gamma$ with the minimum values we found for the constants $n_i$ after having runned the code. We obtain the following plot (**figure 10,** Python code available in the appendix).

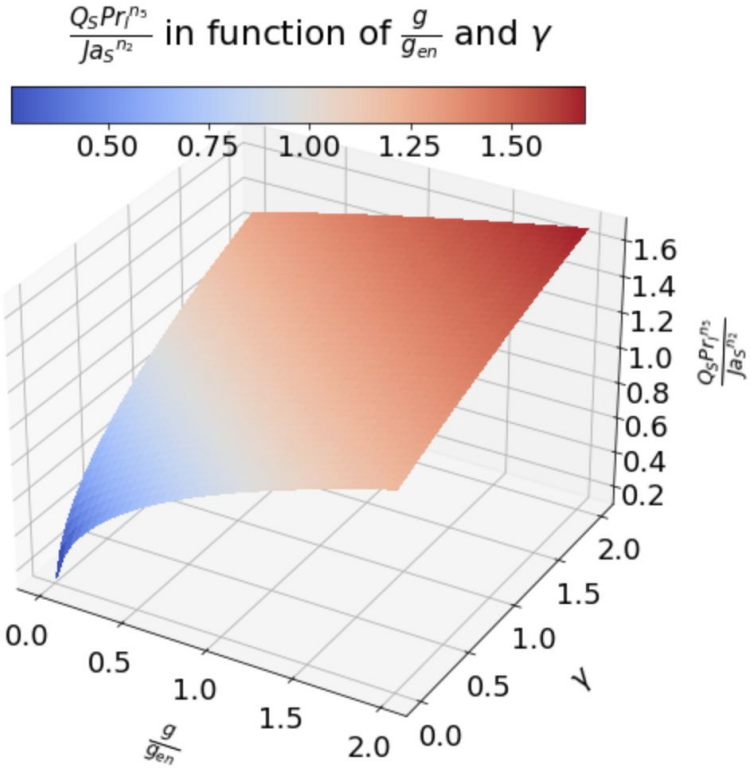



**Figure 10 -** *Plot of $QsPr_l^{n5}/Ja_S^{n2}$ for different values of g/gen and gamma*

# Task 6

- **Subdivision of tasks:**

Most of the code was modified separately in parallel by both Alexis and Lance. Alexis's code was then brought into the final report and checked to make sure that the outputs agreed with Lance's code.

- **Influence of initial guesses:**

Initial guesses play an important role when running a genetic algorithm. The error can change quickly when modifying one constant such as in task 2. Certain parameters seem to have a greater sensitivity on the final result than others. When having more constants - such as in task 3 - the influence of each individual constant on the final error is diminished. We saw that modifying one constant instead of another could easily change the result. Therefore, the stability of the model depends heavily on finding right initial guesses.

The speed of convergence is also influenced by the initial guesses. In fact, we can counter poor initial guesses by augmenting the number of generations (and therefore iterations). However, it is more computationally intense and longer.

Overall, initial guesses play an important role in the speed of convergence and final error. Having a relevant guess or knowing approximately what the final values should be helps finding the best fit.

- **Advantages of dimensionless raw data instead of non-dimensionless ones:**

Dimensionless raw data has advantages in that it is easier to compare results when modifying some parameters. We can also aim the same behavior when having different materials or conditions (example with heat transfer when changing the liquid and therefore changing liquid specific heat or thermal conductivity for instance).

However, it might be less intuitive and harder to analyze results than with dimension parameters. As an example, it is easier to imagine the real influence of temperature in experiences instead of using the dimensionless number $Ja_s$. One potential consequence is that it may be more difficult to come up with good initial guesses when using dimensionless parameters. As discussed in the previous section, having good initially guesses is critical to finding a solution that converges quickly with minimal error. Of course, a knowledgeable researcher may have a good understanding of good guesses for dimensionless parameters.

## Appendix

| | Qs | Jas | g/g_en | gamma | Prl |
|---|---|---|---|---|---|
| **0** | 6.825535505714105 | 5.618279569892473 | 0.01 | 1.79 | 4.83 |
| **1** | 7.336289863284548 | 5.739288668320927 | 0.01 | 1.79 | 4.83 |
| **2** | 7.645837958781787 | 5.912158808933003 | 0.01 | 1.79 | 4.83 |
| **3** | 9.162623626718256 | 6.015880893300248 | 0.01 | 1.79 | 4.83 |
| **4** | 10.493680437356378 | 6.2751861042183625 | 0.01 | 1.79 | 4.83 |
| **5** | 11.39136991429837 | 6.448056244830438 | 0.01 | 1.79 | 4.83 |
| **6** | 11.809259843219643 | 6.534491315136476 | 0.01 | 1.79 | 4.83 |
| **7** | 13.202226272957214 | 6.776509511993384 | 0.01 | 1.79 | 4.83 |
| **8** | 14.93569560774175 | 6.79379652605459 | 0.01 | 1.79 | 4.83 |
| **9** | 17.179919300096728 | 7.312406947890818 | 0.01 | 1.79 | 4.83 |
| **10** | 19.191981920828777 | 7.51985111662531 | 0.01 | 1.79 | 4.83 |
| **11** | 21.080225303361928 | 7.848304383788255 | 0.01 | 1.79 | 4.83 |
| **12** | 22.210075851926856 | 8.073035566583954 | 0.01 | 1.79 | 4.83 |
| **13** | 23.928067781936523 | 8.280479735318446 | 0.01 | 1.79 | 4.83 |
| **14** | 25.243647187799787 | 8.401488833746898 | 0.01 | 1.79 | 4.83 |
| **15** | 26.7449554509614 | 8.799090157154673 | 0.01 | 1.79 | 4.83 |
| **16** | 28.50937959529565 | 8.937386269644335 | 0.01 | 1.79 | 4.83 |
| **17** | 31.527473526393727 | 9.749875930521092 | 0.01 | 1.79 | 4.83 |
| **18** | 5.680207552374325 | 5.220678246484698 | 1.0 | 1.79 | 4.83 |
| **19** | 8.528050030948917 | 5.894871794871795 | 1.0 | 1.79 | 4.83 |
| **20** | 10.447248223031794 | 6.102315963606286 | 1.0 | 1.79 | 4.83 |
| **21** | 12.072375724392295 | 6.534491315136476 | 1.0 | 1.79 | 4.83 |
| **22** | 14.239212392872963 | 6.5863523573200995 | 1.0 | 1.79 | 4.83 |
| **23** | 18.572885729834297 | 7.623573200992555 | 1.0 | 1.79 | 4.83 |
| **24** | 20.786154612639557 | 8.10760959470637 | 1.0 | 1.79 | 4.83 |
| **25** | 23.262539376617465 | 8.384201819685691 | 1.0 | 1.79 | 4.83 |
| **26** | 25.8472659740194 | 8.505210918114145 | 1.0 | 1.79 | 4.83 |
| **27** | 28.478424785745926 | 9.110256410256412 | 1.0 | 1.79 | 4.83 |
| **28** | 30.413100382603663 | 9.179404466501241 | 1.0 | 1.79 | 4.83 |
| **29** | 7.9100183788681155 | 4.888888888888889 | 2.0 | 1.79 | 3.91 |
| **30** | 9.085327713464087 | 5.1158730158730155 | 2.0 | 1.79 | 3.91 |
| **31** | 10.167358529441328 | 5.1682539682539685 | 2.0 | 1.79 | 3.91 |
| **32** | 11.58519201244599 | 4.976190476190476 | 2.0 | 1.79 | 3.91 |
| **33** | 13.208238236411853 | 5.325396825396825 | 2.0 | 1.79 | 3.91 |
| **34** | 13.749253644400476 | 5.29047619047619 | 2.0 | 1.79 | 3.91 |

| | | | | | |
|---|---|---|---|---|---|
| **35** | 15.260365646023866 | 5.342857142857143 | 2.0 | 1.79 | 3.91 |
| **36** | 17.14459172212217 | 6.023809523809524 | 2.0 | 1.79 | 3.91 |
| **37** | 19.383276168971634 | 6.023809523809524 | 2.0 | 1.79 | 3.91 |
| **38** | 22.218943134980957 | 6.18095238095238 | 2.0 | 1.79 | 3.91 |
| **39** | 24.942675878647808 | 6.425396825396826 | 2.0 | 1.79 | 3.91 |
| **40** | 26.099329509520032 | 6.652380952380953 | 2.0 | 1.79 | 3.91 |
| **41** | 27.666408622314663 | 6.826984126984128 | 2.0 | 1.79 | 3.91 |
| **42** | 29.289454846280524 | 6.984126984126984 | 2.0 | 1.79 | 3.91 |
| **43** | 31.546794996853738 | 7.368253968253969 | 2.0 | 1.79 | 3.91 |
| **44** | 33.43102107295204 | 7.542857142857144 | 2.0 | 1.79 | 3.91 |
| **45** | 38.244192633678395 | 8.031746031746032 | 2.0 | 1.79 | 3.91 |
| **46** | 6.562419624541453 | 5.13424317617866 | 2.0 | 1.79 | 4.83 |
| **47** | 7.537496125357754 | 5.358974358974359 | 2.0 | 1.79 | 4.83 |
| **48** | 8.435185602299745 | 5.393548387096774 | 2.0 | 1.79 | 4.83 |
| **49** | 10.958002580602237 | 5.600992555831265 | 2.0 | 1.79 | 4.83 |
| **50** | 11.406847319073233 | 5.428122415219189 | 2.0 | 1.79 | 4.83 |
| **51** | 12.66051710583705 | 5.618279569892473 | 2.0 | 1.79 | 4.83 |
| **52** | 14.223734988098103 | 6.2751861042183625 | 2.0 | 1.79 | 4.83 |
| **53** | 16.081023561081533 | 6.2751861042183625 | 2.0 | 1.79 | 4.83 |
| **54** | 18.43358908686054 | 6.430769230769231 | 2.0 | 1.79 | 4.83 |
| **55** | 20.69329018399038 | 6.638213399503722 | 2.0 | 1.79 | 4.83 |
| **56** | 21.652889280031822 | 6.862944582299422 | 2.0 | 1.79 | 4.83 |
| **57** | 22.952991281120227 | 7.070388751033913 | 2.0 | 1.79 | 4.83 |
| **58** | 24.299525496533207 | 7.191397849462366 | 2.0 | 1.79 | 4.83 |
| **59** | 26.1722914742915 | 7.588999172870141 | 2.0 | 1.79 | 4.83 |
| **60** | 27.73550935655256 | 7.779156327543424 | 2.0 | 1.79 | 4.83 |
| **61** | 31.72867978846693 | 8.280479735318446 | 2.0 | 1.79 | 4.83 |
| **62** | 13.06279915594627 | 7.209891936824605 | 1.0 | 0.0 | 4.54 |
| **63** | 12.044918702236169 | 7.0361596009975065 | 1.0 | 0.0 | 4.54 |
| **64** | 11.196684990811088 | 6.862427265170408 | 1.0 | 0.0 | 4.54 |
| **65** | 10.51809802167102 | 6.688694929343308 | 1.0 | 0.0 | 4.54 |
| **66** | 7.125163175970691 | 5.906899418121363 | 1.0 | 0.0 | 4.54 |
| **67** | 10.178804537100987 | 6.514962593516209 | 1.0 | 0.0 | 4.54 |
| **68** | 8.991277341105873 | 6.42809642560266 | 1.0 | 0.0 | 4.54 |
| **69** | 11.097299223575993 | 6.2924731182795695 | 0.01 | 1.71 | 4.83 |
| **70** | 12.614084891512462 | 6.65550041356493 | 0.01 | 1.71 | 4.83 |
| **71** | 14.038006130799761 | 6.828370554177006 | 0.01 | 1.71 | 4.83 |
| **72** | 15.988159132432358 | 7.191397849462366 | 0.01 | 1.71 | 4.83 |

| | | | | | |
|---|---|---|---|---|---|
| **73** | 18.108563586588442 | 7.450703060380479 | 0.01 | 1.71 | 4.83 |
| **74** | 21.451683017958615 | 7.848304383788255 | 0.01 | 1.71 | 4.83 |
| **75** | 25.026963520951718 | 8.280479735318446 | 0.01 | 1.71 | 4.83 |
| **76** | 32.11561490783848 | 8.799090157154673 | 0.01 | 1.71 | 4.83 |

**Table 3 -** *New dimensionless parameters used for the tasks 4 & 5*

*Appendix 2 - Code used for this project*