# Coursework 2

## Q1

### 1.1 Assumptions to be made and their justifications:

1. The systematic error is a singe realization drawn from some statistical parent distribution. The uncertainty for a systematic error source is estimated by the standard deviation of the assumed distribution. This assumption essentially gives the power for one to estimate the systematic uncertainty b in the same way that random uncertainty s is estimated, that is through the unbiased estimator for population variance (thus population standard deviation) with unknown distribution from a large enough sample: $s\ (or\ b) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}$, note the (N-1) accounts for the loss of one degree of freedom due to one sample already used. This assumption is reasonable when the exact systematic standard uncertainty cannot be known without detailed data such as a calibration for a measurement equipment.
2. The chosen concentration to derive a nominal value for $A_{BA,sample}$ is 480 mg/l, as it can be read from the coursework briefing that the historical average BA concentration in YummyFood's Mayonnaise is 480 mg/l. The derived $A_{BA,sample}$'s nominal value will be used in the calculation for the $X_{dre,best}$ with the historical data.
3. Assume that the AUP data and CalibConc data are related through a linear relationship: $AUP = b_1 CalibConc + b_0$, where $b_1$is the gradient, $b_0$ is the y-intercept. A linear regression is performed on CalibAUP_Historical (y) against CalibConc (x), and the resulting b1 and b0 vectors with size 1x3000 are plotted in terms of histograms. (see plots in Appendix A)
4. Assume that the distributions of both b1 and b0 for historical data can be approximated by normal distributions characterised by the means and standard deviations of the b1 and b0 vectors. Justification on this assumption can be mainly made in two respects:
   - From Central Limit Theorem (CLT), the given 3000 sample size is large enough, the mean of the sample will closely approximate the actual mean of the population distribution of b1 and b0. Therefore the use of sample statistics to characterise the distribution is reasonable.
   - Normal distribution is assumed for b1 and b0 because when experimentally mapping normal distribution with mean and standard deviation stated above to fit both histograms of b1 and b0, a considerably good degree of match occurred. (see plots of normal distribution mapping in Appendix A)
5. Assume all uniform distributions to be symmetrical, meaning they are all two-tailed and the mean is $\frac{1}{2}(a + b) = 0$, hence $a = -b$. Considering that these distributions are for the random errors of variables, the means of those distributions should be zero as the random error terms should fluctuate around 0, with equal degree of fluctuation on both sides of 0. And such uniform distribution must be a symmetrical one, therefore the assumption is valid.
6. Assume the triangular distribution is symmetrical, meaning it is also two-tailed and the symmetric distribution is a special case of triangular distributions, where $c = \frac{(a+b)}{2}$. Since the triangular distribution is also for a random error, it must also fluctuate around 0 and with equal distribution on both sides of 0, which means its peak is at 0, and it is symmetrical.
7. Assume all normal distributions are symmetrical and two-tailed. For random errors, normal distribution will have a mean of 0 and they are symmetrical by nature, therefore the assumption is valid.

8. Assume the standard percentage uncertainty (%std) of an error source can be estimated to be half of its corresponding two-tailed value (i.e. half of the percentages shown in the table). By the assumption of symmetry w.r.t. 0, the two-tailed value covers the combined deviations on both sides of zero, while the one-tailed value is the deviation on a single side of zero, which is exactly half of the two-tailed value. And the standard uncertainty (as well as the %) should be the one-tailed value from the symmetrical distribution, thus the assumption is valid.
9. Assume the standard deviation (std) of each error source (apart from deltaC) can be estimated by the following equation: $std = (nominal\ value) \times (\%std)$. This assumption is based on the definition of percentage uncertainty.
10. Assume the expanded uncertainty $U$ which defines the range $\pm U$ for which the true value of variable lies is calculated via the confidence interval of the associated variable following t distribution. The use of t distribution instead of normal distribution can be justified as such:
    - To estimate the population variance in order to calculate the confidence interval to trap the population mean, from a given sample of size N, the unbiased estimator $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2}$ is used. However, when sample size N is small (i.e. N<30), the estimated standard deviation is no longer accurate to the true value, and the estimated distribution becomes more spread out due to the increased uncertainty. To reflect that, t distribution with larger tails is used.
    - If the sample size is big enough, t distribution will closely approximate the normal distribution, so the result is still quite accurate.
    - Given all above, considering both cases of small and large sample sizes, t distribution is preferred than normal distribution, therefore the assumption is reasonable.
11. Assume a 95% level of confidence (meaning $t_{95} \approx 2$). The expanded uncertainty U is the related to the standard uncertainty $u_c$ through: $U = t_{95}u_c = 2u_c$. The approximation that $t_{95} \approx 2$ is made as in most applications the effective degrees of freedom is considered to be close to infinity, meaning when looking up t values from the table, $t_{95,\infty} = 1.96 \approx 2$, thus $U = 2u_c$.
12. Assume that the standard deviation of the deltaC is calculated through: $std\ of\ \Delta C = (nominal\ value\ of\ X_{dre\ without\ \Delta C}) \times (\%std\ of\ \Delta C)$. This is instructed by the briefing sheet to consider deltaC as a combined uncertainty and only affects the $X_{dre}$.
13. Finally assume that the expanded uncertainty of the $X_{dre}$ can be estimated by the MCM. This is a very reasonable assumption as from table 1 it can be seen that many random error sources are not approximated by normal distributions, and the DRE is a quite complex one involving different variables with errors of different sources. The MCM has the capability to deal with distributions other than normal and complex DREs.

Many of the above assumptions and their explanations not only just apply to only Q1, but also to other questions in this coursework.

## 1.2 The conversion of mean and std for uniform and triangular distributions

There are in total two uniform distributions for error sources: one is for the random error distribution of the $A_{BAsample}$, the other is for the random error distribution of $V_{sample}$. There is also one triangular distribution for the random error distribution of the $V_{HPLC}$.

These distributions' setup requires parameters different from the mean and std that are used to set up normal distributions. However, a set of relationships occur between these statistical parameters.

**For uniform distributions:**

$$\frac{1}{2}(a+b) = mean$$

$$\sqrt{\frac{1}{12}(b-a)^2} = std$$

Where the above a is the lower bound, b is the upper bound.

**For triangular distributions:**

$$\frac{1}{3}(a+b+c) = mean$$

$$\sqrt{\frac{1}{18}(a^2+b^2+c^2-ab-ac-bc)} = std$$

And for the special case of symmetrical triangular distribution:

$$c = \frac{1}{2}(a+b)$$

Where the above a is the lower bound, b is the upper bound, c is the peak.

With the above sets of equations, whenever given the knowledge of mean and std, each a and b for uniform distribution and each a, b and c for triangular distribution can be derived. In fact, this coursework makes use of the matlab symbolic equation solver to facilitate the process (see details in codes attached in Appendix).

## 1.3 The MCM simulation result for historical data

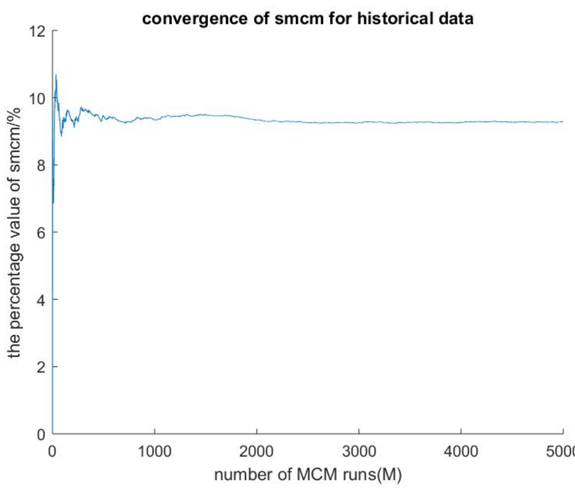


Fig.1. A manual MCM simulation with historical data, for the percentage values of expanded uncertainty (smcm)

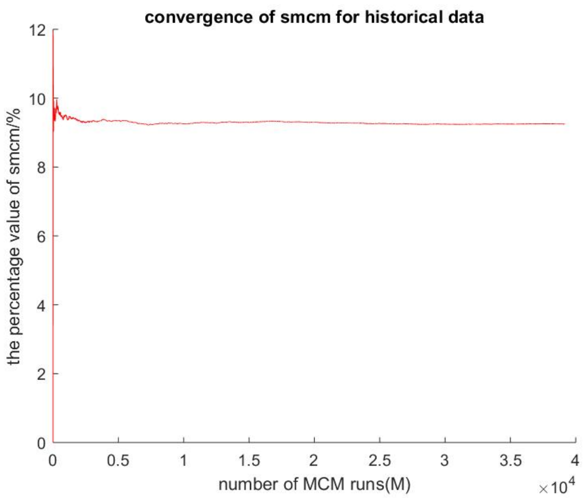


Fig.2. Adaptive MCM simulation with historical data, for the percentage values of expanded uncertainty (smcm)

The calculated $X_{dre,best}$ is 480 mg/l, and the number of Monte Carlo episode M is set to be 5000 for the manual MCM. Plot of M against smcm as a percentage value is produced, and value of the final (converged) smcm % is read from the last element of the corresponding matlab array used for plotting.

As can be seen from figure 1, for the manual MCM, the expanded uncertainty (smcm) first starts to fluctuate, but quickly converge to a relative constant after roughly 1000 episodes. The convergence is visually recognized as degree of fluctuation becomes almost negligible comparing with the initial fluctuations. The value for expanded uncertainty at convergence (M=5000) is 44.5628 mg/l, and the percentage expanded uncertainty at M=5000 is 9.2839%, as shown in figure 1.

To mathematically and deterministically guarantee the convergence, a second MCM which automatically forces the convergence before termination is also implemented.

To set up the adaptive MCM, a very large value of M is selected (M=500000), to avoid the termination of for loop before reaching convergence. Then, a convergence threshold needs to be determined. The design of convergence threshold essentially depends on what kind of value of smcm are to be picked for examination for satisfying convergence. In this approach, the deviation (as an absolute value) from the current value of smcm (as a %) to previous values of smcm recorded is compared against the convergence threshold. Therefore the threshold is set to be below or equal 0.0001. This is a very strict threshold as even if converting 0.0001 % deviation to real value of concentration deviation, that is $480 \times 0.0001\% = 0.00048\, mg/l$, still a concentration difference that is almost close to 0 comparing to level of hundreds, therefore a convergence.

Specifically, from any current value of % smcm as the main for loop is running, a trace-back to as far as 500 recorded % smcm is performed, to calculate an 1x500 array that stores each successive absolute deviation from current one to previous one, previous one to the one before the previous one …etc. Then, an average of this array is used to compare against the threshold. If less or equal to the threshold, the convergence conditional is changed to true. And this change of conditional will break the main for loop and exit the smcm values stored until that point. The algorithm is designed in such a way because convergence depends mainly on the level of deviation along a certain stretch of values. To take periodicity and randomness into account, the algorithm successively tracks back the deviation as an absolute value for 500 times, ensuring a convergence that is continuous for a considerably long stretch, thus approximating a convergence to be global. For detailed execution, please refer to codes in Appendix B.

Finally, from figure 2, the adaptive MCM produced a final (converged) expanded uncertainty value of 9.2469%. The automatic MCM takes much longer time with M=39152, and achieves a slightly lower expanded uncertainty than the manual method in figure 1. However, this difference is not too great (9.28% compared with 9.25%). For reasons of simplicity, it may be acceptable to just use manual MCM for historical data.

## Q2

2.1 Assumptions and their justifications:

1. The chosen concentration to derive a nominal value for $A_{BA,sample}$ is 480 mg/l, same as Q1.
2. Assume that the AUP data and CalibConc data are related through a linear relationship: $AUP = b_1 CalibConc + b_0$, where $b_1$is the gradient, $b_0$ is the y-intercept. A linear regression

is performed on CalibAUP_2019 (y) against CalibConc (x), and the resulting b1 and b0 vectors with size 1x500 are plotted in terms of histograms and both variables are assumed to be approximated by normal distributions, same as Q1. (see corresponding plots in Appendix A 2019 data)

…the rest of assumptions are all the same as Q1

## 2.2 The conversion of mean and std for uniform and triangular distributions

Same as what is described in Q1.

## 2.3 The MCM simulation result for 2019 data

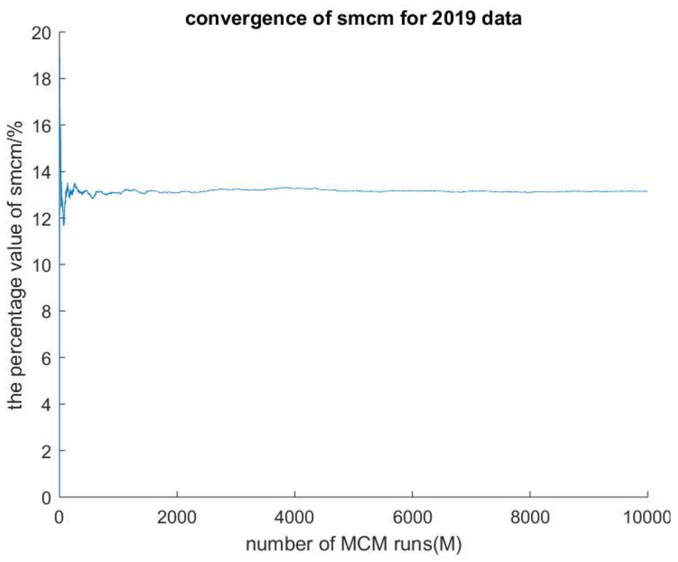


Fig.3. Manual MCM simulation with 2019 data, for percentage values of expanded uncertainty

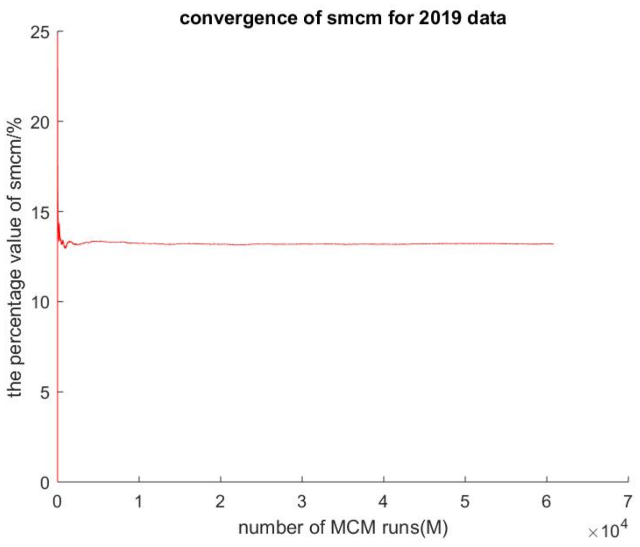


Fig.4. Adaptive MCM simulation with 2019 data, for percentage values of expanded uncertainty

The calculated $X_{dre,best}$ is 480 mg/l, and the number of Monte Carlo episode M is initially set to be 10000, for the manual MCM. The result is shown by plotting M against the expanded uncertainty (smcm) as a percentage of$X_{dre,best}$, as shown in figure 3. The value of expanded uncertainty at M=10000 is 63.0971 mg/l. The calculated % smcm at M=10000 is 13.1452%.

Applying the automatic MCM method described in Q1, an alternative result is obtained shown in figure 4. The calculated % expanded uncertainty at convergence is 13.1978%, which takes a total of 60858 runs to reach. It is interesting to note that for 2019 data, compared with historical data, it takes a lot longer to converge, both for manual and automatic MCM. This increased run time could although be explained by the fact that 2019 data contributes greater uncertainty to the DRE than historical data does, thus a longer time to stabilize. There is also a slight difference between the results of manual MCM and automatic MCM, and now this difference is now a little larger than that in Q1.

**Q3**

Comparing the convergence result from Q2 with that of Q1, the expanded uncertainty is 13.1978% in 2019, while in the past 10 years the expanded uncertainty is 9.2469%. From this comparison alone, the MCM simulated uncertainty does increase to an abnormal level in 2019 comparing with historical data, and the performance in 2019 cannot satisfy the 10% error threshold anymore. But this conclusion is drawn merely with only one concentration value (480 historical average) tested.

To look at this issue more broadly, BA concentrations from a range of [200,400,480,600,800] mg/l are analysed, for both historical and 2019 data. This range is selected as according to the briefing sheet, the maximum allowable BA concentration is set to be 1000 mg/l, and 480 is in there for a reference.

The following MCM simulations are done with the same assumptions listed in Q1 and Q2, except for the concentration value used (now multiple concentrations across a range are tested instead of only one value of 480 before). The expanded uncertainty % values corresponding to concentrations tested are plotted against number of MCM runs M, and all curves are plotted in the same figure for each dataset, for ease of comparison.

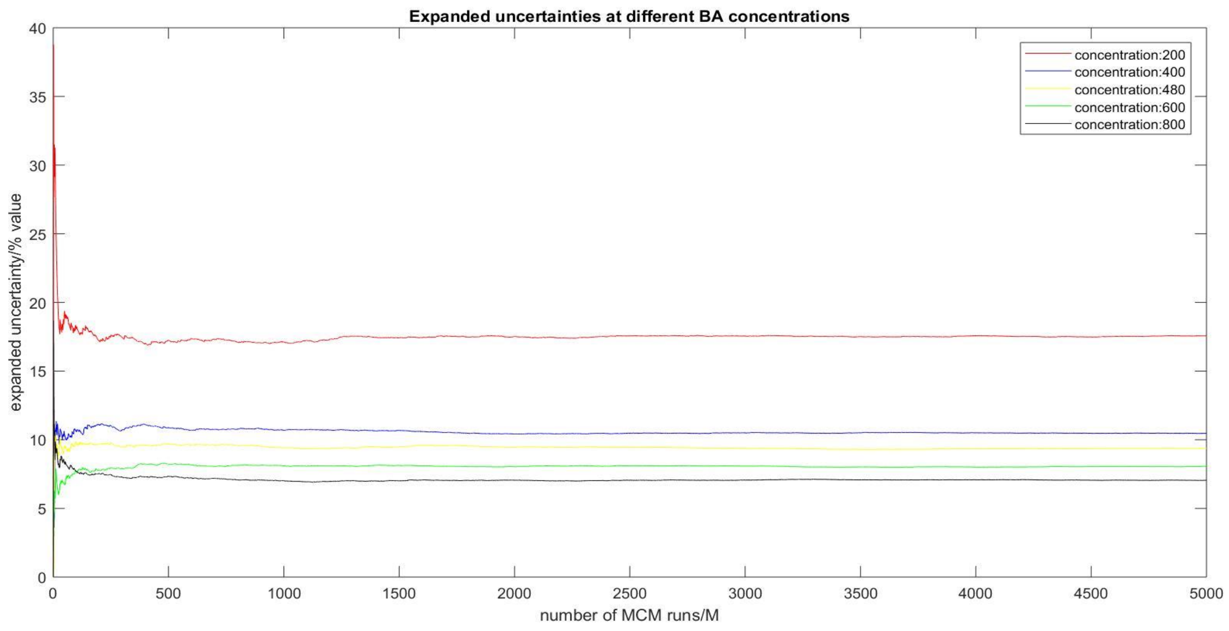


Fig.5. MCM on different expanded uncertainties for different concentrations tested, for historical dataset

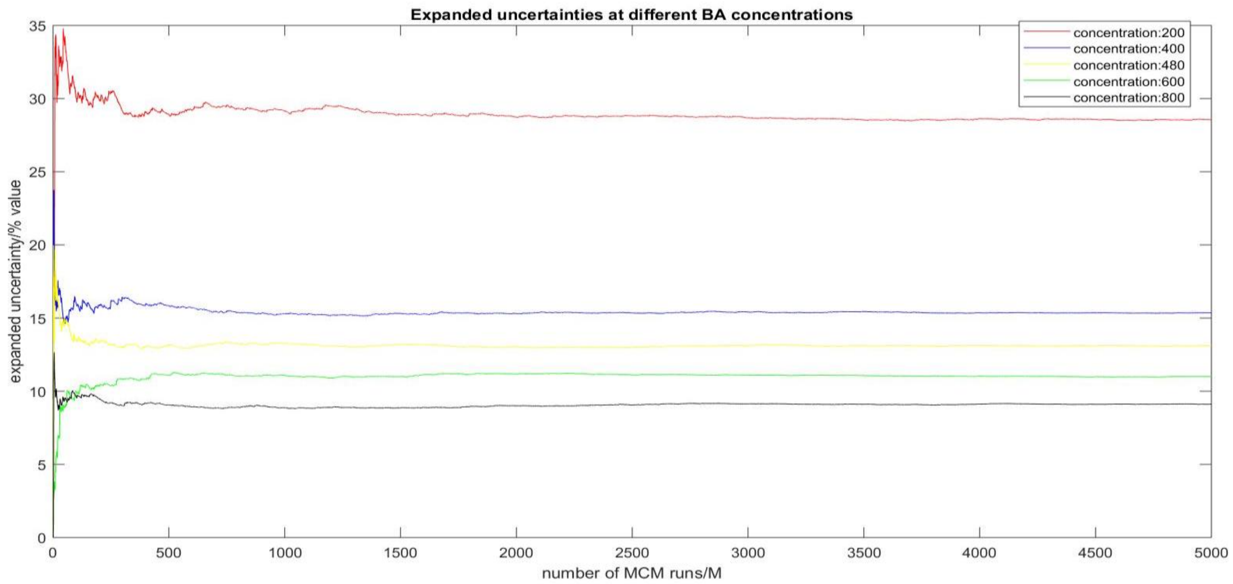


Fig.6. MCM on different expanded uncertainties for different concentrations tested, for 2019 dataset

| Historical data | | | | | |
|---|---|---|---|---|---|
| Concentration tested/mgl-1 | 200 | 400 | 480 | 600 | 800 |
| % expanded uncertainty at convergence | 17.5707 | 10.4596 | 9.3671 | 8.0611 | 7.0437 |

Table 1. Calculated values for different uncertainties corresponding to different concentrations. for historical dataset

| 2019 data | | | | | |
|---|---|---|---|---|---|
| Concentration tested/mgl-1 | 200 | 400 | 480 | 600 | 800 |
| % expanded uncertainty at convergence | 28.5394 | 15.3745 | 13.1255 | 11.0120 | 9.1053 |

Table 2. Calculated values for different uncertainties corresponding to different concentrations, for 2019 dataset

The final value for expanded uncertainty at each concentration is then recorded into table 1 and 2. Now, with a range of concentrations tested for both historical and 2019 data, some general trend can be observed based on results shown in figure 5 and 6, table 1 and 2. It can be observed that at every concentration level, the expanded uncertainty calculated from 2019 data is higher than that from historical data. And looking at historical data alone (figure 5 and table 1), only expanded uncertainties at concentration 200 mg/l and 400 mg/l exceed 10% which is the maximum error tolerance. Looking at 2019 data alone (figure 6 and table 2), expanded uncertainties at concentrations 200 mg/l, 400 mg/l, 480 mg/l and 600 mg/l all exceed the 10% threshold.

From observations above, one can conclusively say that at across the entire concentration range, 2019 data leads to an expanded uncertainty higher than that from historical data, thus supporting the claim that performance of the HPLC equipment has caused a increase in uncertainty in measurement of BA in 2019. This can only be explained by the gradual worsening of HPLC equipment over time as indicated by the clear decline from historical averages.

And in 2019, only for measurement of very high concentrations like 800 mg/l (since 1000 mg/l is the maximum allowed), the expanded uncertainty can be maintained below the 10% error threshold. Other measurements for average or below average BA concentrations do not satisfy the 10% threshold at all. In contrast, for historical data, most measurements (3 out of 5 concentrations tested) of BA does satisfy the 10% error threshold.

From both figure 5 and 6, for both historical and 2019 data, it can also be observed that as the concentration decrease, the expanded uncertainty increases. And again for both datasets, the rate of increase of expanded uncertainty after concentration value decreases down to 400 mg/l is vastly higher than that before 400 mg/l. This can be verified by the wide separation between the red line (200 mg/l) and all other coloured lines, appearing in both figure 5 and 6. Such phenomena may suggest an inverse relationship between expanded uncertainty and the actual concentration that it is trying to measure. The dramatic increase in %uncertainty when measuring smaller BA concentrations may be explained by the fact that if the absolute uncertainty remains the constant, measurement on a smaller nominal (true) value will have higher percentage uncertainty, as$\%uncertainty = \frac{absolute\ uncertainty}{nominal\ value}$.

To further support the relationship described above and avoid bias in separate datasets, historical and 2019 data are merged into one array of size 10x3500 by horizontal matrix concatenation, and another MCM is run with this data. Results are shown in table 3. Again, same trend occurs as small concentrations like the red curve has a far greater % uncertainty than large concentrations.

| Merged data | | | | | |
|---|---|---|---|---|---|
| Concentration tested/mgl-1 | 200 | 400 | 480 | 600 | 800 |
| % expanded uncertainty at convergence | 19.3779 | 11.3118 | 9.8917 | 8.6337 | 7.3931 |

Table 3. Calculated values for different uncertainties corresponding to different concentrations, for merged dataset

One suggestion that can be made to the analytics department is that they may need to run a maintenance check on their HPLC equipment, to see if there is any further wear on those measuring components that has occurred over the course of last few years. If a repair or maintenance is not possible, mark any measurement that is below 400 mg/l as unreliable. If budget allows, an upgrade to more advanced measuring equipment that does not require calibration before every batch test is more ideal, as for HPLC the calibration essentially imposes a linear transformation and complicates the equation to calculate the BA concentration, potentially leading to more uncertainty.

## Q4

The Elementary Effects Method (EEM) is applied with some prior assumptions. Firstly, historical and 2019 data are merged by horizontal concatenation and values and variations of b0 and b1 are obtained from the merged dataset. This is to ensure the sensitivity analysis takes account of all existing data from all past years, improving the accuracy of the final judgement, in case there is some bias in the data of one particular year. The concentration selected to obtain the nominal value for $A_{BAsample}$ is assumed to be the historical average 480 mg/l. Uniform distributions are assumed for all variables, including b0 and b1 with their mean and standard deviations estimated from merged dataset. The mean and standard deviation for each variable are then converted into the range of variation for their respective uniform distributions in the same way as previous questions. All other assumptions and the calculation for distributions follow the same assumptions listed in Q1.

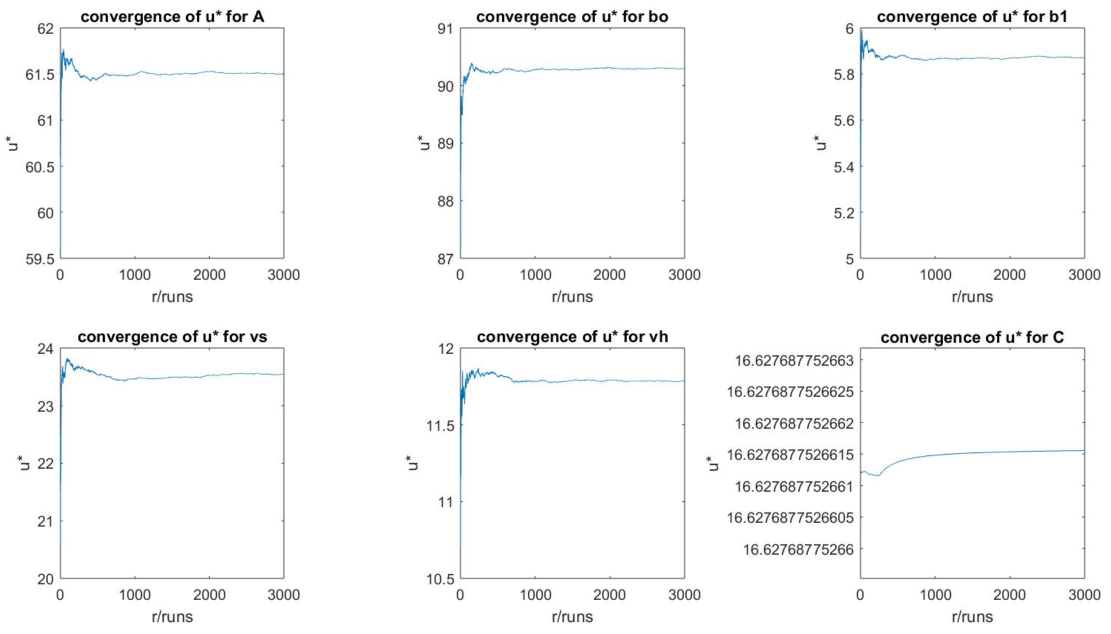


Fig.7. Convergence of sensitivity statistics $\mu^*$ for each of 6 variables, with r=3000 runs, as matlab subplots

To set up parameters of EEM, the number of inputs k for equation (1) in the briefing sheet is 6. For Morris's sampler, level p is set to be even as $p = 4$, and $\Delta = \frac{p}{2(p-1)} = \frac{2}{3}$ is chosen, as it ensures equal chance of sampling from every level within each variable dimension. EEM is then started, elemental effects are calculated and two sensitivity metrics $\mu^*$ and $\sigma^2$ stored across each run, to plot the convergence curves. An automatic convergence mechanism similar to the one introduced in Q1 is employed to help determine the most efficient number of runs r.

The first test run of the EEM with the automatic mechanism converged at convergence threshold 0.001% after 2333 runs (see the corresponding plot in Appendix D). With this information, the parameter r is chosen to be 3000 to guarantee a convergence. The result for $\mu^*$ can be seen in figure 7. Visually examining the convergence plots, $\mu^*$ for all variables converge as the curves all level at r=3000 runs. The proof for convergence for the other crucial sensitivity metrics $\sigma^2$ is not included in the main report, in order not to exceed the limit for total number of figures, but please go to Appendix D for the full proof of convergence.

All elementary effects (EE) for all 6 variables across r runs are stored in the EE_matrix, and sensitivity metrics $\mu, \mu^*$ $and$ $\sigma^2$ for each of the 6 input variable are calculated from the cumulative EE_matrix and then recorded into table 4 after convergence. Now, from a brief observation of table 4's result, it seems $\mu^*$ for all variables are merely the absolute values of their $\mu$. After investigating deeper into the EE_matrix that stores all EEs for each variable, it is recognized that all EEs for all variables take the same sign, and for $b_0, b_1, V_{sample}$ their EEs all take negative sign, thus causing $\mu^* = |\mu|$ for all variables. This result could be justified as no variable has the nominal value capable of changing the sign of the output, as the output value by nature of the equation will be positive no matter how much each variable deviate itself.

| Variable name | $A_{BA,sample}$ | $b_0$ | $b_1$ | $V_{sample}$ | $V_{HPLC}$ | $\Delta C_{HPLC}$ |
|---|---|---|---|---|---|---|
| $\mu$ | 61.4127 | -90.1526 | -5.8742 | -23.6042 | 11.7864 | 16.6277 |
| $\mu^*$ | 61.4127 | 90.1526 | 5.8742 | 23.6042 | 11.7864 | 16.6277 |
| $\sigma^2$ | 1.6464 | 3.5836 | 0.2610 | 4.2445 | 1.0768 | $1.2646 \times 10^{-25}$ (very close to 0) |

Table 4. Sensitivity metrics for all 6 input variables, at convergence

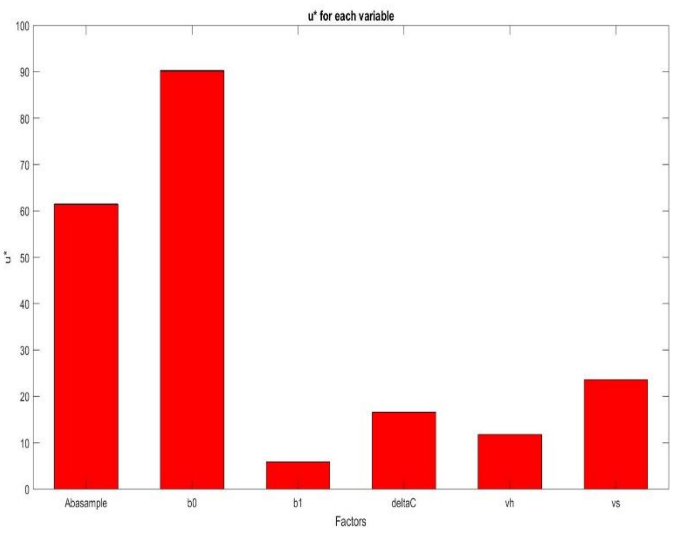


Fig.8. Barplot for $\mu^*$ for all 6 input factors of the model

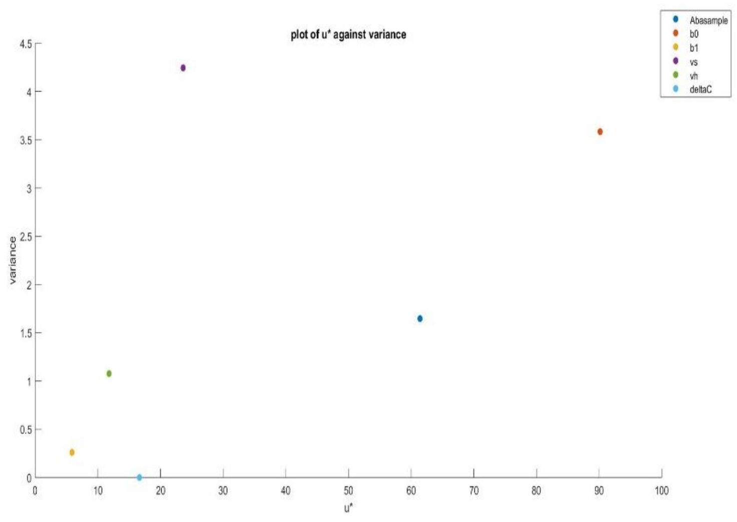


Fig.9. Relationship between $\mu^*$ and $\sigma^2$ for all 6 input factors of the model

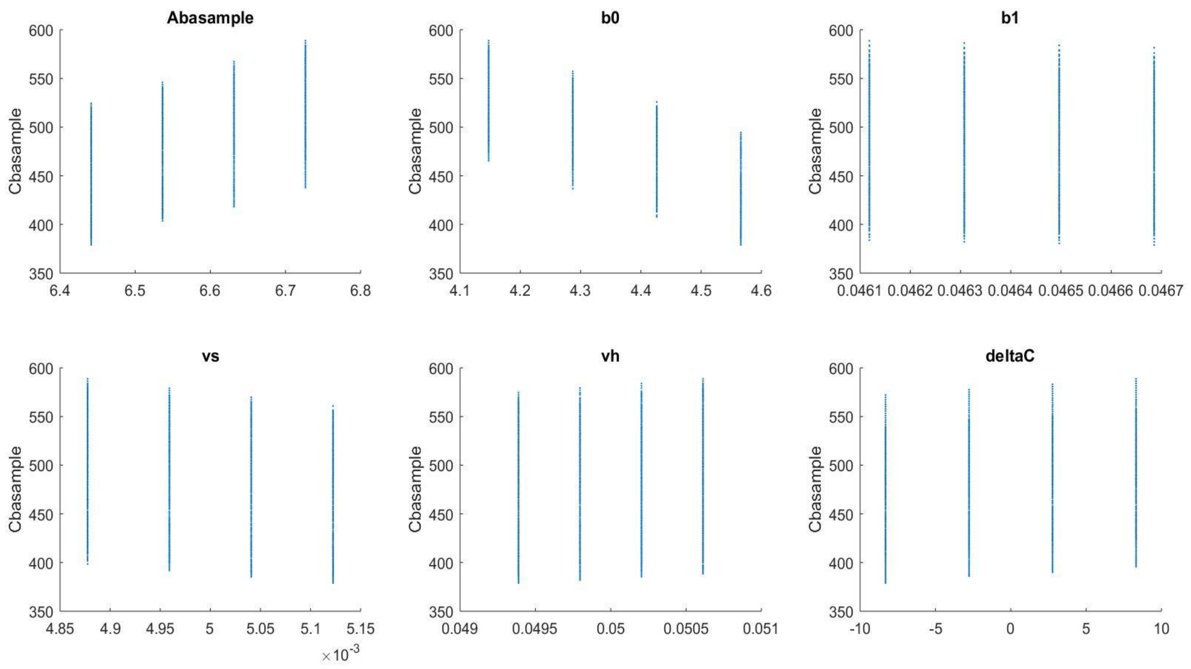


Fig.10. Scatter plots of model output $C_{BA,sample}$ against each input factor, as matlab subplots

As all EEs take the same sign, it can be concluded that all 6 input factors do not have the effect of oscillating signs, and $\mu^*$ and $\mu$ have the same value. Thus the factor fixing setting can be sufficiently determined by only looking at $\mu^*$ to determine the relative importance of each factor. From figure 8, $b_0$ and $A_{BA,sample}$ are the highest and second highest variable, others are significantly lower, with $b_1$ being the lowest of all. This indicates that $b_0$ and $A_{BA,sample}$ are essential and cannot be fixed. Variable $b_1$ with its very low $\mu^*$ can be fixed without significantly affecting the variance of the output of model, thus it proves the change in $b_1$ does not contribute to a great change of the expanded uncertainty. While the other factors $V_{sample}, V_{HPLC}, \Delta C_{HPLC}$ can be regarded as flexible, meaning they could be fixed if for a simplification and the high degree of accuracy for the output is not a primary requirement. Figure 9 is a scatter plot that visualises table 4's result in the $\mu^*$-$\sigma^2$ space. It considered $\sigma^2$ which can indicate interaction in model alongside with $\mu^*$ to decide which factor is the most important. The scatter dot for $b_0$ appears far on the top-right corner, suggesting both a high importance and high interaction. $A_{BA,sample}$ also has a quite high importance but with less interaction. Interestingly, $V_{sample}$ appears highest in $\sigma^2$ direction, indicating highest interaction, but with low $\mu^*$. And for $\Delta C_{HPLC}$ it has zero interaction (its $\sigma^2$ closely approximate 0). The number of inputs of this model is 6 so not too large, facilitating the use of scatter plots of each variable with their model outputs to help determine the important factor. From figure 10, the b0 scatters shows the strongest pattern of declination, indicating b0 plays the key role in producing the output. While $A_{BA,sample}$ has a second strongest trend of ascent, also indicating an important role. And $V_{sample}, V_{HPLC}$ and $\Delta C_{HPLC}$ show some weak patterns, indicating they all affect model output to much smaller extents.

Given all above, this report based on the sensitivity analysis result above concludes that the variable $\boldsymbol{b_0}$ and $\boldsymbol{A_{BA,sample}}$ are the **important factor** and should be prioritised for measurement by high-fidelity equipment.

Re-examining equation (1) of the model and relating to knowledge gained from previous questions, results seen from sensitivity analysis may be explained. The high importance of b0 may be partly due to its high %uncertainty (much higher than b1) obtained from the linear regression result. $A_{BA,sample}$ also has the highest %uncertainty in briefing table, and $(A_{BA,sample} - b_0)$ is located on numerator position in equation (1) to be magnified many times larger due to values of b1, vh/vs take. Such mathematical nature causes uncertainties of A and b0 to grow and propagate in model more than any other. Others like b1 have its %uncertainty and nominal value too small to make an impact. The $\sigma^2$ for deltaC is 0 may be because deltaC is linearly added to the model, does not go through any multiplication or division with any other variable, thus lacking the interaction.

*The main report ends here.*

## Appendices:

**A:** analysis of historical and 2019 data for b0 and b1

**B:** **Q1** and **Q2**

**C:** **Q3**

**D:** **Q4** and additional proof for convergence

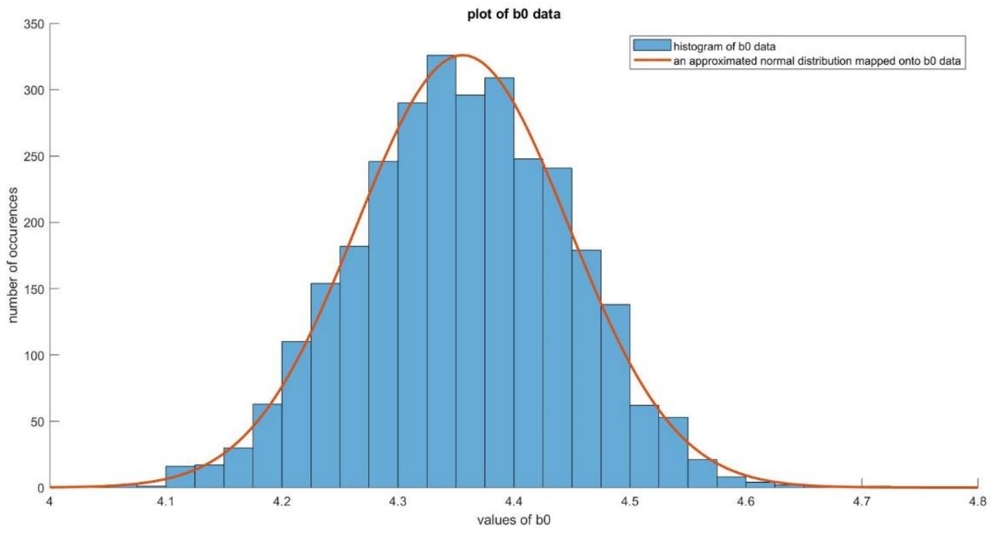
## Appendix A

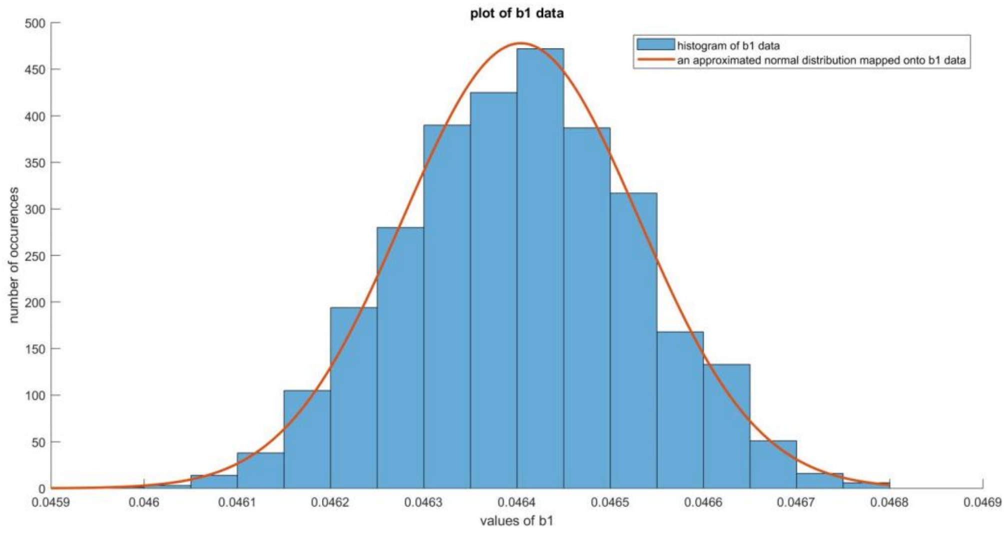
Codes below are for analysis of **historical** data for distributions of b0 and b1:

```
%for CalibAUP_Historical data
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(CalibAUP_Historical)]
    y_data=CalibAUP_Historical(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
%estimate percentage uncertainty for random error source b0
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)));
perc_s_b0=b0_std/b0_mean;
%estimate percentage uncertainty for systematic error source b1
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)));
perc_b_b1=b1_std/b1_mean;
figure(1)
title('plot of b1 data')
ylabel('number of occurences')
xlabel('values of b1')
hold on
histogram(b1,'BinWidth',0.00005)
fit_n_b1=makedist('Normal','mu',b1_mean,'sigma',b1_std);
x_b1=0.0459:0.000001:0.0468;
y_b1=pdf(fit_n_b1,x_b1)/6.6;
plot(x_b1,y_b1,'LineWidth',2)
legend('histogram of b1 data','an approximated normal distribution mapped onto b1
data')
hold off
figure(2)
title('plot of b0 data')
ylabel('number of occurences')
xlabel('values of b0')
hold on
histogram(b0,'BinWidth',0.025);
fit_n_b0=makedist('Normal','mu',b0_mean,'sigma',b0_std);
x_b0=4:0.001:4.8;
y_b0=pdf(fit_n_b0,x_b0)/0.0134;
plot(x_b0,y_b0,'LineWidth',2)
```

```
legend('histogram of b0 data','an approximated normal distribution mapped onto b0
data')
hold off
```

Plots below are for the histograms of b0 and b1 from historical data with their normal distribution mapping:

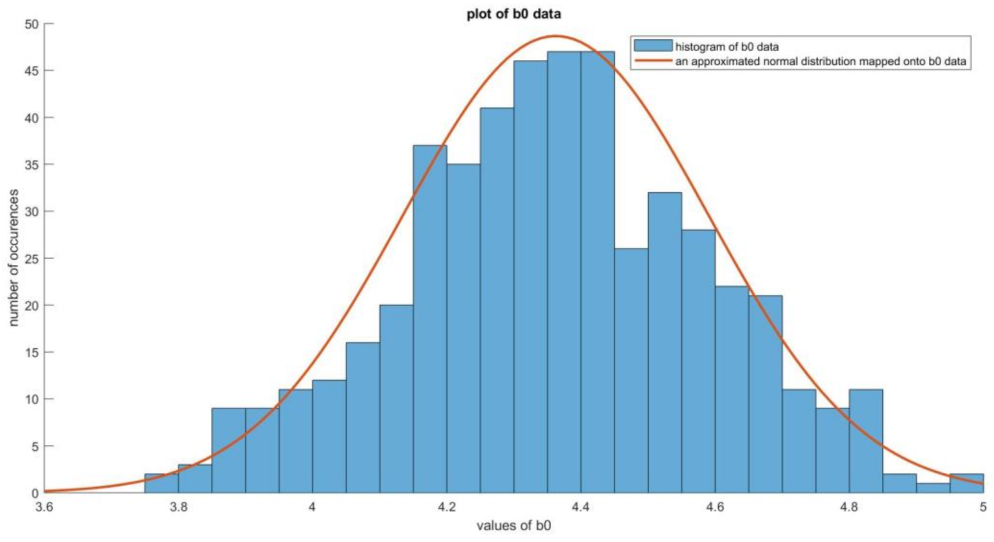


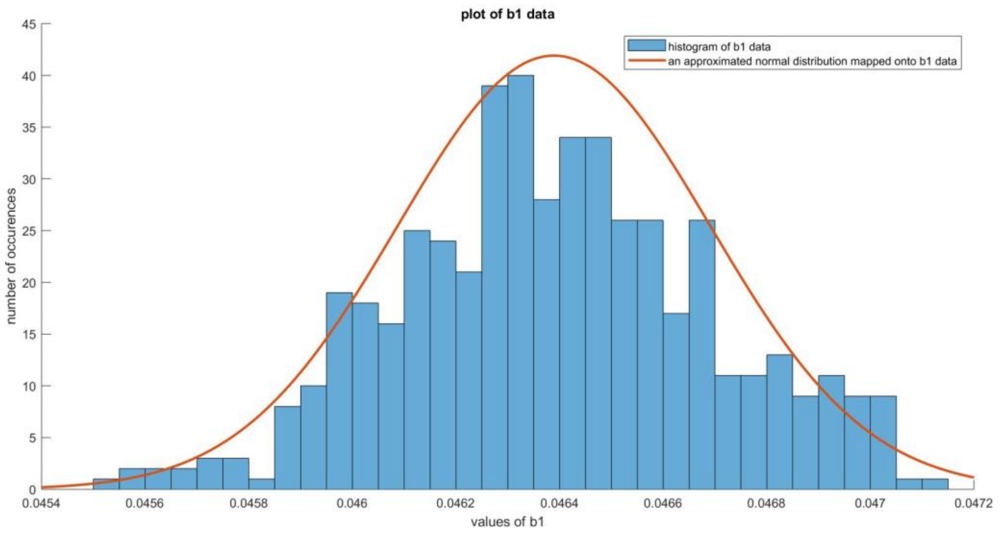


Codes below are for analysis of **2019** data for distributions of b0 and b1:

```
%for CalibAUP_2019 data
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(CalibAUP_2019)]
    y_data=CalibAUP_2019(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
%estimate percentage uncertainty for random error source b0
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)));
perc_s_b0=b0_std/b0_mean;
%estimate percentage uncertainty for systematic error source b1
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)));
perc_b_b1=b1_std/b1_mean;
figure(3)
title('plot of b1 data')
ylabel('number of occurences')
xlabel('values of b1')
hold on
histogram(b1,'BinWidth',0.00005)
fit_n_b1=makedist('Normal','mu',b1_mean,'sigma',b1_std);
x_b1=0.0454:0.00001:0.0472;
y_b1=pdf(fit_n_b1,x_b1)/31.5;
plot(x_b1,y_b1,'LineWidth',2)
legend('histogram of b1 data','an approximated normal distribution mapped onto b1
data')
hold off
figure(4)
title('plot of b0 data')
ylabel('number of occurences')
xlabel('values of b0')
hold on
histogram(b0,'BinWidth',0.05);
fit_n_b0=makedist('Normal','mu',b0_mean,'sigma',b0_std);
x_b0=3.6:0.01:5;
y_b0=pdf(fit_n_b0,x_b0)/0.035923;
plot(x_b0,y_b0,'LineWidth',2)
legend('histogram of b0 data','an approximated normal distribution mapped onto b0
data')
hold off
```

Plots below are for the histograms of b0 and b1 from 2019 data with their normal distribution mapping:





## Appendix B

Codes below are for **Q1:**

```
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(CalibAUP_Historical)]
    y_data=CalibAUP_Historical(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
% figure(1)
% histogram(b1,'BinWidth',0.00001)
% figure(2)
% histogram(b0,'BinWidth',0.01)
%estimate percentage uncertainty for random error source b0 from the
%sample, using unbiased estimator
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)-1));
perc_s_b0=b0_std/b0_mean
%estimate percentage uncertainty for systematic error source b1 from the sample,
using
%unbiased estimator
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)-1));
perc_b_b1=b1_std/b1_mean

%choose a best concentration value from CalibConc range
conc=480;          %the avergae of the historical concentration is used here
%calculate the best Abasample
A_best=48*b1_mean+b0_mean;

%start the MCM
%calculate the best value for DRE
vs_best=0.005;  %convert units: ml to l
vh_best=0.05;   %convert units: ml to l
C_best=0;
DRE_best=(((A_best-b0_mean)/b1_mean)*(vh_best/vs_best)+C_best)
%input percentage uncertainties for each elemnetal error sources
std_s_A=0.025*A_best/2;
std_s_vs=0.02*vs_best/2;
std_b_vs=0.02*vs_best/2;
std_s_vh=0.01*vh_best/2;
std_b_vh=0.01*vh_best/2;
std_s_C=0.02*DRE_best/2;
%convert std and mean to a and b for uniform distribution for std_s_A
syms a b
eqns=[0.5*(a+b)==0,(1/12)*(b-a)^2==std_s_A^2];
vars=[a,b];
[a,b]=solve(eqns,vars);
a_uni=double(a(1));
b_uni=double(b(1));
%convert std and mean to a and b for uniform distribution for std_s_vs
syms p q
eqns=[0.5*(p+q)==0,(1/12)*(p-q)^2==std_s_vs^2];
vars=[p,q];
```

```
[p,q]=solve(eqns,vars);
a_uni2=double(p(1));
b_uni2=double(q(1));
%convert std and mean to a, b and c for triangular distribution of std_s_vh
syms x y z
%assume symmetric triangular distribution
eqns=[(1/3)*(x+y+z)==0,z==0.5*(x+y),(1/18)*(x^2+y^2+z^2-x*y-y*z-x*z)==std_s_vh^2];
vars=[x,y,z];
[x,y,z]=solve(eqns,vars);
a_tri=double(x(1));
b_tri=double(y(1));
c_tri=double(z(1));
%input pds for all elemental sources of error, with uncertainty ranges
pd_A_s=makedist('uniform','lower',b_uni,'upper',a_uni);
%s_A=pd_A_s.random(1)
pd_b0_s=makedist('Normal','mu',0,'sigma',b0_std/2);
%s_b0=pd_b0_s.random(1)
pd_b1_b=makedist('Normal','mu',0,'sigma',b1_std/2);
%b_b1=pd_b1_b.random(1)
pd_vs_s=makedist('uniform','lower',b_uni2,'upper',a_uni2);
%s_vs=pd_vs_s.random(1)
pd_vs_b=makedist('Normal','mu',0,'sigma',std_b_vs);
%b_vs=pd_vs_b.random(1)
pd_vh_s=makedist('Triangular','a',b_tri,'b',c_tri,'c',a_tri);
%s_vh=pd_vh_s.random(1)
pd_vh_b=makedist('Normal','mu',0,'sigma',std_b_vh);
%b_vh=pd_vh_b.random(1)
pd_C_s=makedist('Normal','mu',0,'sigma',std_s_C);
%s_C=pd_C_s.random(1)

M=5000;   %define the M
DRE_sum=[];  %initilize the sample accumulation
smcm=[]; %initilize the standard deviation vector
smcm2=[];
%uncomment the below section to perform the manual MCM
% for i=[1:M]
%     %generate a random instance for each variable
%     A=A_best+pd_A_s.random(1);
%     b0_value=b0_mean+pd_b0_s.random(1);
%     b1_value=b1_mean+pd_b1_b.random(1);
%     vh=vh_best+pd_vh_s.random(1)+pd_vh_b.random(1);
%     vs=vs_best+pd_vs_s.random(1)+pd_vs_b.random(1);
%     C=C_best+pd_C_s.random(1);
%     %compute the DRE value
%     DRE(i)=(((A-b0_value)/b1_value)*(vh/vs)+C);
%     DRE_sum(end+1)=DRE(i);
%     DRE_std=std(DRE_sum);
%     %assume a 95% confidence interval
%     %smcm(i)=2*abs(DRE_mean-DRE_best);
%     smcm(i)=(2*DRE_std/DRE_best)*100;
%     smcm2(i)=2*DRE_std;
% end
%uncomment the below section to perform the adaptive/automatic MCM
M=50000;    %set the M to be very large
conv_thres=0.0001;  %set the convergence threshold
convergence=0;   %acts as a boolean, 0 means no convergence, 1 means convergence
counter=0;
for i=[1:M]
    counter=counter+1;
    %generate a random instance for each variable
    A=A_best+pd_A_s.random(1);
    b0_value=b0_mean+pd_b0_s.random(1);
    b1_value=b1_mean+pd_b1_b.random(1);
    vh=vh_best+pd_vh_s.random(1)+pd_vh_b.random(1);
    vs=vs_best+pd_vs_s.random(1)+pd_vs_b.random(1);
    C=C_best+pd_C_s.random(1);
    %compute the DRE value
    DRE(i)=(((A-b0_value)/b1_value)*(vh/vs)+C);
    DRE_sum(end+1)=DRE(i);
    DRE_std=std(DRE_sum);
    %assume a 95% CI
    smcm(i)=(2*DRE_std/DRE_best)*100;
    %get the average difference between current and previous %std tracing back 500
data points recorded
    if i>=501
        diff_sum=[];    %initialize the sum of differences list
        for j=[1:500]
            diff_sum(j)=abs(smcm(i-j+1)-smcm(i-j));
        end
        if mean(diff_sum)<=conv_thres
           convergence=1;
        end
    end
    %check if the average difference is less than the threshold value
    if convergence==1
        %if true, break the entire for loop
        break
    end
end

figure(1)
title('convergence of smcm for historical data')
xlabel('number of MCM runs(M)')
ylabel('the percentage value of smcm/%')
hold on
plot([1:counter],smcm,'r')
hold off
% figure(2)
% title('convergence of smcm for historical data')
% xlabel('number of MCM runs(M)')
% ylabel('the value of smcm/mgl^-1')
% hold on
% plot([1:M],smcm2,'r')
% hold off
```

Codes below are for **Q2:**

```
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(CalibAUP_2019)]
    y_data=CalibAUP_2019(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
% figure(1)
% histogram(b1,'BinWidth',0.00001)
% figure(2)
% histogram(b0,'BinWidth',0.01)
%estimate percentage uncertainty for random error source b0 from sample, using
unbiased
%estimator
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)-1));
perc_s_b0=b0_std/b0_mean
%estimate percentage uncertainty for systematic error source b1 from
%sample, using unbiased estimator
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)-1));
perc_b_b1=b1_std/b1_mean

%choose a best concentration value from CalibConc range
conc=480;          %the avergae of the historical concentration is used here
%calculate the best Abasample
A_best=48*b1_mean+b0_mean;

%start the MCM
%calculate the best value for DRE
vs_best=0.005;   %convert units: ml to l
vh_best=0.05;    %convert units: ml to l
C_best=0;
DRE_best=(((A_best-b0_mean)/b1_mean)*(vh_best/vs_best)+C_best)
%input percentage uncertainties for each elemnetal error sources
std_s_A=0.025*A_best/2;
std_s_vs=0.02*vs_best/2;
std_b_vs=0.02*vs_best/2;
std_s_vh=0.01*vh_best/2;
std_b_vh=0.01*vh_best/2;
std_s_C=0.02*DRE_best/2;
%convert std and mean to a and b for uniform distribution for std_s_A
syms a b
eqns=[0.5*(a+b)==0,(1/12)*(b-a)^2==std_s_A^2];
vars=[a,b];
[a,b]=solve(eqns,vars);
a_uni=double(a(1));
b_uni=double(b(1));
%convert std and mean to a and b for uniform distribution for std_s_vs
syms p q
eqns=[0.5*(p+q)==0,(1/12)*(p-q)^2==std_s_vs^2];
vars=[p,q];
[p,q]=solve(eqns,vars);
a_uni2=double(p(1));
b_uni2=double(q(1));
```

```
%convert std and mean to a, b and c for triangular distribution of std_s_vh
syms x y z
%assume symmetric triangular distribution
eqns=[(1/3)*(x+y+z)==0,z==0.5*(x+y),(1/18)*(x^2+y^2+z^2-x*y-y*z-x*z)==std_s_vh^2];
vars=[x,y,z];
[x,y,z]=solve(eqns,vars);
a_tri=double(x(1));
b_tri=double(y(1));
c_tri=double(z(1));
%input pds for all elemental sources of error, with uncertainty ranges
pd_A_s=makedist('uniform','lower',b_uni,'upper',a_uni);
%s_A=pd_A_s.random(1)
pd_b0_s=makedist('Normal','mu',0,'sigma',b0_std/2);
%s_b0=pd_b0_s.random(1)
pd_b1_b=makedist('Normal','mu',0,'sigma',b1_std/2);
%b_b1=pd_b1_b.random(1)
pd_vs_s=makedist('uniform','lower',b_uni2,'upper',a_uni2);
%s_vs=pd_vs_s.random(1)
pd_vs_b=makedist('Normal','mu',0,'sigma',std_b_vs);
%b_vs=pd_vs_b.random(1)
pd_vh_s=makedist('Triangular','a',b_tri,'b',c_tri,'c',a_tri);
%s_vh=pd_vh_s.random(1)
pd_vh_b=makedist('Normal','mu',0,'sigma',std_b_vh);
%b_vh=pd_vh_b.random(1)
pd_C_s=makedist('Normal','mu',0,'sigma',std_s_C);
%s_C=pd_C_s.random(1)

M=10000;   %define the M
DRE_sum=[];  %initilize the sample accumulation
smcm=[]; %initilize the standard deviation vector
smcm2=[];
%uncomment the below section to perform manual MCM
% for i=[1:M]
%     %generate a random instance for each variable
%     A=A_best+pd_A_s.random(1);
%     b0_value=b0_mean+pd_b0_s.random(1);
%     b1_value=b1_mean+pd_b1_b.random(1);
%     vh=vh_best+pd_vh_s.random(1)+pd_vh_b.random(1);
%     vs=vs_best+pd_vs_s.random(1)+pd_vs_b.random(1);
%     C=C_best+pd_C_s.random(1);
%     %compute the DRE value
%     DRE(i)=(((A-b0_value)/b1_value)*(vh/vs)+C);
%     DRE_sum(end+1)=DRE(i);
%     DRE_std=std(DRE_sum);
%     %assume a 95% confidence interval
%     %smcm(i)=2*abs(DRE_mean-DRE_best);
%     smcm(i)=(2*DRE_std/DRE_best)*100;
%     smcm2(i)=2*DRE_std;
% end

%uncomment the below section to perform the adaptive MCM
M=500000;    %set the M to be very large
conv_thres=0.0001;  %set the convergence threshold
convergence=0;   %acts as a boolean, 0 means no convergence, 1 means convergence
counter=0;
for i=[1:M]
    counter=counter+1;
    %generate a random instance for each variable
    A=A_best+pd_A_s.random(1);
    b0_value=b0_mean+pd_b0_s.random(1);
    b1_value=b1_mean+pd_b1_b.random(1);
    vh=vh_best+pd_vh_s.random(1)+pd_vh_b.random(1);
    vs=vs_best+pd_vs_s.random(1)+pd_vs_b.random(1);
    C=C_best+pd_C_s.random(1);
    %compute the DRE value
    DRE(i)=(((A-b0_value)/b1_value)*(vh/vs)+C);
    DRE_sum(end+1)=DRE(i);
    DRE_std=std(DRE_sum);
    %assume a 95% CI
    smcm(i)=(2*DRE_std/DRE_best)*100;
    %get the average difference between current and previous %std tracing back 500
data points recorded
    if i>=501
        diff_sum=[];    %initialize the sum of differences list
        for j=[1:500]
            diff_sum(j)=abs(smcm(i-j+1)-smcm(i-j));
        end
        if mean(diff_sum)<=conv_thres
           convergence=1;
        end
    end
    %check if the average difference is less than the threshold value
    if convergence==1
        %if true, break the entire for loop
        break
    end
end

figure(1)
title('convergence of smcm for 2019 data')
xlabel('number of MCM runs(M)')
ylabel('the percentage value of smcm/%')
hold on
plot([1:counter],smcm,'r')
hold off
% figure(2)
% title('convergence of smcm for 2019 data')
% xlabel('number of MCM runs(M)')
% ylabel('the value of smcm/mgl^-1')
% hold on
% plot([1:M],smcm2,'r')
% hold off
```

## Appendix C

Codes below are for **Q3:**

```
%join two datasets horizontally
merged_data=[CalibAUP_Historical CalibAUP_2019];
%merge completed
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(merged_data)]
    y_data=merged_data(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
% figure(1)
% histogram(b1,'BinWidth',0.00001)
% figure(2)
% histogram(b0,'BinWidth',0.01)
%estimate percentage uncertainty for random error source b0 from sample,
%using unbiased estimator
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)-1));
perc_s_b0=b0_std/b0_mean
%estimate percentage uncertainty for systematic error source b1 from sample
%using unbiased estimator
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)-1));
perc_b_b1=b1_std/b1_mean

%choose a best concentration value from CalibConc range
%conc=600;          %the avergae of the historical concentration is used here
conc_val=[200,400,480,600,800];
col=['r','b','y','g','k'];
counter=0;
smcm_list=[];
for conc=conc_val
    counter=counter+1;
    %calculate the best Abasample
    A_best=(conc/10)*b1_mean+b0_mean;

    %start the MCM
    %calculate the best value for DRE
    vs_best=0.005;  %convert units: ml to l
    vh_best=0.05;   %convert units: ml to l
    C_best=0;
    DRE_best=(((A_best-b0_mean)/b1_mean)*(vh_best/vs_best)+C_best)
    %input percentage uncertainties for each elemnetal error sources
    std_s_A=0.025*A_best/2;
    std_s_vs=0.02*vs_best/2;
    std_b_vs=0.02*vs_best/2;
    std_s_vh=0.01*vh_best/2;
    std_b_vh=0.01*vh_best/2;
    std_s_C=0.02*DRE_best/2;
    %convert std and mean to a and b for uniform distribution for std_s_A
    syms a b
    eqns=[0.5*(a+b)==0,(1/12)*(b-a)^2==std_s_A^2];
    vars=[a,b];
    [a,b]=solve(eqns,vars);
    a_uni=double(a(1));
    b_uni=double(b(1));
    %convert std and mean to a and b for uniform distribution for std_s_vs
    syms p q
    eqns=[0.5*(p+q)==0,(1/12)*(p-q)^2==std_s_vs^2];
    vars=[p,q];
    [p,q]=solve(eqns,vars);
    a_uni2=double(p(1));
    b_uni2=double(q(1));
    %convert std and mean to a, b and c for triangular distribution of std_s_vh
    syms x y z
    %assume symmetric triangular distribution
    eqns=[(1/3)*(x+y+z)==0,z==0.5*(x+y),(1/18)*(x^2+y^2+z^2-x*y-y*z-
x*z)==std_s_vh^2];
    vars=[x,y,z];
    [x,y,z]=solve(eqns,vars);
    a_tri=double(x(1));
    b_tri=double(y(1));
    c_tri=double(z(1));
    %input pds for all elemental sources of error, with uncertainty ranges
    pd_A_s=makedist('uniform','lower',b_uni,'upper',a_uni);
    %s_A=pd_A_s.random(1)
    pd_b0_s=makedist('Normal','mu',0,'sigma',b0_std/2);
    %s_b0=pd_b0_s.random(1)
    pd_b1_b=makedist('Normal','mu',0,'sigma',b1_std/2);
    %b_b1=pd_b1_b.random(1)
    pd_vs_s=makedist('uniform','lower',b_uni2,'upper',a_uni2);
    %s_vs=pd_vs_s.random(1)
    pd_vs_b=makedist('Normal','mu',0,'sigma',std_b_vs);
    %b_vs=pd_vs_b.random(1)
    pd_vh_s=makedist('Triangular','a',b_tri,'b',c_tri,'c',a_tri);
    %s_vh=pd_vh_s.random(1)
    pd_vh_b=makedist('Normal','mu',0,'sigma',std_b_vh);
    %b_vh=pd_vh_b.random(1)
    pd_C_s=makedist('Normal','mu',0,'sigma',std_s_C);
    %s_C=pd_C_s.random(1)

    M=5000;   %define the M
    DRE_sum=[];  %initilize the sample accumulation
    smcm=[]; %initilize the standard deviation vector
    for i=[1:M]
        %generate a random instance for each variable
        A=A_best+pd_A_s.random(1);
        b0_value=b0_mean+pd_b0_s.random(1);
        b1_value=b1_mean+pd_b1_b.random(1);
        vh=vh_best+pd_vh_s.random(1)+pd_vh_b.random(1);
        vs=vs_best+pd_vs_s.random(1)+pd_vs_b.random(1);
        C=C_best+pd_C_s.random(1);
        %compute the DRE value
        DRE(i)=(((A-b0_value)/b1_value)*(vh/vs)+C);
        DRE_sum(end+1)=DRE(i);
        DRE_std=std(DRE_sum);
        %assume a 95% confidence interval
        smcm(i)=2*100*DRE_std/DRE_best;
    end
    smcm_list(end+1)=smcm(end)
    figure(1)
    title('Expanded uncertainties at different BA concentrations')
    ylabel('expanded uncertainty/% value')
    xlabel('number of MCM runs/M')

plot([1:M],smcm,col(counter),'DisplayName',strcat('concentration:',string(conc)))
    hold on
end
legend('-DynamicLegend')
hold off
```

## Appendix D

Codes below are for **Q4:**

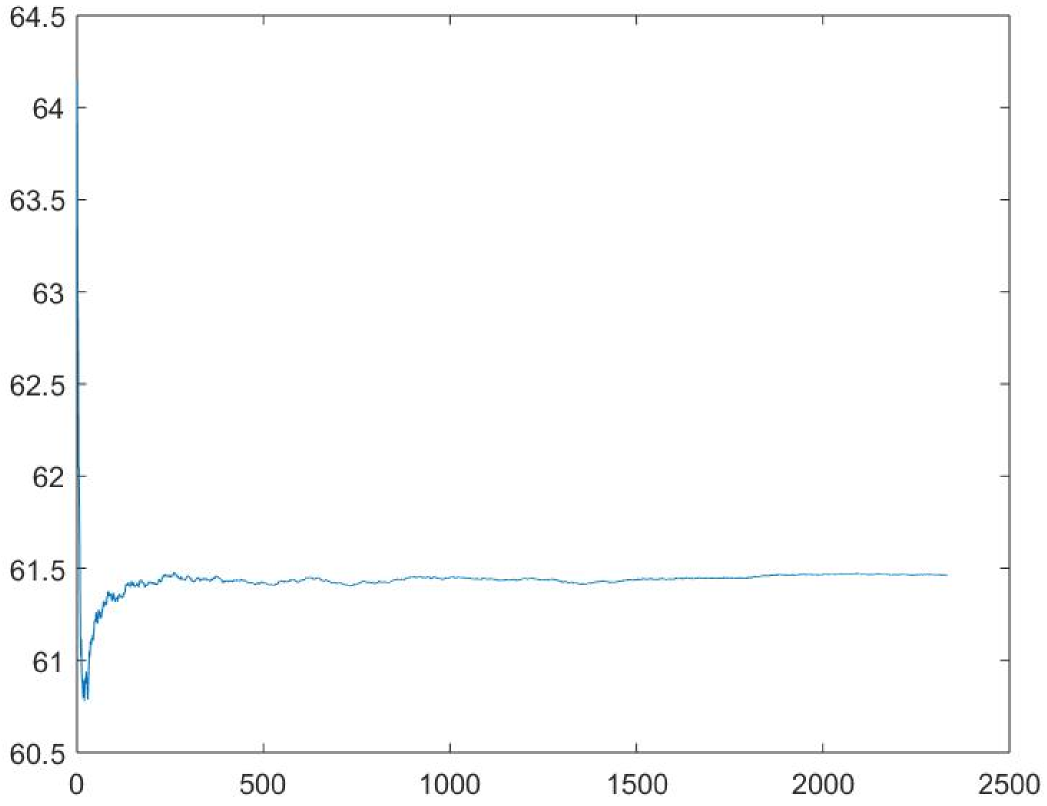
```
%get values and variations for b0 and b1 from merged historical and 2019
%data for use in Q4
%join two datasets horizontally
merged_data=[CalibAUP_Historical CalibAUP_2019];
%merge completed
x_data=CalibConc;
x_mean=mean(x_data);
for i=[1:length(merged_data)]
    y_data=merged_data(:,i);
    y_mean=mean(y_data);
    %perform linear regression
    Sxx=transpose(x_data-x_mean)*(x_data-x_mean);
    b1(i)=(transpose(x_data-x_mean)*(y_data-y_mean))/Sxx;  %slope
    b0(i)=y_mean-b1(i)*x_mean;  %intercept
end
%estimate percentage uncertainty for random error source b0 from sample,
%using unbiased estimator
sum_b0=0;
b0_mean=sum(b0)/length(b0);
for i=[1:length(b0)]
    sum_b0=sum_b0+(b0(i)-b0_mean)^2;
end
b0_std=sqrt(sum_b0/(length(b0)-1));
%estimate percentage uncertainty for systematic error source b1 from sample
%using unbiased estimator
sum_b1=0;
b1_mean=sum(b1)/length(b1);
for i=[1:length(b1)]
    sum_b1=sum_b1+(b1(i)-b1_mean)^2;
end
b1_std=sqrt(sum_b1/(length(b1)-1));

%choose a best concentration value from CalibConc range
conc=480;           %the avergae of the historical concentration is used here

%calculate mean and std for each variable
A_mean=(conc/10)*b1_mean+b0_mean;
A_std=0.025*A_mean/2;
vs_mean=0.005;  %convert units: ml to l
vs_std=sqrt((0.02*vs_mean/2)^2+(0.02*vs_mean/2)^2); %to get the combined standard
uncertainty uc
vh_mean=0.05; %convert units: ml to l
vh_std=sqrt((0.01*vh_mean/2)^2+(0.01*vh_mean/2)^2); %to get the combined standard
uncertainty uc
C_mean=0;
DRE_mean=(((A_mean-b0_mean)/b1_mean)*(vh_mean/vs_mean)+C_mean);  %get the DRE's
output mean
C_std=0.02*DRE_mean/2;

%get the lower and upper bounds for all variables
A_lower=conv_b_uni(A_mean,A_std);
A_upper=conv_a_uni(A_mean,A_std);
b0_lower=conv_b_uni(b0_mean,b0_std);
b0_upper=conv_a_uni(b0_mean,b0_std);
b1_lower=conv_b_uni(b1_mean,b1_std);
b1_upper=conv_a_uni(b1_mean,b1_std);
vs_lower=conv_b_uni(vs_mean,vs_std);
vs_upper=conv_a_uni(vs_mean,vs_std);
vh_lower=conv_b_uni(vh_mean,vh_std);
vh_upper=conv_a_uni(vh_mean,vh_std);
C_lower=conv_b_uni(C_mean,C_std);
C_upper=conv_a_uni(C_mean,C_std);
```

```
%initialize the parameters for Elemental Effect Analysis
k=6;   %number of inputs
r=3000; %number of trajectories
p=4;  %p is even
delta=p/(2*(p-1));
%initialize the discrete p level from 0 to 1
p_level=[0:1/(p-1):1-delta];
%initialize a matrix to store all EEs for each variable across r runs
EE_matrix=zeros(r,2*k);
%initialize matrices to store the statistics of each variable across r runs
stat_mean_matrix=zeros(r,2*k);  %contain the mean_star and mean for each vairable's
EE
stat_sigma2_matrix=zeros(r,k);  %contain the variance for each variable's EE
%initialize a matrix x_y_points_accumulation to store all variable's points
%and their corresponding outputs across r runs
x_y_points_accumulation=[];
convergence=0;     %for automatic ocnvergence, acts as a boolean, 0 is false, 1 is
true
for run=[1:r]
    %initialize the starting points x*
    x_star=zeros(1,k);
    for j=[1:k]
        idx=randperm(length(p_level),1);
        x_star(j)=p_level(idx);
    end
    %set up sampling matrices
    B=tril(ones(k+1,k),-1);
    J=ones(k+1,k);
    BmJ=2*B-J;
    D=diag(rand(k,1));
    D(D>0.5)=1;
    D(D<=0.5&D>0)=-1;
    D_star=D;
    BmJtD=BmJ*D_star;
    BmJtDpJ=BmJtD+J;
    BmJtDpJtD2=BmJtDpJ*(delta/2);
    P=eye(k);
    P_star=P(randperm(k),:);
    B_s=BmJtDpJtD2*P_star;
    %get the order of trajectory points
    order_matrix=zeros(k,2);
    for i=[1:k]
        for ord=[1:k]
            e=zeros(1,k);
            e(ord)=1;
            delta_plus=B_s(i,:)+delta*e;
            delta_minus=B_s(i,:)-delta*e;
            if B_s(i+1,:)==delta_plus
                %assign the variable index to the row index of order_matrix
                %assign the row index of the smaller row in B to the first column
of order_matrix
                %assign the row index of the larger row in B to the second column
of order_matrix
                order_matrix(ord,1)=i;
                order_matrix(ord,2)=i+1;
            end
            if B_s(i+1,:)==delta_minus
                order_matrix(ord,1)=i+1;
                order_matrix(ord,2)=i;
            end
        end
    end
    %get B*
    Jtx_star=zeros(k+1,k);
    for i=[1:k+1]
        Jtx_star(i,:)=x_star;
    end
    B_star=Jtx_star+B_s;
    %scale each variable of B* from upper and lower bounds of the variable's
distribution
    B_star_scaled=zeros(k+1,k);
    %use helper function uni_scaler
    %1st column of B* is variable A
    for i=[1:k+1]
        B_star_scaled(i,1)=uni_scaler(B_star(i,1),A_lower,A_upper);
    end
    %2nd column is variable b0
    for i=[1:k+1]
        B_star_scaled(i,2)=uni_scaler(B_star(i,2),b0_lower,b0_upper);
    end
    %3rd column is variable b1
    for i=[1:k+1]
        B_star_scaled(i,3)=uni_scaler(B_star(i,3),b1_lower,b1_upper);
    end
    %4th column is variable vs
    for i=[1:k+1]
        B_star_scaled(i,4)=uni_scaler(B_star(i,4),vs_lower,vs_upper);
    end
    %5th column is variable vh
    for i=[1:k+1]
        B_star_scaled(i,5)=uni_scaler(B_star(i,5),vh_lower,vh_upper);
    end
    %6th column is variable C
    for i=[1:k+1]
        B_star_scaled(i,6)=uni_scaler(B_star(i,6),C_lower,C_upper);
    end
    %perform model evaluation with helper function model_eva
    B_result=horzcat(B_star_scaled,zeros(k+1,1));
    for i=[1:k+1]

B_result(i,end)=model_eva(B_star_scaled(i,1),B_star_scaled(i,2),B_star_scaled(i,3),
B_star_scaled(i,4),B_star_scaled(i,5),B_star_scaled(i,6));
    end
    %calculate the elemental effect EE for each variable
    %for 1st variable A
    EE_A=(B_result(order_matrix(1,2),end)-B_result(order_matrix(1,1),end))/delta;
    %for 2nd variable b0
    EE_b0=(B_result(order_matrix(2,2),end)-B_result(order_matrix(2,1),end))/delta;
    %for 3rd variable b1
    EE_b1=(B_result(order_matrix(3,2),end)-B_result(order_matrix(3,1),end))/delta;
    %for 4th variable vs
    EE_vs=(B_result(order_matrix(4,2),end)-B_result(order_matrix(4,1),end))/delta;
    %for 5th varaible vh
    EE_vh=(B_result(order_matrix(5,2),end)-B_result(order_matrix(5,1),end))/delta;
    %for 6th variable C
    EE_C=(B_result(order_matrix(6,2),end)-B_result(order_matrix(6,1),end))/delta;
    %store all EEs into an array, where each variable takes up two columns, one for
EE, one for absolute EE
    EE_array=[];

EE_array=[EE_A,abs(EE_A),EE_b0,abs(EE_b0),EE_b1,abs(EE_b1),EE_vs,abs(EE_vs),EE_vh,a
bs(EE_vh),EE_C,abs(EE_C)];
    EE_matrix(run,:)=EE_array;
    for j=[2:2:2*k]
        %store mean_star
        stat_mean_matrix(run,j/2)=mean(EE_matrix(1:run,j));
    end
    for j=[1:2:2*k]
        %store mean
        stat_mean_matrix(run,k+(j+1)/2)=mean(EE_matrix(1:run,j));
    end
    for j=[1:k]
        %use helper function var_EE to calculate the variance
        stat_sigma2_matrix(run,j)=var_EE(EE_matrix(1:run,2*j-1));
    end
    %accumulate the B_result data into a matrix for variable-output scatter plot
later
    x_y_points_accumulation=vertcat(x_y_points_accumulation,B_result);
    %uncomment the below section to use the forced convergence to find the
    %number of runs needed
%     if run>=501
%         conv_thres=0.00001;
%         diff_sum=[];    %initialize the sum of differences list
%         for back=[1:500]
%             diff_sum(j)=abs(stat_mean_matrix(run-back+1,1)-stat_mean_matrix(run-
back,1));
%         end
%         if mean(diff_sum)<=conv_thres
%            convergence=1;
%         end
%     end
%     %check if the average difference is less than the threshold value
%     if convergence==1
%         %if true, break the entire for loop
%         break
%     end
end
VARIABLES=categorical({'Abasample','b0','b1','vs','vh','deltaC'});
MEAN_STATS=[stat_mean_matrix(end,1),stat_mean_matrix(end,2),stat_mean_matrix(end,3)
,stat_mean_matrix(end,4),stat_mean_matrix(end,5),stat_mean_matrix(end,6)];
VAR_STATS=[stat_sigma2_matrix(end,1),stat_sigma2_matrix(end,2),stat_sigma2_matrix(e
nd,3),stat_sigma2_matrix(end,4),stat_sigma2_matrix(end,5),stat_sigma2_matrix(end,6)
];
figure(1)
title('u* for each variable')
hold on
bar(VARIABLES,MEAN_STATS,0.6,'r')
xlabel('Factors')
ylabel('u*')
hold off
figure(2)
title('variance for each variable')
hold on
bar(VARIABLES,VAR_STATS,0.6,'Y')
xlabel('Factors')
ylabel('variance')
hold off
figure(3)
title('plot of u* against variance')
xlabel('u*')
ylabel('variance')
hold on
scatter(stat_mean_matrix(end,1),stat_sigma2_matrix(end,1),'filled')
scatter(stat_mean_matrix(end,2),stat_sigma2_matrix(end,2),'filled')
scatter(stat_mean_matrix(end,3),stat_sigma2_matrix(end,3),'filled')
scatter(stat_mean_matrix(end,4),stat_sigma2_matrix(end,4),'filled')
scatter(stat_mean_matrix(end,5),stat_sigma2_matrix(end,5),'filled')
scatter(stat_mean_matrix(end,6),stat_sigma2_matrix(end,6),'filled')
legend('Abasample','b0','b1','vs','vh','deltaC')
hold off

figure(4)
subplot(2,3,1)
scatter(x_y_points_accumulation(1:7000,1),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('Abasample')
ylabel('Cbasample')
subplot(2,3,2)
scatter(x_y_points_accumulation(1:7000,2),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('b0')
ylabel('Cbasample')
subplot(2,3,3)
```

```
scatter(x_y_points_accumulation(1:7000,3),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('b1')
ylabel('Cbasample')
subplot(2,3,4)
scatter(x_y_points_accumulation(1:7000,4),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('vs')
ylabel('Cbasample')
subplot(2,3,5)
scatter(x_y_points_accumulation(1:7000,5),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('vh')
ylabel('Cbasample')
subplot(2,3,6)
scatter(x_y_points_accumulation(1:7000,6),x_y_points_accumulation(1:7000,end),1.5,'
filled')
title('deltaC')
ylabel('Cbasample')

%plot([1:run],stat_mean_matrix(1:run,1))
figure(5)
subplot(2,3,1)
plot([1:run],stat_mean_matrix(1:run,1))
title('convergence of u* for A')
xlabel('r/runs')
ylabel('u*')
subplot(2,3,2)
plot([1:run],stat_mean_matrix(1:run,2))
title('convergence of u* for b0')
xlabel('r/runs')
ylabel('u*')
subplot(2,3,3)
plot([1:run],stat_mean_matrix(1:run,3))
title('convergence of u* for b1')
xlabel('r/runs')
ylabel('u*')
subplot(2,3,4)
plot([1:run],stat_mean_matrix(1:run,4))
title('convergence of u* for vs')
xlabel('r/runs')
ylabel('u*')
subplot(2,3,5)
plot([1:run],stat_mean_matrix(1:run,5))
title('convergence of u* for vh')
xlabel('r/runs')
ylabel('u*')
subplot(2,3,6)
plot([1:run],stat_mean_matrix(1:run,6))
title('convergence of u* for C')
xlabel('r/runs')
ylabel('u*')

figure(6)
subplot(2,3,1)
plot([1:run],stat_sigma2_matrix(1:run,1))
title('convergence of variance for A')
xlabel('r/runs')
ylabel('variance')
subplot(2,3,2)
plot([1:run],stat_sigma2_matrix(1:run,2))
title('convergence of variance for b0')
xlabel('r/runs')
ylabel('variance')
subplot(2,3,3)
plot([1:run],stat_sigma2_matrix(1:run,3))
title('convergence of variance for b1')
xlabel('r/runs')
```

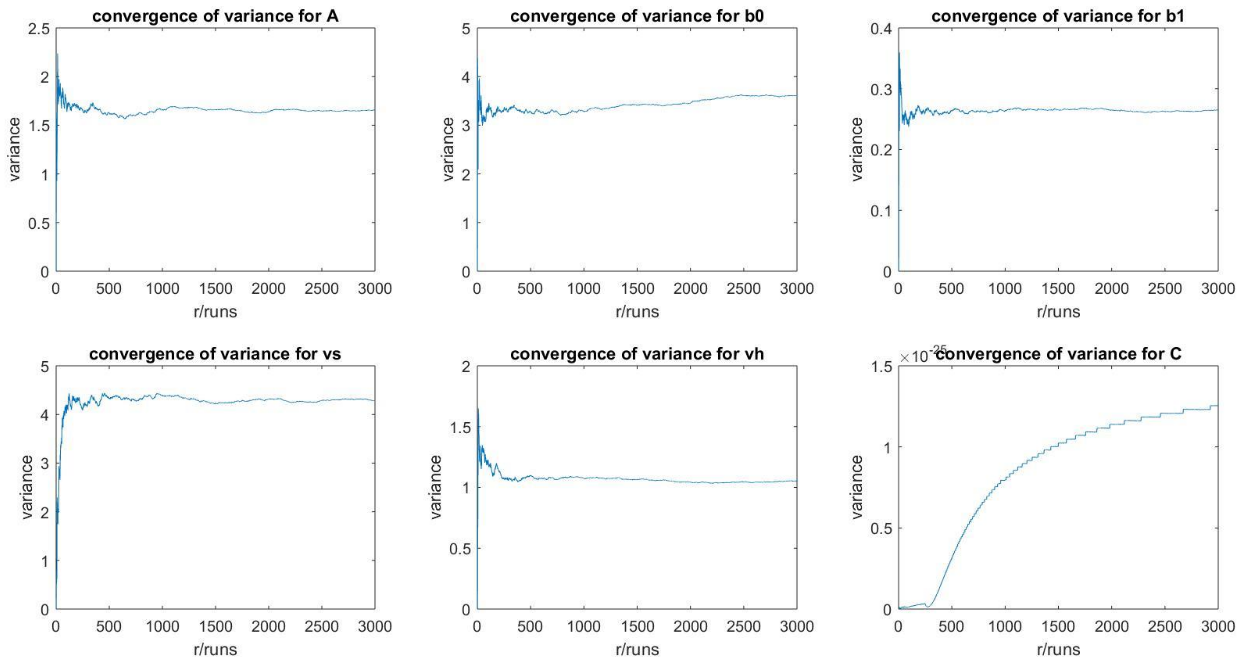
```
ylabel('variance')
subplot(2,3,4)
plot([1:run],stat_sigma2_matrix(1:run,4))
title('convergence of variance for vs')
xlabel('r/runs')
ylabel('variance')
subplot(2,3,5)
plot([1:run],stat_sigma2_matrix(1:run,5))
title('convergence of variance for vh')
xlabel('r/runs')
ylabel('variance')
subplot(2,3,6)
plot([1:run],stat_sigma2_matrix(1:run,6))
title('convergence of variance for C')
xlabel('r/runs')
ylabel('variance')

%helper functions
%define the helper function conv_a_uni to convert mean and std to upper (a) bound
for uniform distribution
function [a_uni]=conv_a_uni(mean, std)
syms a b
eqns=[0.5*(a+b)==mean,(1/12)*(b-a)^2==std^2];
vars=[a,b];
[a,b]=solve(eqns,vars);
a_uni=double(a(1));
end
%define the helper function conv_b_uni to convert mean and std to lower (b) bound
for uniform distribution
function [b_uni]=conv_b_uni(mean, std)
syms a b
eqns=[0.5*(a+b)==mean,(1/12)*(b-a)^2==std^2];
vars=[a,b];
[a,b]=solve(eqns,vars);
b_uni=double(b(1));
end
%define helper function uni_scaler to scale 0-1 uniform dist to b-a bound
function [X]=uni_scaler(U,lower,upper)
X=U*(upper-lower)+lower;
end
%define helper function model_eva to use sampled points to get the DRE result
function [Y]=model_eva(A,b0,b1,vs,vh,C)
Y=(((A-b0)/b1)*(vh/vs)+C);
end
%define helper function var_EE to calculate the variance of each variable's
%EE, according to the sensitivity measure formula
function [var]=var_EE(column_vector)
mean=sum(column_vector)/length(column_vector);
sum_squares=0;
for i=[1:length(column_vector)]
    sum_squares=sum_squares+(column_vector(i)-mean)^2;
end
var=sum_squares/(length(column_vector-1));
end
```

Plot below is the first time run with automatic convergence to find the r needed for EEM:

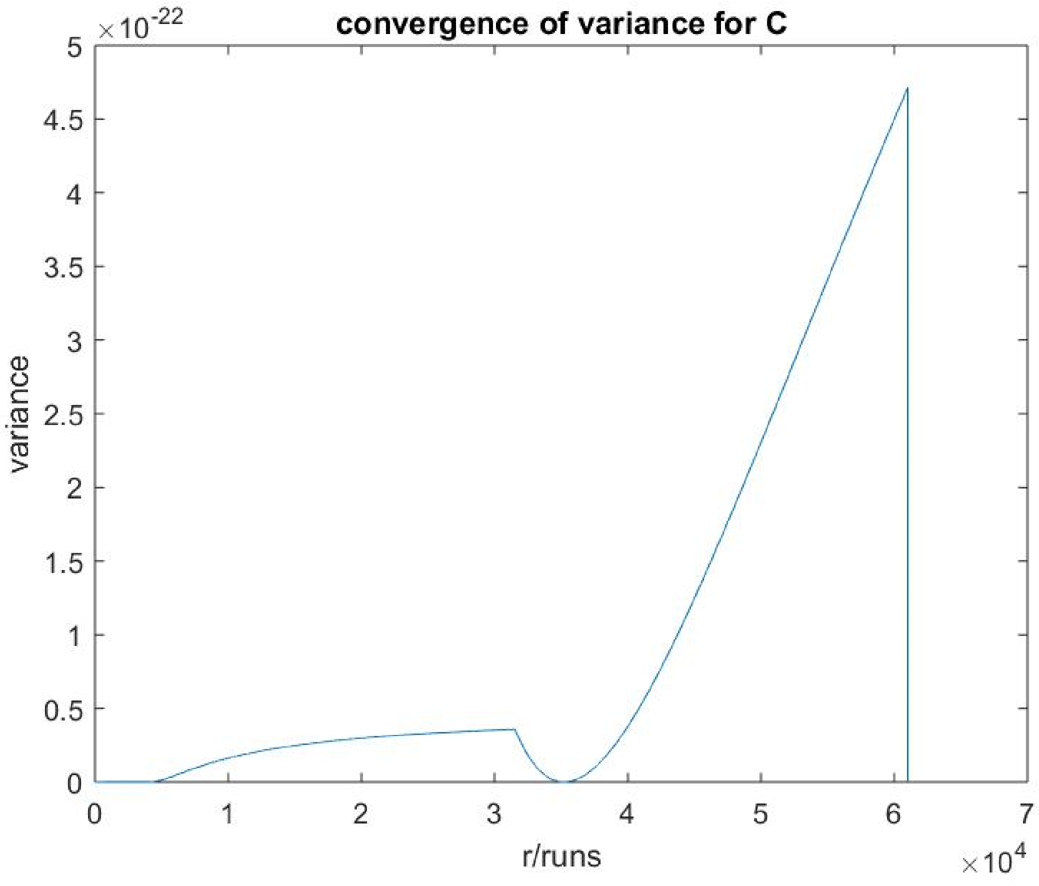


The auto-convergence stops at r=2333 runs, thus it is decided that parameter r would be 3000 runs to be sufficient for convergence.

Plot below is the convergence proof for the other sensitivity metric $\sigma^2$, after r=3000 runs:



From the above figure, all variables' $\sigma^2$ seems to be able to converge after 3000 runs, except for $\sigma^2$ of deltaC. The convergence for deltaC is made so difficult mainly because of the extremely low value (10 to the power of -25) of its $\sigma^2$. Thus it is anticipated that a very large number of runs has to be set in order to produce a convergence for deltaC. The automatic convergence used so many times before is used again to try to determine the number of runs r necessary to make deltaC converge. A plot of the result is shown below:



The software crashes before roughly 60000 runs and as can be seen, the number needed to converge the very small value of $\sigma^2$ for deltaC will be at least somewhere above 100000, which is just not economical. With such r, the total computing time drastically increases, and all other variables have long converged before it ends. Therefore it is reasonable to keep r=3000 to focus on the convergence of other variables and keep the EEM efficient as it is designed to be.

*End of the entire coursework*