Visual basic for design assignment – the residence time distribution in a packed column

Bethany Mulliner

**Contents**

|  |  |
| --- | --- |
| Topic | Page Number |
| Title Page | 1 |
| Contents Page | 2 |
| Introduction | 3 |
| Total Amount of Tracer Used | 3 |
| Mean Residence Time | 5 |
| Variance | 7 |
| Appendix I | 10 |
| References | 11 |

**Introduction**

The residence time distribution is an important characteristic for continuously operated equipment in the field of chemical engineering. The residence time distribution tells us what the probability is that a small amount of liquid requires a specific amount of time to pass through the equipment. In order to measure the residence time distribution (RTD), the equipment is assumed to operate at steady state, and a minute amount of tracer is injected into the feed stream. By measuring the exit stream, the time taken for the tracer to pass through the equipment can be determined, usually in the form of a distribution of concentration over time, C(t).

The aforementioned experiment was performed using a small packed column. The tracer was a salt solution and was detected using conductance measurements.

**Graph 1 – Results**

Using **equation 1:** , where k = calibration factor = 500M/S (M=molL-1, S=Siemens), a graph of conductance, G(t), could be plotted against time, t.

The task was then to calculate the total amount of tracer used; the mean residence time; and the variance, using Visual Basic functions.

(Bock, n.d.)

The data provided was a table of times, t, in seconds with corresponding conductivity values, G(t), in microSiemens.

For ease of reading, the individual functions for the code have been given in their own sections and a complete version has been provided in **Appendix I**.

Note: Line numbers for code are obtained using Notepad++. (Ho, 2016)

**Total Amount of Tracer Used**

In order to calculate the total amount of tracer, **equation 2** was provided:

(Bock, n.d.)

Where:

n – Total amount of tracer used (mol)

– Volumetric flow rate through the column = 3x10-2 Ls-1

C(t) – Concentration (molL-1)

t – Time (s)

A limitation of this equation is that it is an integral with an upper limit of infinity, which cannot be experimentally measured to, therefore, the equation is instead integrated between the minimum and maximum times given by the data.

**Concept**

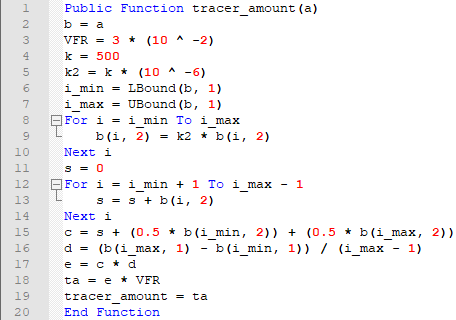
Firstly, the user input was defined as a 2D array, Next the VFR was stored, and the k stored and converted to the appropriate units. The upper and lower bounds of the rows of the array were then found and all G(t) values were converted to C(t) values. Next, the integral in equation 2 was approximated using the trapezium rule as follows:

**Equation 3**

**Equation 4** (Bock, n.d.)

The approximation was then multiplied by the volumetric flow rate to obtain the amount of tracer used.

**Function 1 – tracer\_amount**



**Methodology**

Firstly, we define the Public Function ‘**tracer\_amount**’, taking an argument of ‘**a**’, which is a 2D array of unlimited size. In order to make this a ‘proper’ array, we must call a new variable ‘**b**’ to take the values of the array ‘**a**’.

From the brief, we are told that the volumetric flow rate through the column (**VFR**) is 3x10-2 Ls-1. This value is therefore stored in the code as ‘**VFR**’. We are also told that the calibration factor, **k**, has a value of 500M/S. In order to use this to convert G(t) to C(t), we must convert the conversion factor from M/S to M/μS by multiplying it by 10-6, giving us ‘**k2**’.

Next, we define the upper and lower bounds of the number of rows of the array using the upper and lower bound functions. These are then stored as ‘**i\_min**’ and ‘**i\_max**’ respectively. Then, the conversion factor is applied to the G(t) to transform it to C(t). This involves a For loop in which between the upper and lower bounds of the rows of the array, the conversion factor is multiplied by C(t), repeating until all values are converted.

Now that the conversions are complete, the code enters into the integral, which is approximated using the trapezium rule:

This is broken down into several smaller steps:

1. =

The sum, ‘**s**’, is then defined as initially being equal to 0. The code then enters another For loop in which it sums all the values in the now C(t) column between the first and last values (exclusive). This final value is then stored as ‘**s**’. The first and final values in the C(t) column are then divided by two and added to ‘**s**’ to give a new variable, ‘**c**’.

Then, the first value in the t column is subtracted from the last value and the result divided by the number of rows in the array minus one to give Δx, stored as ‘**d**’. Finally, to complete the trapezium rule approximation of the integral, ‘**c**’ and ‘**d**’ are multiplied together, which is stored as ‘**e**’.

In order to complete the function based on **equation 2**, ‘**e**’ is then multiplied by the volumetric flow rate ‘**VFR**’, to give a result for the total amount of tracer used.

**Mean Residence Time**

In order to calculate the mean residence time, **equation 5** was provided:

(Bock, n.d.)

Where:

- Mean residence time (s)

t – Time (s)

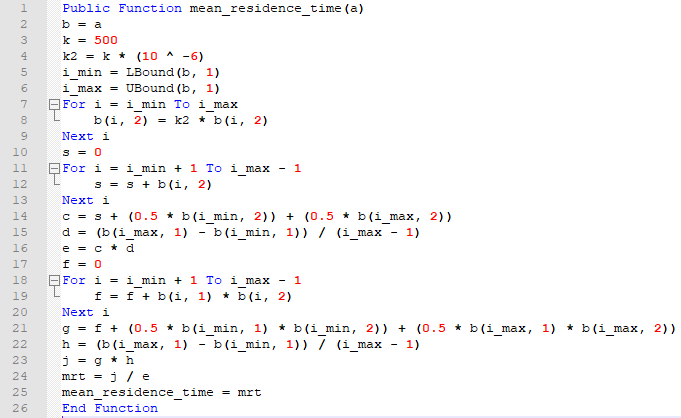
C(t) – Concentration (molL-1)

Once again, a limitation of this equation is that there are two integrals with an upper limit of infinity, which cannot be experimentally measured to, therefore, the integrals are instead integrated between the minimum and maximum times given by the data.

**Concept**

The first 16 lines of this function are almost identical to **tracer\_amount**, as they deal with , however, the value of the volumetric flow rate is left out, as it is not needed for this function. Then, the integration on the numerator is dealt with. The structure of the code is identical, but the code multiplies all values of C(t) by their corresponding t to obtain tC(t). These values can then be used in the trapezium rule to obtain an approximation for the integral. To obtain the mean residence time, we then divide the approximation for by the approximation for .

**Function 2 – mean\_residence\_time**



**Methodology**

Please refer to the **methodology** for **tracer\_amount** for an explanation of the first 16 lines of code, as they are essentially the same, and for an overall explanation of how to apply the trapezium rule in Visual Basic to approximate definite integrals.

We pick up the story at line 17, where we define a new variable ‘**f**’, as having a value of 0. As can be observed, the overall format is very similar to the trapezium rule for , but with some key differences. For the steps for the trapezium rule (defined in the **methodology** for **tracer\_amount**), we replace all instances of ‘f(x)’ with ‘g(x)’, and let:

I.E:

Therefore:

1. =

Once the approximation for the numerator is obtained, we then divide the approximation for by the approximation for to give us the mean residence time.

**Variance**

In order to calculate variance, **equation 6** was provided:

(Bock, n.d.)

Where:

σ - Variance

- Mean residence time (s)

t – Time (s)

C(t) – Concentration (molL-1)

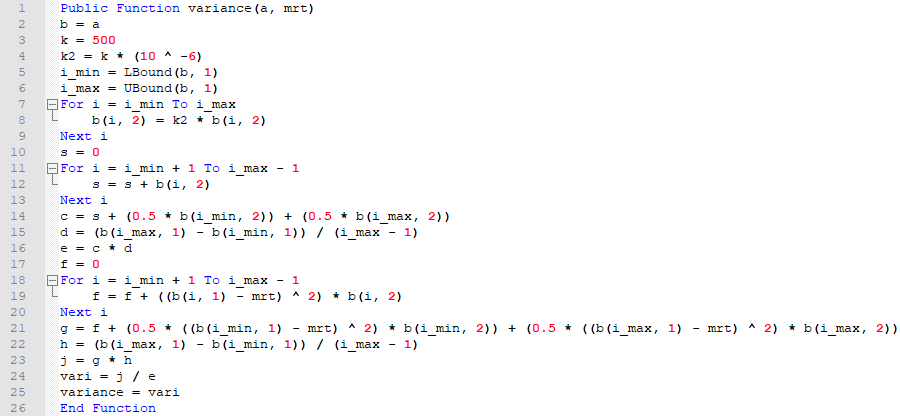
Once again, a limitation of this equation is that there are two integrals with an upper limit of infinity, which cannot be experimentally measured to, therefore, the integrals are instead integrated between the minimum and maximum times given by the data.

**Concept**

Conceptually, this function is again extremely similar to the two functions already discussed. However, the user must also specify the cell containing the mean residence time in the arguments. The first 16 lines are used to calculate the trapezium rule approximation for , which has been explained in the **methodology** for **tracer\_amount**.

For the numerator, the structure of the code is identical, but the code multiplies all values of C(t) by their corresponding value to obtain . These values can then be used in the trapezium rule to obtain an approximation for the integral. To obtain the variance, we then divide the approximation for by the approximation for .

**Function 3 – variance**



**Methodology**

Please refer to the **methodology** for **tracer\_amount** for an explanation of the first 16 lines of code, as they are essentially the same, and for an overall explanation of how to apply the trapezium rule in Visual Basic to approximate definite integrals.

From line 17, we calculate the numerator of the equation. Let:

I.E:

Therefore:

1. =

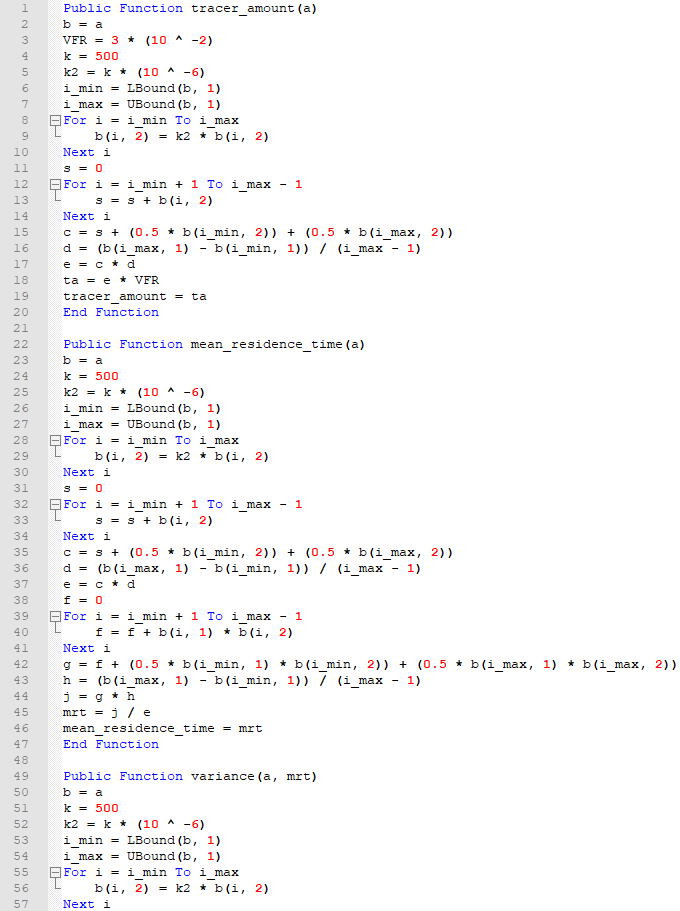
Once the approximation for the numerator is obtained, we then divide the approximation for by the approximation for to give us the variance.

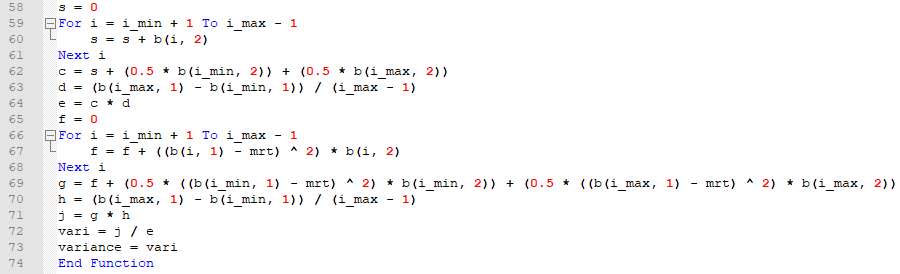
**Line-by-Line Explanation**

|  |  |
| --- | --- |
| **Lines** | **variance(a, mrt)** |
| 1 | Function declaration with argument of 2D array of any size, cell in which the mean residence time is present from **mean\_residence\_time.** |
| 2 | Create and store the ‘proper’ 2D array. |
| 3-4 | Storage and conversion of calibration factor. |
| 5-6 | Define the upper and lower bounds of the rows of the array. |
| 7-9 | [Begin calculation of Convert all values in the G(t) column to C(t) values. |
| 10 | Define new variable. |
| 11-13 | Sum all values in the C(t) column excluding the first and last values, as per trapezium rule. |
| 14 | Half the first and last values and add to the previous sum, as per trapezium rule. |
| 15 | Calculate Δx by subtracting the minimum time from the maximum time and dividing by the number of rows minus one, as per trapezium rule. |
| 16 | Multiply the sum and Δx, as per trapezium rule. [End calculation of |
| 17 | Define new variable. |
| 18-20 | [Begin calculation of Multiply all values in the C(t) column by the corresponding time minus the mean residence time squared, excluding the first and last values, as per trapezium rule. |
| 21 | Half the first and last values and multiply by the corresponding time minus the mean residence time squared and add to the previous sum, as per trapezium rule. |
| 22 | Calculate Δx by subtracting the minimum time from the maximum time and dividing by the number of rows minus one, as per trapezium rule. |
| 23 | Multiply the sum and Δx, as per trapezium rule. [End calculation of |
| 24-25 | Divide by and give the function the result. |
| 26 | End the function. |

**Appendix I**

**Full code**





**References**

Bock, H., n.d. *Assignment - The residence time distribution of a packed column,* Edinburgh: Heriot-Watt University.

Bock, H., n.d. *Visual Basic for Design - Tutorial 4/5,* Edinburgh: Heriot-Watt University.

Ho, D., 2016. *About.* [Online]   
Available at: https://notepad-plus-plus.org/  
[Accessed 16 February 2018].