International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2016): 79.57 | Impact Factor (2015): 6.391

A Computational Algorithm for the Numerical Integration of a Function of One or More Variables

Alexandre César Balbino Barbosa Filho

Department of Chemical Engineering, Federal University of Campina Grande , PB, BRAZIL

Abstract: ***The objective of the study is to present a computational algorithm for the numerical calculation of integrals of a function with any amount of variables, with defined lower and upper limits of integration for its variables. The method find the integral value for functions of any dimensions and comprises of an algorithm that is kind of a modified version of a Monte Carlo method for numerical integration, but utilizes more other concepts and structure. The algorithm generates organized points in the Euclidian space towards the function, for increasing the accuracy of the results. The results shown the high efficiency of the method through calculated errors compared to analytical values, and also show the linked algorithms execution time to its functions.***

Keywords: Integral, Numerical integration, Mathematical programming, Computational algorithm

1. Introduction

Computational simulations are advancing each day more, assuming an essential role in the human activities. One example of prospering, by using these techniques, is the universe discovering by human beings, on which before the rockets being launched, a partial differential equations (PDE) system derived from conservative laws are solved. Such methods avoid mistakes and accidents, reduce costs, and predict phenomena occurring in a process.

Stanley J. Farlow [1] cites 10 methods to solve PDE in his book, and one of them is the *Integral Equations*, which consists on changing the PDE to an integral equation. What if integrals could be solved by a new numerical integration method with high accuracy? The results would have more approximation of the reality.

So the present paper shows a method to improve numerical simulation technology by the use of a computational algorithm that solves integrals. It can be used not only in the simulation field, but also to solve other problems that includes integral.

The Monte Carlo Method are in the class of computational algorithms and use random sampling to obtain numerical results, but aleatory generation of points makes the problem more dependent of luck, increasing the chance of obtaining uncertainties in the results.

The advantage of placing points with organized distributions with an algorithm comprised by conditional statements, is that the computer do not need to count the points inside the interested region, fleeing the *inside or outside problem* [2].

The Monte Carlo integration method for calculating the area under the curve of a function of one variable, as shown in Figure 1.1, comprises of the following steps [3,4]:

1) Put the function curve inside of a rectangle with known area;
2) Place a known amount of random points inside the rectangle;
3) Count the number of points that lie inside the rectangle;
4) The area under the curve of the function is proportional to the number of points that lie below it and is given by Equation (1):

$$A_f = A_R \cdot \frac{N_f}{N_R} \quad (1)$$

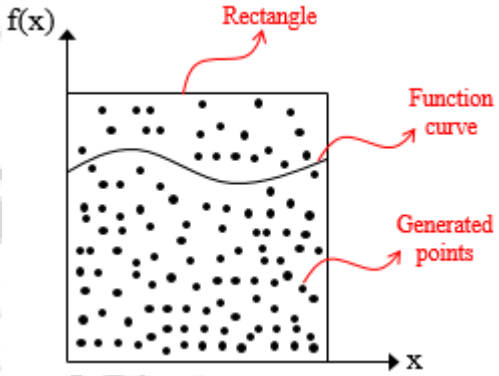

Figure 1.1: Figure merely illustrative of the Monte Carlo method for the calculation of the integral of the function of one variable

A_f is the area below the function curve; A_R is the area of the rectangle; N_f is the number of points below the function curve; N_R is the number of points inside the rectangle.

One disadvantage, if the amount of the generated points is huge and if the function has a few amount of variables, is the slow convergence. But with the advancing of the technology, more specifically of the super and quantum computers, the convergence time will expressively reduce.

The methodology's technique of the present paper is to calculate the integrals of a function of any variables by generating organized points in an imaginary region, and through the use of analogous equations of Equation (1), and others equations. By definition, in this method, the imaginary region is a region on a Euclidian space to which covers all the function surface or curve at specified limits. If it is being calculated a single integral, the imaginary region is an area

and has therefore two dimensions. The integration of a function of two variables gives a volume of three dimensions, but for a function of more than two variables, it gives a "volume" of more than three dimensions and it starts to become a complex observing problem, but a feasible one.

The generation of points is done through the computational algorithm. The count of points, between the function surface and the Euclidian space origin of the imaginary region, is done through conditional statements in the computational algorithm. The total points inside the imaginary region is also counted through conditional statements.

The space dimension of an imaginary region of the method, in the Euclidian space, are given by the number of variables that are being integrated plus one. The dimensional limits of the imaginary region are the variables limits of integration and the chosen maximum and minimum limits of the *y* axis value for the range of generated points. The values for the function that is being integrated are in the same axis of the mentioned *y* axis above. Figure 1.2 illustrates a Euclidian space with the surface of a function of two variables, and also shows an imaginary region covering all the function at a limited domain.

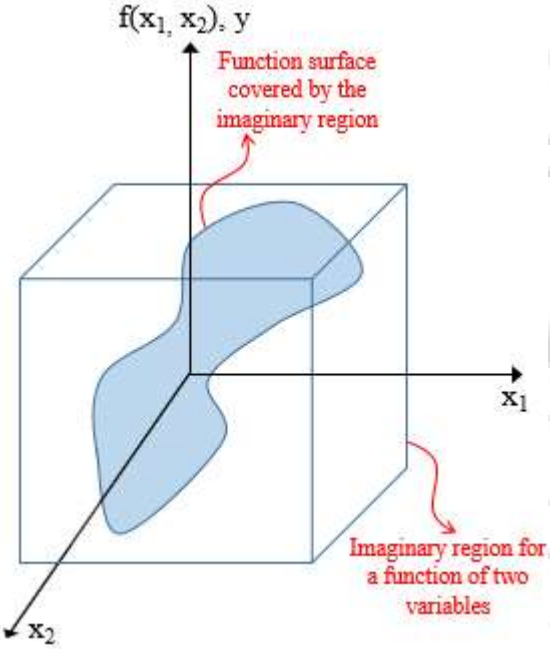

Figure 1.2: Figure merely illustrative of a Euclidian space with an imaginary region covering the surface of a function of two variables

Some concepts must be presented before using the method, more specifically, the meaning of the variables PSNC, PSNTP, NSNC, NSNTP, PSTR, PSR, NSTR and NSR:

- ➔ PSNC: The sum of points that are distributed in the positive side of the imaginary region (also in positive side of Euclidian space) and are placed between the function surface and the Euclidian's space origin;
- ➔ PSNTP (Positive side's number of total points): The sum of all points that are distributed in the positive side of the imaginary region (also in positive side of Euclidian space);
- ➔ NSNC: The sum of points that are distributed in the negative side of the imaginary region (also in negative side of Euclidian space) and are placed between the function surface and the Euclidian's space origin;
- ➔ NSNTP (Negative side's number of total points): The sum of all points that are distributed in the negative side of the imaginary region (also in negative side of Euclidian space);
- ➔ PSTR (positive side total region): Quantity of points distributed in all the positive side of the imaginary region;
- ➔ PSR (positive side region): Quantity of points distributed between the function surface inside the positive side of the imaginary region, and the Euclidian's space origin;
- ➔ NSTR (negative side total region): Quantity of points distributed in all the negative side of the imaginary region;
- ➔ NSR (negative side region): Quantity of points distributed between the function surface inside the negative side of the imaginary region, and the Euclidian's space origin.

On analogous with Equation (1), Equations (2) and (3) below, calculate the PSR and NSR value:

$$PSR = \frac{PSNC \cdot PSTR}{PSNTP} \quad (2)$$

$$NSR = \frac{NSNC \cdot NSTR}{NSNTP} \quad (3)$$

PSNC, PSNTP, NSNC and NSNTP values are determined during the code execution through the counting of the generated points under certain conditions. The PSTR and NSTR values are calculated by Equations (4) and (5):

$$PSTR = \left| \left(\sum_{i}^{n} x_{i,UL} - x_{i,LL} \right) \cdot (y_{max}) \right| \quad (4)$$

$$NSTR = \left| \left(\sum_{i}^{n} x_{i,UL} - x_{i,LL} \right) \cdot (y_{min}) \right| \quad (5)$$

In the summation of Equations (4) and (5), "n*"* and "i", are respectively, the quantity of the function integrating variables and the integrating variable. "$x_{i,UL}$" and "$x_{i,LL}$", are respectively, the variable "i" upper and lower limits of integration. "y_{max}" and "y_{min}", are respectively, the function maximum and minimum possible value at the domain of the given limits of integration at the Euclidian space. If "n" is equal to 2, e.g., Equations (4) and (5) can only represent the volume of a rectangle or cube.

The value of the y axis for a point in the positive side of the Euclidian space is only comprised by positive values for the y coordinate, while the opposite is true for the negative side. For example, each of the positive side and the negative side of a third dimensional imaginary region is comprised by 4 octants of a Euclidian three-dimensional coordinate system.

With the values of PSR and NSR in hands, the integral value of the function with defined limits of integration is calculated by Equation (6) below:

$$Integral\ value = PSR - NSR \quad (6)$$

2. Methodology and Solved Examples

The algorithm calculate the definite multiple integral of any function that has all of its components defined with the limits of integration. For example, the Gaussian function defined by Equation (7), may not be compatible with the method described on this article if the constants a, b and c are not specified as a numerical value:

$$f(x) = a.e^{\frac{-(x-b)^2}{2c}} \quad (7)$$

The reason for this is that Equation (7) has a constant variable that is defined as a symbol instead of a number, compromising the reading of the algorithm by the computer. In the use of this method, all the variables inside the integrated function must be defined as a number, and its limits of integration must be specified. Meaning that if a, b and c are numerically defined, Equation (7) can be solved by this paper methodology.

The method calculate the definite integral or the definite multiple integrals (F) of a function (f(x1,…,xn)) of any variables, just like in Equation (8):

$$F = \int_{x_{nmin}}^{x_{nmax}} \dots \int_{x_{1min}}^{x_{1max}} f(x_1, \dots, x_n) \cdot dx_1 \dots dx_n \quad (8)$$

2.1. Algorithm's Methodology

The algorithm is comprised by multiple nested loops and 6 conditional statements, which the construction of the computational algorithm is comprised by the following steps:

Step 1: Define the lower and upper limits of integration for all the function variables;
Step 2: Use an analytical or numerical method to find the global minimum and maximum values of the function that is being integrated. These found values must be the function's possible minimum and maximum values comprised between the variables' limits of integration;

On another words, Step 2 also says that the minimum and maximum global values do not need to be an inflection point. The found global minimum and maximum values, are respectively, the lower and upper limits for the y axis of the imaginary region along the Euclidian space, and are used in the y axis loop at Step 4. Step 2 is used to guarantee that the imaginary region will cover all the function surface, saving computational effort and execution time in the program running, comparing to the case wherein chosen values for the y axis of the imaginary region are used. At this step, a lot of numerical methods can be used, and Domain's Sweep algorithm [5] is strongly recommended, because it is a method that finds minimum and maximum values of functions, in a specified range of domain, even if they are not an inflection point.

Step 3: Set the initial values for PSNC, PSNTP, NSNC and NSNTP as zero;

Step 4: Put nested loops statements, one loop for each variable, to which the outer loop is always linked to the first variable (x1). Each loop vary its variable value with a given step size (step size for the generation of points), starting in its lower integration limit until the upper limit of integration. The last inner loop of these nested loops, must correspond to the y axis values (linked to y variable) of the generated points in the imaginary region, which is the same axis of the integrated function as mentioned earlier;

In this step, each variable step size must be defined according to the users will. When the nested loop is being executed by the computer, it is like the program is walking into the Euclidian space while generate points on each conjunct of variables values (x1, …, xn, y). These generation is organized because this "walk" is chosen by the user when it is specified each variable step size.

Step 5: Inside the last inner loop, define the function (f(x1, …, xn)) that is being integrated and calculate its value with the currently variables values supplied by the nested loops in Step 4;

Step 6: Yet inside the last inner loop, put the first conditional statement which says that if the y value is higher than zero, then, add the currently PSNTP value by one and store the new value;

If the first conditional statement is not satisfied during the code execution, then, the currently PSNTP value is not updated and it skips to the next step.

Step 7: Inside the first conditional statement, put the second conditional statement which says that if the y value is less than the function currently value, then, add the currently PSNC value by one and store the new value;

If the second conditional statement is not satisfied during the code execution, then, the currently PSNC value is not updated and it skips to the next step.

Step 8: Put an end statement for the second and first conditional statements;

Step 9: Yet inside the last inner loop, put the third conditional statement which says that if the y value is less than zero, then, add the currently NSNTP value by one and store the new value;

If the third conditional statement is not satisfied during the code execution, then, the currently NSNTP value is not updated and it skips to the next step.

Step 10: Inside the third conditional statement, put the fourth conditional statement which says that if the y value is higher than the function currently value, then, add the currently NSNC value by one and store the new value;

If the fourth conditional statement is not satisfied during the code execution, then, the currently NSNC value is not updated and it skips to the next step.

Step 11: Put an end statement for the third and fourth conditional statements;

Step 12: Put an end statement for each loop statement;

With these technique, the generated points' coordinates will always lie inside of the chosen variables limits, because the loop statements are only working inside the variables limits of integration.

Step 13: Put the fifth conditional statement which says that if the last stored PSNTP value is higher than zero, then, calculate the PSTR value using Equation (4), and PSR value using Equation (2);

If the fifth conditional statement is not satisfied during the code execution, then, the PSR value is zero.

Step 14: Put an end statement for the fifth conditional statement;
Step 16: Put the sixth conditional statement which says that if the last stored NSNTP value is higher than zero, then, calculate the NSTR value using Equation (5), and NSR value using Equation (3);

If the sixth conditional statement is not satisfied during the code execution, then, the NSR value is zero.

Step 17: Put an end statement for the sixth conditional statement;

Step 18: Calculate the integral using Equation (6).

2.2. Computational Code Example

The present method was used to calculate the integral of functions with one, two and three variables. It was used the MATLAB® (R2015a, Mathworks, Natick, MA, USA) for making the implementation of the algorithm. In this software, the conditional and the loop statements commands are "if" and "for", respectively. It was calculated the minimum and the maximum global values of each exemplified functions before the implementation of the method.

2.2.1. Algorithm for the function of one variable

The function of one variable that is being integrated is given by Equation (9) below:

$$f(x_1) = x_1^3 + x_1 + 1 \quad (9)$$

The lower and upper limits of integration for x1 are -2 and 5, respectively. The steps size of variation for x1 and y, in the algorithm, are 0.0001. The minimax method used was the Domain's Sweep [5], and the function's maximum and minimum possible value in this range of integration are, respectively, 131 and -9. The integral that is being exemplified is given by Equation (10):

$$F = \int_{-2}^{5} (x_1^3 + x_1 + 1).dx_1 \quad (10)$$

The algorithm for the calculation of this integral is:

```
1- x1LL = -2;
2- x1UL = 5;
3- ymin = -9;
4- ymax = 131;
5- PSNC = 0;
6- PSNTP = 0;
7- NSNC = 0;
8- NSNTP = 0;
9- for x1 = x1LL: 0.0001 : x1UL
10- for y = ymin: 0.0001 : ymax
11- func = (x1^3)+ x1 + 1;
12- if y > 0
13- PSNTP = PSNTP + 1;
14- if y < func
15- PSNC = PSNC + 1;
16- end
17- end
18- if y < 0
19- NSNTP = NSNTP + 1;
20- if y > func
21- NSNC = NSNC + 1;
22- end
23- end
24- end
25- end
26- if PSNTP>0
27- PSTR = abs ((x1UL-x1LL) * (ymax-0));
28- PSR = (PSNC*PSTR)/(PSNTP);
29- else PSR=0;
30- end
31- if NSNTP>0
32- NSTR = abs((x1UL-x1LL) * (ymin-0));
33- NSR = (NSNC*NSTR)/(NSNTP);
34- else NSR =0;
35- end
36- INTEGRALFUNC = PSR - NSR;
37- fprintf ('The integral of the function of one variable at the given limits of integration is = %10.8f \n',INTEGRALFUNC)
```

2.2.2. Algorithm for the function of two variables

The function of two variables that is being integrated is given by Equation (11) below:

$$f(x_1, x_2) = x_1^2 + x_2^3 + 1 \quad (11)$$

The lower and upper limits of integration for x1 and x2 are -2 and 3 for both. The steps size of variation for x1, x2 and y, in the algorithm, are 0.005. The minimax method used was the Domain's Sweep [5], and the function's maximum and minimum possible value in this range of integration are, respectively, 37 and -7. The integral that is being exemplified is given by Equation (12):

$$F = \int_{-2}^{3}\int_{-2}^{3} (x_1^2 + x_2^3 + 1).dx_2.dx_3 \quad (12)$$

The algorithm for the calculation of this integral, with the including of Domain's Sweep algorithm in lines 1 to 17, is:

```
1- YMINIMUM = 10^32;
2- YMAXIMUM = -(10^32);
3- for x1 = -2:0.01:3
4- for x2 = -2:0.01:3
5- fun = (x1^2)+(x2^3)+1;
6- if fun > YMAXIMUM;
7- YMAXIMUM = fun;
8- X1MAX = x1;
9- X2MAX = x2;
10- end
11- if fun < YMINIMUM
12- YMINIMUM = fun;
13- X1MIN = x1;
14- X2MIN = x2;
15- end
16- end
17- end
18- x1LL = -2;
19- x2LL = -2;
20- ymin = YMINIMUM;
21- x1UL = 3;
22- x2UL = 3;
23- ymax = YMAXIMUM;
24- PSNC = 0;
25- PSNTP = 0;
26- NSNC = 0;
27- NSNTP = 0;
28- for x1 = x1LL: 0.005 : x1UL
29- for x2 = x2LL : 0.005: x2UL
30- for y = ymin: 0.005 : ymax
31- func =  (x1^2) + (x2^3) + 1;
32- if y > 0
33- PSNTP = PSNTP + 1;
34- if y < func
35- PSNC = PSNC + 1;
36- end
37- end
38- if y < 0
39- NSNTP = NSNTP + 1;
40- if y > func
41- NSNC = NSNC + 1;
42- end
43- end
44- end
45- end
46- end
47- if PSNTP>0
48- PSTR = abs( (x1UL-x1LL) * (x2UL-x2LL) * (ymax-0) );
49- PSR = (PSNC*PSTR)/(PSNTP);
50- else PSR=0;
51- end
52- if NSNTP>0
53- NSTR = abs( (x1UL-x1LL) * (x2UL-x2LL) * (ymin-0) );
54- NSR = (NSNC*NSTR)/(NSNTP);
55- else NSR =0;
56- end
57- INTEGRALFUNC = PSR - NSR;
58- fprintf ('The integral of the function of  two variables at the given limits of integration is = %10.8f \n',INTEGRALFUNC)
```

2.2.3. Algorithm for the function of three variables

The function of three variables that is being integrated is given by Equation (13):

$$f(x_1, x_2, x_3) = x_1^2 + x_2^2 + x_3^3 - x_1 \quad (13)$$

The lower and upper limits of integration for x1, x2 and x3 are respectively: -2 and 1, -2 and 1, -2 and 1. The steps size of variation for x1, x2, x3 and y, in the algorithm, are 0.01. The minimax method used was the Domain's Sweep, and the function's maximum and minimum possible value in this range of integration are, respectively, 11 and -8.25. The integral that is being exemplified is given by Equation (14):

$$F = \int_{-2}^{1}\int_{-2}^{1}\int_{-2}^{1}(x_1^2 + x_2^2 + x_3^3 - x_1).dx_3.dx_2.dx_1 \quad (14)$$

The algorithm for the calculation of this integral is:

```
1- x1LL = -2;
2- x2LL = -2;
3- x3LL = -2;
4- x1UL = 1;
5- x2UL = 1;
6- x3UL = 1;
7- ymin = -8.25;
8- ymax = 11;
9- PSNC = 0;
10- PSNTP = 0;
11- NSNC = 0;
12- NSNTP = 0;
13- for x1 = x1LL: 0.01 : x1UL
14- for x2 = x2LL : 0.01: x2UL
15- for x3 = x3LL : 0.01 : x3UL
16- for y = ymin: 0.01 : ymax
17- func = x1^2 + x2^2 + x3^3 - x1;
18- if y > 0
19- PSNTP = PSNTP + 1;
20- if y < func
21- PSNC = PSNC + 1;
22- end
23- end
24- if y < 0
25- NSNTP = NSNTP + 1;
26- if y > func
27- NSNC = NSNC + 1;
28- end
29- end
30- end
31- end
32- end
33- end
34- if PSNTP>0
35- PSTR = abs( (x1UL-x1LL) * (x2UL-x2LL) * (x3UL-x3LL) * (ymax-0) );
36- PSR = (PSNC*PSTR)/(PSNTP);
37- else PSR=0;
38- end
39- if NSNTP>0
40- NSTR = abs( (x1UL-x1LL) * (x2UL-x2LL) * (x3UL-x3LL) * (ymin-0) ) ;
41- NSR = (NSNC*NSTR)/(NSNTP);
```

```
42- else NSR =0;
43- end
44- INTEGRALFUNC = PSR - NSR;
45- fprintf ('The integral of the function of three variables at the given limits of integration is = %10.8f \n',INTEGRALFUNC)
```

3. Results and Discussions

The exposed results are based in the discussed conditions in the methodology and through the use of a computer Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz, 64 bits and RAM of 8 GB. The computational algorithms were executed with windows 10 (operating system) with a total utilization of 25% of the processor in all the cores.

3.1. Results for the function of one variable

The algorithm execution time for the function of one variable was 42489.41 seconds and its integral value was 169.75345. The analytical integration of the function of one variable, at the given limits of integration, provides a value equal to 169.75. The error between the analytical value and the present method's value is given by Equation (15) below:

$$Error_1 = \frac{169.75345 - 169.75}{169.75} = 2.0324 \cdot 10^{-3}\% \quad (15)$$

If the steps size of variation for x1 and y were 0.001 instead of 0.0001, the algorithm execution time and the function's integral value would be respectively, 487 seconds and 169.78457206. Then, the error would be given by Equation (16):

$$Error_1 = \frac{169.78457206 - 169.75}{169.75} = 2.03664 \cdot 10^{-2}\% \quad (16)$$

3.2 Results for the function of two variables

The algorithm execution time for the function of two variables was 3853.36 seconds and its integral value was 164.79618309. The analytical integration of the function of two variables, at the given limits of integration, provides a value equal to 164.5833. The error between the analytical value and the present method's value is given by Equation (17) below:

$$Error_3 = \frac{164.79618309 - 164.5833}{164.5833} = 1.923 \cdot 10^{-1}\% \quad (17)$$

3.3. Results for the function of three variables

The algorithm execution time for the function of three variables was 22816.73 seconds and its integral value was 33.76086042. The analytical integration of the function of three variables, at the given limits of integration, provides a value equal to 33.75. The error between the analytical value and the present method's value is given by Equation (18):

$$Error_3 = \frac{33.76086042 - 33.75}{33.75} = 3.2179 \cdot 10^{-2}\% \quad (18)$$

4. Conclusion

The present method calculate the definite multiple integral of a multivariable function, even if it does not have an analytical solution. The shown algorithm execution time is directly proportional to the given variables' steps of variation, to the quantity of integrated variables, to the limits of integration, and to the difference value between the function's maximum and minimum possible value (imaginary region *y* axis limits) at the variables limits of integration. Smaller the step size of variation for a variable, higher the tendency to get an optimum accuracy value. The calculated errors proved that the method has high efficiency.

5. Future Scope

One scope is to use the present method for the numerical integration of partial differential equations, e. g. fluid mechanics and electromagnetism PDEs, wave equation etc. Another scope is to use the shown method for the numerical integration of mathematical functions of all areas, e.g. potential theory, and to compare the accuracy of the results with others numerical methods.

References

[1] Farlow, S. J. (1993). Partial differential equations for scientists and engineers. Courier Corporation.
[2] Milgram, M. S. (1989). Does a point lie inside a polygon?. Journal of Computational Physics, 84(1), 134-144. Available from: https://doi.org/10.1016/0021-9991(89)90185-X [Accessed 19th January 2018]
[3] Robert, C. P. (2004). Monte carlo methods. John Wiley & Sons, Ltd.
[4] Caflisch, R. E. (1998). Monte carlo and quasi-monte carlo methods. Acta numerica, 7, 1-49.
[5] Balbino Barbosa Filho, A.C. Domain's Sweep, a Computational Method to Optimise Chemical and Physical Processes. *Preprints* **2017**, 2017070087 (doi: 10.20944/preprints201707.0087.v1). Available from: https://www.preprints.org/manuscript/201707.0087/v1 [Accessed 26th January 2018]

Author Profile

Alexandre César Balbino Barbosa Filho is in his last year of Chemical Engineering graduation at Federal University of Campina Grande, Campina Grande, PB, Brazil. Alexandre have been actuating in the areas of modeling, simulation, optimization and control of processes, development of processes design and mathematical programming. During the last semester of the course, he carried out an internship in a sugar cane mill, Usina Santo Antonio, located in São Luiz do Quitunde –AL, Brazil.