The American University in Cairo School of Sciences and Engineering Associate Dean for Undergraduate Studies Summer 2020

Engineering Analysis and Computation- ENGR 3202
Section: 01
Term Project

Abdelsalam ElTamawy 900170376
Habiba Abdelwahed 900170816
Madiha Ahmed 900162823
Mazen Ahmed 900161021

18/07/2020

Abstract

The following report is discussing the methodology of coding a user-friendly program that helps recreational pools designers to calculate the various volumes and surface areas of the pools because it is hard for them to do it manually because of the various irregular shapes of these specific kinds of pools. The report also includes the validation of this coding and the results of it. The validation was tested by using a manually calculated example of a specific pool and comparing the results. The program was meant to be as simple and user-friendly as possible and to give the user the luxury to choose from more than one option to calculate the volume and surface area. He can either input data points or functions of the pool shape and the program is manipulated to deal with every condition to finally calculate what is required. It was taken into consideration while programming to consider the best estimate method for the most accurate results either by using the most efficient methods of integration or with the consistency by iteration.

Contents

List of Figures

List of Tables

1 Introduction

In the following report, we will provide the designer of the pool with a computational software that calculates the exact volume of the water inside the pool in order to design its main foundation, select the filters, chemical treatment equipment, circulation pump ,and the overflow tank. In addition, the software also computes the exact surface area which assists both the designer and builder in order to estimate the total amount of cladding material needed for this project. Moreover, the computational software is based on the use of multidimensional numerical integration; therefore, it accepts the shape of the pool in both functions and numerical formats.

2 Methodology

2.1 Description

At the first glance, it was thought of a program that can absorb dealing with as many function as possible in their general form so the first trial of the coding was by using the numerical methods of integration to get an accurate solution to whatever functions the user would input. The integration was calculated by using data points, pushing the shape to the first quadrant, calculating the levels and overlapping then subtracting the integration of the overlapping parts to get the actual surface area and with the same way getting the volume considering the z axis. This method also gave the user the freedom to input either data points or a function that describes the shape of the pool. After discussions, it seemed that we can do it analytically with more simpler ways using the built in integration functions in MATLAB. It's known that it won't be that accurate and general as the first method ,but that's what was required in this level of the course. So two simple function were used in the code; one get the length of the line and the other gets the integration through points and we used these functions to calculate the surface area and the volume analytically.

2.2 Compilation Code

Note: the code contains some defined function that MATLAB calls

```
is_pts = input('Points (0) or Function (1) ? [0] : ');
        f2 = input(['Enter f2(x) at z = ' , num2str(I(i, 1)), ': ' ]);
        x_lower = input('Enter x lower limit: ');
        x_upper = input('Enter x upper limit: ');
        I(i, 2) = abs(int(sym(f1), x_lower, x_upper)) + abs(int(sym(f2), x_lower, x_upper));
        if x_lower < 0
            x_upper = x_upper + abs(x_lower);
            x_lower = 0;
        end
        I(i, 2) = int(sym(f1), x_lower, x_upper) + int(sym(f2), x_lower, x_upper);
        if i != z_num
            A(i,2) = abs(int(sym(sqrt(1+diff(f1)^2)), x_lower, x_upper)) + abs(int(sym(sqrt(1+diff(f2)^2)), x_lower, x_upper));
        else
            bottom = abs(int(sym(f1), x_lower, x_upper)) + abs(int(sym(f2), x_lower, x_upper));
        end
    end
    fprintf("\n%s\nSurface Area of Functoin: \n A ~ %10.10f\n%s\n", repelem('*', 50), best_estimate_2D(best_estimate_2D(A(:,1), A(:,2)) + bottom), repelem('*', 50));
    fprintf("\n%s\nFunction Integration: \n I ~ %10.10f\n%s\n", repelem('*', 50), best_estimate_2D(I(:,1), I(:,2)), repelem('*', 50));
end
```

2.3 2D Best Estimate Method

The 1st dimension is fixed then the points along the 2^{nd} dimension are integrated, and we repeat this process by iterating over all the possible unique values of the 1^{st} dimension. The method can be represented as follows:

$$\begin{gathered}\text{Iteration 1: integrate}(f(x_1, y_1), f(x_1, y_2), ..., f(x_1, y_n)) \\ \text{Iteration 2: integrate}(f(x_2, y_1), f(x_2, y_2), ..., f(x_2, y_n) \\ ... \\ \text{Iteration n: integrate}(f(x_n, y_1), f(x_n, y_2), ..., f(x_n, y_n)\end{gathered}$$

How the "integrate" function works is by looking at the spacing between the consecutive x points:

$$\begin{gathered}\text{if 3 equal segments: Simpson's 3/8 rule is used} \\ \text{if 2 equal segments: Simpson's 1/3 rule is used} \\ \text{if no equal segment: Trapezoidal rule is used} \\ \text{if only 1 point exists: the image of the point is used}\end{gathered}$$

2.4 3D Pool Model

The models that we were using during our testing of the program are 3D pools that we designed in the Blender software and dumped its points to a csv file. Find below a sample of the points:

x	y	z
-1	-1	0
-1	0.818	0

Table 1: Sample points

3 Volume

Our program can calculate the volume of the pool given either a set of points or a set of functions describing the pool.

3.1 Points Integration

The user enters a set of (x, y, z) coordinates in a csv file. Then, the program preprocesses the file, then outputs the integration.

3.1.1 Data Preprocessing

The csv file is read, then the unique values in the X and Y dimensions are computed and sorted in order to use them as the index for any Z value. The data structure looks as follows:

	x_1	x_2	...	x_n
y_1	$z(x_1, y_1)$	$z(x_2, y_1)$	...	$z(x_n, y_1)$
y_2	$z(x_1, y_2)$	$z(x_2, y_1)$	...	$z(x_n, y_2)$
...	...	...	...	...
y_n	$z(x_1, y_n)$	$z(x_2, y_n)$	...	$z(x_n, y_n)$

Table 2: Points data structure

3.1.2 Double Integration

After preprocessing the data, the best estimate method (2.3) is used to integrate over the y & z dimensions, then is used once more to integrate the computed integrals over the x dimension.
In other words, the x-axis is fixed then the z points along the y-axis are integrated, and we repeat this process by iterating over all the possible unique values of x (found in the csv). Finally, after integrating along y&z, we integrate the values computed from the y & z integration with the x-axis to obtain the final value of the double integration.

3.2 Functions Integration

The user enters a set of $f(x)$ functions via the console. For every function entered, the user should enter the z-level that this function is describing, along with the lower and upper x (i.e. the integration boundaries). Then, the program computes analytically and saves the integral of every entered 2D function. Finally these computed values are integrated over the z dimension using the best estimate method (2.3).

3.3 Surface Area with Points

To obtain the surface area, we consider each Horizontal slice of the pool, starting from the first "slice" along x-axis. We then calculate the perimeter of this slice, then the next and so on until we obtain the perimeter of every single arc along the pool. using the result of the surface area, we can then integrate it to find the surface area of the pool. We integrate similarly to every other instance of integration in the program, through using the best estimate through the Simpsons and trapezoidal rules.

4 Data & Results

4.1 Pool drawing & data points

4.2 results & validation

the model seen in figure 1, this is the object we extracted data points from in order to calculate it's surface area and volume. Calculating the volume of the

model by hand offers the value 1.2 and our program calculates it to be 1.8. Proving it to be accurate to the degree of human error. Calculating the surface area of the model by hand offers the value 3.6 and out program calculates it as 3.5. Also proving it to be accurate.

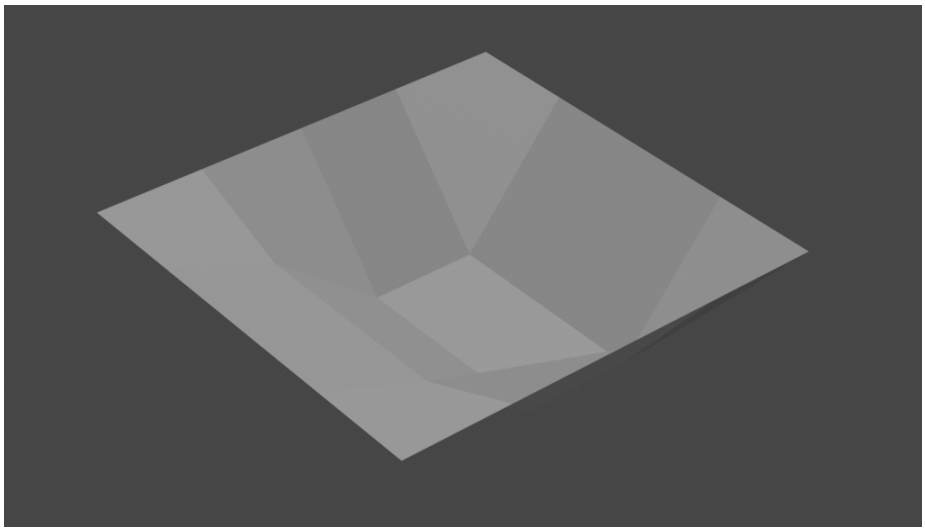

Figure 1: Render of test model

5 Conclusion

In conclusion, the designed computational software assists the designer in calculating both volumes and surface areas in order to ease the process not only that, but consider the best estimate method for the most accurate results either by using the most efficient methods of integration or with the consistency by iteration to assist the designer in the project. Therefore, as discussed in the methodology the code is based on the use of multidimensional numerical integration which accepts the shape of the pool in both functions and numerical formats. The results indicate that the computational software is valid because it was found to be accurate to the degree of human error not only that, but also the surface area was proven to be highly accurate.

6 appendix

Box 21.1 Derivation of Trapezoidal Rule

Before integration, Eq. (21.2) can be expressed as

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + f(a) - \frac{af(b) - af(a)}{b - a}$$

Grouping the last two terms gives

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + \frac{bf(a) - af(a) - af(b) + af(a)}{b - a}$$

or

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + \frac{bf(a) - af(b)}{b - a}$$

which can be integrated between $x = a$ and $x = b$ to yield

$$I = \frac{f(b) - f(a)}{b - a}\frac{x^2}{2} + \frac{bf(a) - af(b)}{b - a}x\Big|_a^b$$

This result can be evaluated to give

$$I = \frac{f(b) - f(a)}{b - a}\frac{(b^2 - a^2)}{2} + \frac{bf(a) - af(b)}{b - a}(b - a)$$

Now, since $b^2 - a^2 = (b - a)(b + a)$,

$$I = [f(b) - f(a)]\frac{b + a}{2} + bf(a) - af(b)$$

Multiplying and collecting terms yields

$$I = (b - a)\frac{f(a) + f(b)}{2}$$

which is the formula for the trapezoidal rule.

Figure 2: Trapezoidal rule

21.2.3 Simpson's 3/8 Rule

In a similar manner to the derivation of the trapezoidal and Simpson's 1/3 rule, a third-order Lagrange polynomial can be fit to four points and integrated:

$$I = \int_a^b f(x)\,dx \cong \int_a^b f_3(x)\,dx$$

to yield

$$I \cong \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$$

where $h = (b - a)/3$. This equation is called *Simpson's* 3/8 *rule* because h is multiplied by 3/8. It is the third Newton-Cotes closed integration formula. The 3/8 rule can also be expressed in the form of Eq. (21.5):

$$I \cong \underbrace{(b - a)}_{\text{Width}} \underbrace{\frac{f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)}{8}}_{\text{Average height}} \tag{21.20}$$

Thus, the two interior points are given weights of three-eighths, whereas the end points are weighted with one-eighth. Simpson's 3/8 rule has an error of

$$E_t = -\frac{3}{80}h^5 f^{(4)}(\xi)$$

or, because $h = (b - a)/3$,

$$E_t = -\frac{(b - a)^5}{6480}f^{(4)}(\xi) \tag{21.21}$$

Figure 3: Simpson's $\frac{3}{8}$ rule

After integration and algebraic manipulation, the following formula results:

$$I \cong \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] \tag{21.14}$$

where, for this case, $h = (b - a)/2$. This equation is known as *Simpson's 1/3 rule*. It is the second Newton-Cotes closed integration formula. The label "1/3" stems from the fact that h is divided by 3 in Eq. (21.14). An alternative derivation is shown in Box 21.3 where the Newton-Gregory polynomial is integrated to obtain the same formula.

Simpson's 1/3 rule can also be expressed using the format of Eq. (21.5):

$$I \cong \underbrace{(b - a)}_{\text{Width}} \underbrace{\frac{f(x_0) + 4f(x_1) + f(x_2)}{6}}_{\text{Average height}} \tag{21.15}$$

Figure 4: Simpson's $\frac{1}{3}$ rule