**Numerical Integration of Vector Integrals**

**Lab 7**

IDName: Niruyan Rakulan Zhiwei Chen

ID:214343438 212654265

**Introduction:**

The purpose of this lab is to construct two functions responsible for solving path integrals. The ‘pathIntegral’ function is receives a 3xn array (each row is x, y, or z component of a point, and n corresponds to the number of points), and returns a 3D vector consisting of the x, y, and z components of the integral of x2+ y2+ z2, along the path specified by the array.

The ‘dotPathIntegral’ receives 3xn array and returns a the integral of (sin 3𝑥) + (𝑧 cos 2𝑦) + 𝑥𝑧2 along the path specified by the array.

**Methodology**

For the ‘pathIntegral’ function, the approach taken was to take the integral of two points along x, then y, and then z, for all the points in the array. The sperate integrals were summed together and outputted. This approach was taken because the path cannot be parameterized beforehand, and hard coded in the code, so the program takes a point-by-point approach to calculating the total integral.

Example: For straight line between (0,0,0) and

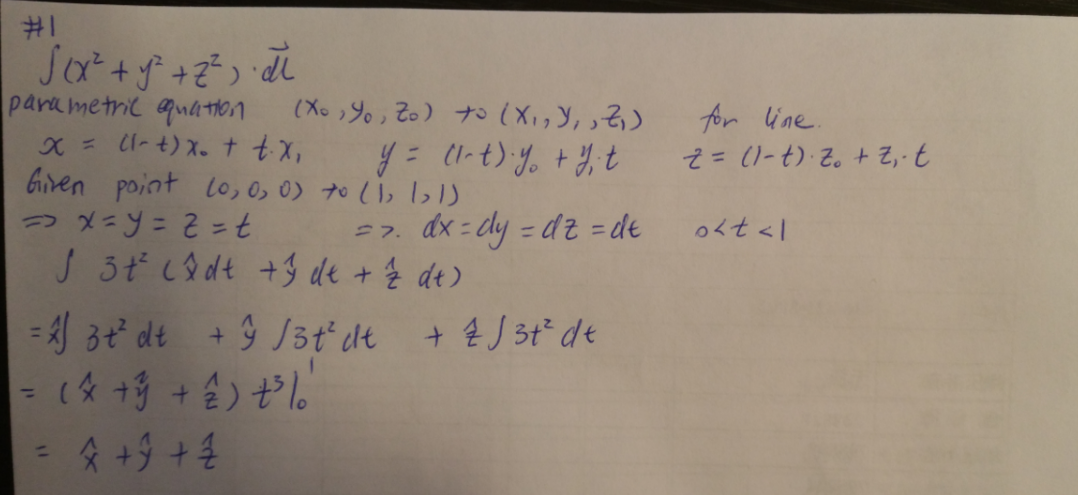


Figure 1: The integration of the function from (0,0,0) to

For the ‘dotPathIntegral’ function, the approach taken was to parameterize the function. This approach was taken because a line can be easily parameterized between two points, and the dot integral can be easily done by substituting the parameterized path into the function. The path between two points is a line defined by

**r(t)** =(1-t)<x0,y0,z0>+t<x1,y1,z1>

**r(t)** = < x0- x0t+ x1t, y0- y0t+ y1t, z0- z0t+ z1t>

The x, y, and z components were subbed into the function so,

**f(r(t))=** <sin 3(x0- x0t+ x1t)) , ((z0- z0t+ z1t) cos 2(y0- y0t+ y1t)) , (x0- x0t+ x1t)( z0- z0t+ z1t)2>

The derivate of the path function was deemed to be:

**r’(t)** = < -x0+ x1, - y0+ y1, - z0+ z1>

Example: For straight line between (0,0,0) and

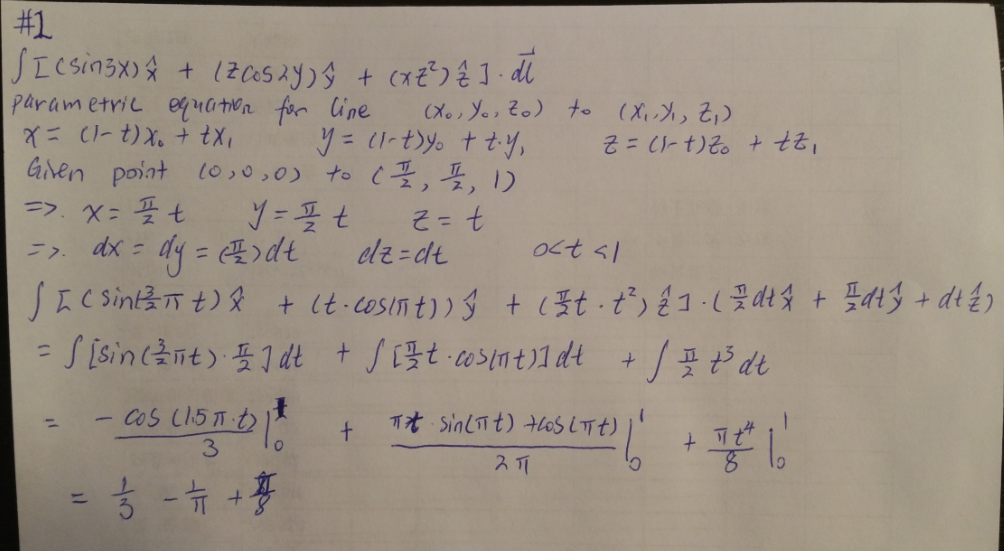


Figure 2: The integration of the function from (0,0,0) to

**Results:**

Table 1: The pathIntegration from (0,0,0) to with different steps

|  |  |  |
| --- | --- | --- |
| Step | Expected | Resulting |
| 0.1 | <1,1,1> |  |
| 0.01 | <1,1,1> |  |
| 0.001 | <1,1,1> |  |
| 0.0001 | <1,1,1> |  |

Table 2: The dotPathIntegration from (0,0,0) (with different steps

|  |  |  |
| --- | --- | --- |
| Step | Expected | Resulting |
| 0.1 | 4.078 |  |
| 0.01 | 4.078 |  |
| 0.001 | 4.078 |  |
| 0.0001 | 4.078 |  |

It was shown that the bigger the step was between the points, the expected value varied more significantly from the resulting value. If smaller steps were taken between points, the resulting value is more accurate; however, if the steps were infinitesimally small, the Matlab program will take significantly longer to complete.

**Conclusion**

It was shown in the lab that if the points were closely spread out, the answer is more accurate, however, if the points were infinitesimally far apart, the Matlab program will take much longer to finish.

**References**

**http://tutorial.math.lamar.edu/Classes/CalcIII/LineIntegralsPtI.aspx**

[**http://tutorial.math.lamar.edu/Classes/CalcIII/LineIntegralsVectorFields.aspx**](http://tutorial.math.lamar.edu/Classes/CalcIII/LineIntegralsVectorFields.aspx)