23 November 2023

Name: **Nijat Kazimli**

Index Number: **323983**

Group: **JA2**

Integration Over a Unit Circle using mid-point rule.

Project number **25 /** Subgroup **A**

1 Task Description

The task is about numerical calculation of the integral $\iint_D f(x,y)dxdy$, where D = {(x, y) ∈ $\mathbf{R^2}$: $x^2 + y^2 \leq 1$}. Polar coordinates need to be used to transform the area of the rectangle [0, 1] x [0, 2π], and then use the composite mid-point rectangle rule with respect to each variable.

2 Project Folder Content

integrateOverUnitCircle.m file contains the main function which takes one argument – a handle to the function to be integrated.

test.m file contains the testing function, where you can find a sanity check and numerical analysis for different cases. This function does not take any arguments.

testFixedFunctions.m file contains a testing Matlab function for fixed functions. It has been used for analyzing different step sizes.

3 Method Description

At the beginning the unit circle is discretized into a polar grid with *num_theta* angular subdivisions and *num_r* radial subdivisions. The angular parameter *theta* ranges from 0 to 2π, and the radial parameter *r* spans the interval [0, 1].

For each polar coordinate (*theta, r)* the corresponding cartesian coordinates (*x, y)* are computed. The function value *f (x, y)* is then evaluated.

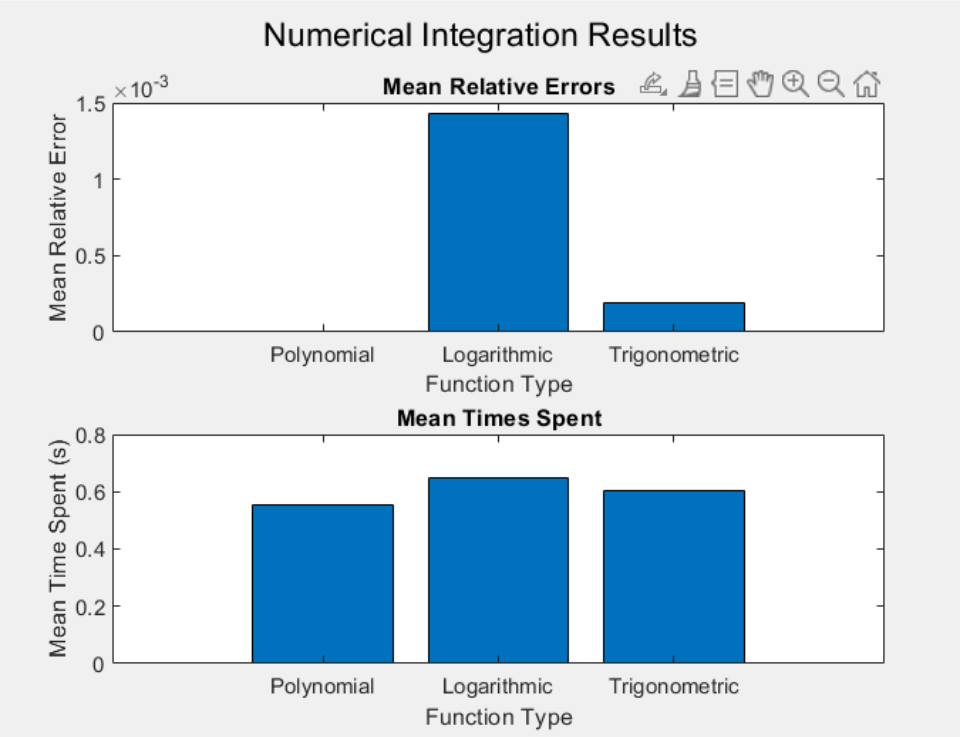
The differential area element *dA* for each polar coordinate is determined as *r * delta_r * delta_theta*.

The product of the function value and *dA* is accumulated in one variable which at the end will be the result.

4 Testing Methodology

At the beginning, I did a sanity check. Next, I proceed to evaluate specific function types, recording both the time spent on each function and its accuracy. It's worth noting that identical polynomials are employed within logarithmic and trigonometric functions to ensure a fair comparison. Here is the outcome:

Graphical Representation:



Polynomials:

Relative Error	Time Spent (in seconds)
0.000001	0.503179
0.000005	0.539516
0.000001	0.496244
0.000007	0.536923
0.000005	0.559012
0.000002	0.562297
0.000001	0.562449
0.000002	0.580236
0.000004	0.609201
0.000005	0.572764
Mean: 0.000003	Mean: 0.552182

Trigonometrics:

Relative Error	Time Spent (in seconds)
0.000097	0.625062
0.000425	0.603864
0.000001	0.593822
0.000001	0.605856
0.000003	0.598135
0.000015	0.595835
0.000190	0.612208
0.000002	0.583812
0.000708	0.611186
0.000432	0.584988
Mean: 0.000187	Mean: 0.601477

Logarithmics:

Relative Error	Time Spent (in seconds)
0.001547	0.657791
0.001313	0.657485
0.002065	0.618115
0.001093	0.661218
0.001332	0.645634
0.001474	0.659822
0.001469	0.652475
0.001354	0.628990
0.001361	0.669580
0.001248	0.631897
Mean: 0.001426	Mean: 0.648301

Then, I proceeded to analyze whether the step size affects the performance and accuracy. To this purpose, I defined 4 different functions with different behaviors, and integrated them. Here is the outcome:

The number of divisions: 1000 (default)

Function	Relative Error	Time Spent (in seconds)
$x^2 + y^2$	0.000001	0.084670
$10x^7 + 8y^9$	0.892684	0.482470
$\log (3x^2 + 2y^3)$	0.000146	0.360869
$\sin (3x^2) + \cos (2y^3)$	0.000000	0.321515

The number of divisions: 500

Function	Relative Error	Time Spent (in seconds)
$x^2 + y^2$	0.000002	0.019936
$10x^7 + 8y^9$	1.013950	0.137288
$\log (3x^2 + 2y^3)$	0.000036	0.083371
$\sin (3x^2) + \cos (2y^3)$	0.000001	0.074228

The number of divisions: 5 (extreme case)

Function	Relative Error	Time Spent (in seconds)
$x^2 + y^2$	0.020000	0.006207
$10x^7 + 8y^9$	~3e3	0.001277
$\log (3x^2 + 2y^3)$	0.116120	0.002131
$\sin (3x^2) + \cos (2y^3)$	0.009404	0.000334

The number of divisions: 10000 (extreme case)

Function	Relative Error	Time Spent (in seconds)
$x^2 + y^2$	0.000000	6.719743
$10x^7 + 8y^9$	0.619404	46.325285
$\log (3x^2 + 2y^3)$	0.000117	35.184710
$\sin (3x^2) + \cos (2y^3)$	0.000000	32.869876

5 Analysis

As you can see, the code works almost perfectly for polynomial functions, and overall, the performance was good.

Because of the oscillatory behavior of trigonometric functions, and because of the behavior of logarithmic functions near zero may require more samples for accurate integration.

When it comes to the number of divisions, as you can see from the results, more division means more accuracy but less efficiency. However, for the functions the value of which do not change very fast the default setting is more than enough.