CSxxxx : Numerical Integration
Assignment 1

Rajat Vadiraj Dwaraknath
EE16B033

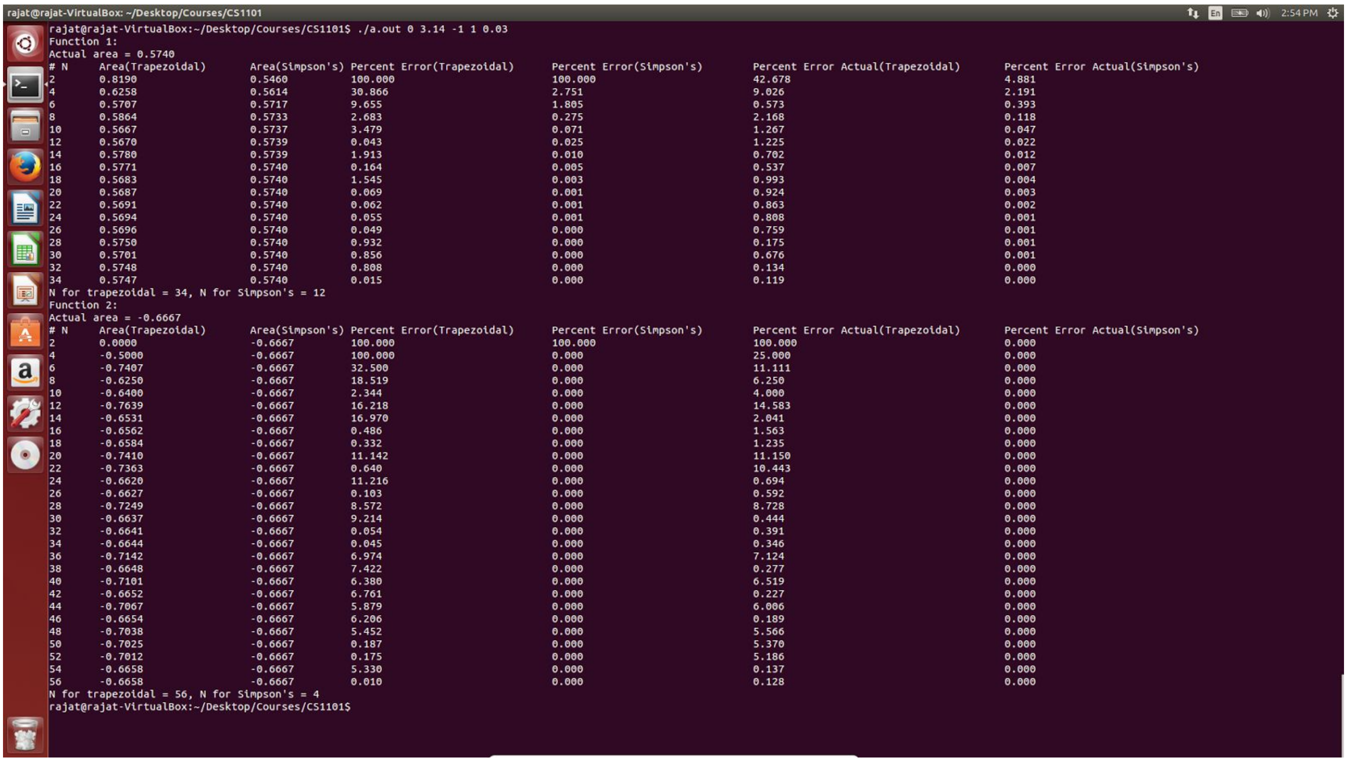
Source Code

```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

/*
   Numerical integration using trapeziodal method and Simpson's method of the functions:
   function1: e^-x * cos^2(x) from 0 to pi
   function2: cos(2*acos(x)) from -1 to 1
   Author: Rajat Vadiraj Dwaraknath, EE16B033
   Date: 14th August 2016

   Input: Limits of integration x1 and x2 for both functions and the required percent error
threshold.
   Output: Table containing the area and the respective errors for the trapezoidal and Simpson's
methods.
   Format:
   Actual area = <value obtained using antiderivative>
   Table:
   # N(number of intervals)\tTrapezoidal Area\tSimpson's Area\tTrap Consecutive Error %\tSimp
Consecutive Error %\tActual Trap Error %\tActual Simp Error %
   <table entries>
   Number of intervals at which the error falls below given threshold for Trapezoidal and Simpsons's
methods.
 */

// get the value of function1 at x
float function1(float x){
       float y = exp(-x)*pow(cos(x),2);
       return(y);
}

// get value of the antiderivative of function1 at x
float inty1(float x){
       float y = exp(-x)*(-0.5 + 0.1*(2*sin(2*x) - cos(2*x)));
       return(y);
}

// get the value of funtion2 at x
float function2(float x){
       float y = cos(2*acos(x));
       return(y);
}

// get value of the antiderivative of function2 at x
float inty2(float x){
       float y = 2.0/3*pow(x,3) - x;
```

```
	return(y);
}
/*
   This function numerically integrates the passed function(funcy) from x1 to x2 using both the 
trapezoidal
   method and Simpson's rule until the percent error between successive values is less than 
errorThreshold.
   It also displays the percent error with respect to the actual area calculated using the 
antiderivative(inty).
 */
void numIntegrate(float (*funcy)(float), float (*inty)(float), float x1, float x2, float 
errorThreshold){

	float N = 2.0; // Initial number of intervals is 2 as it must be even for Simpson's method
	float dx = (x2-x1)/N; // Stepsize
	float y1, y2, trapArea = 0.0, oldTArea = 0.0, simpArea = 0.0, oldSArea = 0.0, x, trapError = 
1, simpError = 1; // Errors initialised to 1 so that inital iteration can occur
	float trapN = 1, simpN = 1; // Number of intervals at which error falls below given threshold
	int i = 0; // Iterator used for Simpson's method
	float actualArea = inty(x2) - inty(x1); // Area calculated using antiderivative
	float actualTrapError = 0, actualSimpError = 0; // Errors with respect to actual area

	printf("Actual area = %.4f\n", actualArea);
	printf("# N\tArea(Trapezoidal)\tArea(Simpson\'s)\tPercent Error(Trapezoidal)\tPercent 
Error(Simpson\'s)\tPercent Error Actual(Trapezoidal)\tPercent Error Actual(Simpson\'s)\n");
	// Table header printed

	while(trapError>errorThreshold){ // Using trapError to compare as simpError will always be 
//less than trapError
		dx = (x2-x1)/N;

		// Trapezoidal Method Start
		oldTArea = trapArea;
		trapArea = 0.0;
		// Using x as the iterator itself
		for(x = x1;x <= x2-dx; x+=dx){
			y1 = funcy(x);
			y2 = funcy(x+dx);
			trapArea += 0.5*(y1+y2)*dx; // Area of a single trapezoid
		}
		// Error calculated
		if(trapArea == 0){
			trapError = 100;
			actualTrapError = 100;
		}else{
			trapError = fabs(100*(1 - oldTArea/trapArea));
			actualTrapError = fabs(100*(1 - trapArea/actualArea));
		}
		// Trapezoidal Method End


		// Simpson's Method Start
		if(simpError != 0){ // Condition to prevent unecessary calculations if Simpson's method 
//is exact(if the function is a cubic polynomial)
```

```
			oldSArea = simpArea;
			simpArea = 0;
			for(i = 0;i < N/2;i++){
				// Simpson's rule formula
				simpArea += funcy(x1 + 2*i*dx) + 4*funcy(x1+(2*i+1)*dx) +
funcy(x1+(2*i+2)*dx);
			}
			simpArea*=dx/3.0;
			// Error Calculation
			if(simpArea == 0){
				simpError = 100;
				actualSimpError = 100;
			}else{
				simpError = fabs(100*(1 - oldSArea/simpArea));
				actualSimpError = fabs(100*(1 - simpArea/actualArea));
			}
			// To find the point at which error falls below given threshold
			if(simpError < errorThreshold && simpN == 1){
				simpN = N;
			}
		}
		// Simpson's Method End

		// Print table entires
		printf("%.0f\t%.4f\t\t\t%.4f\t\t%.3f\t\t\t\t%.3f\t\t\t\t%.3f\t\t\t\t\t%.3f\n", N,
trapArea, simpArea, trapError, simpError, actualTrapError, actualSimpError);
		N+=2.0; // Increment by 2 to keep N even
	}
	trapN = N-2; // Threshold for Trapezoidal method crossed just before ending of loop, so trapN
//= N-2
	printf("N for trapezoidal = %.0f, N for Simpson\'s = %.0f\n", trapN, simpN); // Display of
//threshold points
}

int main(int argc, char **argv){

	if(argc != 6){
		printf("Usage:%s x1 x2(limits for function1) x1 x2(limits for function 2)
percentErrorThreshold\n",argv[0]);
		return(0);
	}
	float x1 = atof(argv[1]);
	float x2 = atof(argv[2]);
	float xx1 = atof(argv[3]);
	float xx2 = atof(argv[4]);
	float errorThreshold = atof(argv[5]);

	// Integrating using the given arguments
	printf("Function 1:\n");
	numIntegrate(function1, inty1, x1, x2, errorThreshold);
	printf("Function 2:\n");
	numIntegrate(function2, inty2, xx1, xx2, errorThreshold);
	return(0);
}
```

Screenshot of Output:



```
rajat@rajat-VirtualBox: ~/Desktop/Courses/CS1101
rajat@rajat-VirtualBox:~/Desktop/Courses/CS1101$ ./a.out 0 3.14 -1 1 0.03
Function 1:
Actual area = 0.5740
# N     Area(Trapezoidal)       Area(Simpson's) Percent Error(Trapezoidal)      Percent Error(Simpson's)        Percent Error Actual(Trapezoidal)       Percent Error Actual(Simpson's)
2       0.8190                  0.5460          100.000                         100.000                         42.678                                  4.881
4       0.6258                  0.5614          30.866                          2.751                           9.026                                   2.191
6       0.5707                  0.5717          9.655                           1.805                           0.573                                   0.393
8       0.5864                  0.5733          2.683                           0.275                           2.168                                   0.118
10      0.5667                  0.5737          3.479                           0.071                           1.267                                   0.047
12      0.5670                  0.5739          0.043                           0.025                           1.225                                   0.022
14      0.5780                  0.5739          1.913                           0.010                           0.702                                   0.012
16      0.5771                  0.5740          0.164                           0.005                           0.537                                   0.007
18      0.5683                  0.5740          1.545                           0.003                           0.993                                   0.004
20      0.5687                  0.5740          0.069                           0.001                           0.924                                   0.003
22      0.5691                  0.5740          0.062                           0.001                           0.863                                   0.002
24      0.5694                  0.5740          0.055                           0.001                           0.808                                   0.001
26      0.5696                  0.5740          0.049                           0.000                           0.759                                   0.001
28      0.5750                  0.5740          0.932                           0.000                           0.175                                   0.001
30      0.5701                  0.5740          0.856                           0.000                           0.676                                   0.001
32      0.5748                  0.5740          0.808                           0.000                           0.134                                   0.000
34      0.5747                  0.5740          0.015                           0.000                           0.119                                   0.000
N for trapezoidal = 34, N for Simpson's = 12
Function 2:
Actual area = -0.6667
# N     Area(Trapezoidal)       Area(Simpson's) Percent Error(Trapezoidal)      Percent Error(Simpson's)        Percent Error Actual(Trapezoidal)       Percent Error Actual(Simpson's)
2       0.0000                  -0.6667         100.000                         100.000                         100.000                                 0.000
4       -0.5000                 -0.6667         100.000                         0.000                           25.000                                  0.000
6       -0.7407                 -0.6667         32.500                          0.000                           11.111                                  0.000
8       -0.6250                 -0.6667         18.519                          0.000                           6.250                                   0.000
10      -0.6400                 -0.6667         2.344                           0.000                           4.000                                   0.000
12      -0.7639                 -0.6667         16.218                          0.000                           14.583                                  0.000
14      -0.6531                 -0.6667         16.970                          0.000                           2.041                                   0.000
16      -0.6562                 -0.6667         0.486                           0.000                           1.563                                   0.000
18      -0.6584                 -0.6667         0.332                           0.000                           1.235                                   0.000
20      -0.7410                 -0.6667         11.142                          0.000                           11.150                                  0.000
22      -0.7363                 -0.6667         0.640                           0.000                           10.443                                  0.000
24      -0.6620                 -0.6667         11.216                          0.000                           0.694                                   0.000
26      -0.6627                 -0.6667         0.103                           0.000                           0.592                                   0.000
28      -0.7249                 -0.6667         8.572                           0.000                           8.728                                   0.000
30      -0.6637                 -0.6667         9.214                           0.000                           0.444                                   0.000
32      -0.6641                 -0.6667         0.054                           0.000                           0.391                                   0.000
34      -0.6644                 -0.6667         0.045                           0.000                           0.346                                   0.000
36      -0.7142                 -0.6667         6.974                           0.000                           7.124                                   0.000
38      -0.6648                 -0.6667         7.422                           0.000                           0.277                                   0.000
40      -0.7101                 -0.6667         6.380                           0.000                           6.519                                   0.000
42      -0.6652                 -0.6667         6.761                           0.000                           0.227                                   0.000
44      -0.7067                 -0.6667         5.879                           0.000                           6.006                                   0.000
46      -0.6654                 -0.6667         6.206                           0.000                           0.189                                   0.000
48      -0.7038                 -0.6667         5.452                           0.000                           5.566                                   0.000
50      -0.7025                 -0.6667         0.187                           0.000                           5.370                                   0.000
52      -0.7012                 -0.6667         0.175                           0.000                           5.186                                   0.000
54      -0.6658                 -0.6667         5.330                           0.000                           0.137                                   0.000
56      -0.6658                 -0.6667         0.010                           0.000                           0.128                                   0.000
N for trapezoidal = 56, N for Simpson's = 4
rajat@rajat-VirtualBox:~/Desktop/Courses/CS1101$
```

Answers to the questions:

1. Using integration by parts, we arrive at the following indefinite integral

$$\int f(x)dx = -e^{-x}(0.5 + 0.1(2\sin 2x - \cos 2x))$$

The function is smooth and continuous everywhere in $[0, \pi]$ as it is a product of continuous functions. It has one extrema in the domain because its derivative, $f(x)$, is 0 only when $x = \pi/2$.

The value of the definite integral can be calculated as follows

$$\left[-e^{-x}(0.5 + 0.1(2\sin 2x - \cos 2x))\right]_0^{\pi} = 0.6(1 - e^{-\pi}) \approx 0.5740$$

2. The errors are printed in the output screenshot. Since decreasing the stepsize by half is equivalent to doubling the number of intervals, it is clear from the output screenshot that the error decreases. This is because a higher number

of smaller line segments(the tops of the trapezoids) results in a more accurate approximation of the curved part of a function's graph, leading to lesser error in the calculated area under the curve.

3. It is clear from the output that Simpson's method performs far better than the trapezoidal method. This can be explained by the fact that Simpson's method locally approximates the curve using a quadratic function which is more accurate than the linear function used by the trapezoidal method. This leads to an $\mathcal{O}(\Delta x^5)$ error which decreases more rapidly than the $\mathcal{O}(\Delta x^3)$ error of the trapezoidal method.

4. Using the simple trigonometric identity

$$\cos 2\theta = 2\cos^2\theta - 1$$

the integral can be easily solved to yield

$$\int \cos(2\arccos x)dx = \frac{2}{3}x^3 - x$$

The definite integral is then

$$\left[\frac{2}{3}x^3 - x\right]_{-1}^{1} = -\frac{2}{3} \approx 0.6667$$

The results of the trapezoidal method and Simpson's method applied on this function are included in the output screenshot. We notice that the second function is actually a second degree polynomial with a restricted domain

$$\cos(2\arccos x) = 2x^2 - 1$$

This means that Simpson's method will be exact for this function(as it is a polynomial of degree 3 or lower). This is evident from the output which shows that the actual error is 0% for all values of N from $N = 2$.