TUT 8

1C.cpp

```cpp
#include <iostream>
#include <bits\stdc++.h>
using namespace std;

int main() {
    float f1, f2, f3, f4;
    float a, b, h, I, i;

    f1 = 0;
    f2 = 9.5120;
    f3 = 2.8087;
    f4 = 0.3537;
    I = 129.52;
    a = 0;
    b = 30;
    h = (b - a) / 3;

    i = (3 * h * (f1 + 3 * f2 + 3 * f3 + f4)) / 8;
    cout << "Calculated integral from Simpson's 3/8 rule is " << i << endl;
    cout << "Error is " << abs(I - i) * 100 / I << "%" << endl;

    return 0;
}
```

Output Clear

```
/tmp/mQjhWneHZt.o
Calculated integral from Simpson's 3/8 rule is 139.934
Error is 8.04064%


=== Code Execution Successful ===
```

2C.cpp

```cpp
#include <iostream>
#include <cmath>

const double T = 1.0; // Period
const double TRUE_IRMS = 3.92588945948; // True RMS value

double iSquared(double t) {
    return (t >= 0 && t <= T / 2) ? (10 * exp(-t / T) * sin(2 * M_PI * t / T)) * (10 * exp(-t / T) * sin(2 * M_PI * t / T)) : 0.0;
}

double trapezoidalRule(int n) {
    double h = (T / 2) / (2 * n);
    double sum = 0.5 * (iSquared(0) + iSquared(T / 2));
    for (int k = 1; k < 2 * n; ++k) {
        sum += iSquared(k * h);
    }
    return h * sum;
}

double simpsonsRule(int n) {
    double h = (T / 2) / (2 * n);
    double sum = iSquared(0) + iSquared(T / 2);
    for (int k = 1; k < 2 * n; ++k) {
        sum += (k % 2 == 0 ? 2 : 4) * iSquared(k * h);
```

```
    }
    return (h / 3) * sum;
}

double richardsonExtrapolation(double I1, double I2, int p) {
    return (std::pow(2, p) * I2 - I1) / (std::pow(2, p) - 1);
}

int main() {
    // Trapezoidal Method
    double I64 = trapezoidalRule(64);
    double I128 = trapezoidalRule(128);
    double trapezoidalIRMS = sqrt(richardsonExtrapolation(I64, I128, 2) /
T);
    std::cout << "Trapezoidal Extrapolated IRMS: " << trapezoidalIRMS
              << ", Error = " << fabs(TRUE_IRMS - trapezoidalIRMS) << "\n";

    // Simpson's Method
    double I16 = simpsonsRule(16);
    double I32 = simpsonsRule(32);
    double simpsonsIRMS = sqrt(richardsonExtrapolation(I16, I32, 4) / T);
    std::cout << "Simpson's Extrapolated IRMS: " << simpsonsIRMS
              << ", Error = " << fabs(TRUE_IRMS - simpsonsIRMS) << "\n";

    return 0;
}
```

Output Clear

```
/tmp/5rQwoP3JOF.o
work estimated by integral for unequally spaced data is 135.324
true error of the corresponding value is 4.48139%


=== Code Execution Successful ===
```

Page 2 Q1,Q2

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>  // for std::setprecision

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// Define the function i(t)
double i_t(double t) {
    if (t <= 1.0 / 2) {
        return 10 * exp(-t / 1) * sin(2 * M_PI * t / 1);
    } else {
        return 0;
    }
}

// Define the square of the function i(t)
double i_squared(double t) {
    double current = i_t(t);
    return current * current;
}

// Trapezoidal Rule function
double trapezoidal_rule(int num_segments) {
    double T = 1.0;
    double h = T / num_segments;
    double integral = 0.5 * i_squared(0) + 0.5 * i_squared(T);

    for (int i = 1; i < num_segments; i++) {
        double t = i * h;
        integral += i_squared(t);
    }

    integral *= h;
    return integral;
}

// Simpson's 1/3 Rule function
double simpsons_rule(int num_segments) {
```

```
    if (num_segments % 2 != 0) {
        std::cerr << "Simpson's rule requires an even number of
intervals.\n";
        return 0;
    }

    double T = 1.0;
    double h = T / num_segments;
    double integral = i_squared(0) + i_squared(T);

    for (int i = 1; i < num_segments; i++) {
        double t = i * h;
        integral += (i % 2 == 0 ? 2 : 4) * i_squared(t);
    }

    integral *= h / 3.0;
    return integral;
}

int main() {
    int n_values_trap[] = {1, 2, 4, 8, 16, 32, 64, 128};
    int n_values_simp[] = {2, 4, 8, 16, 32};
    double true_value_trap = trapezoidal_rule(128); // Reference value for
error
    double true_value_simp = simpsons_rule(32);     // Reference value for
error

    // Set output to fixed-point notation with 8 decimal precision
    std::cout << std::fixed << std::setprecision(8);

    std::cout << "Trapezoidal Rule Results:\n";
    for (int i = 0; i < 8; i++) {
        int num_segments = n_values_trap[i];
        double integral = trapezoidal_rule(num_segments);
        double error = fabs(100 * (true_value_trap - integral) /
true_value_trap);
        std::cout << "Segments: " << num_segments << ", Integral: " <<
integral << ", Error: " << error << "\n";
    }

    std::cout << "\nSimpson's 1/3 Rule Results:\n";
    for (int i = 0; i < 5; i++) {
        int num_segments = n_values_simp[i];
```

```
        double integral = simpsons_rule(num_segments);
        double error = fabs(100 * (true_value_simp - integral) /
true_value_simp);
        std::cout << "Segments: " << num_segments << ", Integral: " <<
integral << ", Error: " << error << "\n";
    }

    double inte1 = trapezoidal_rule(64);
    double inte2 = trapezoidal_rule(124);
    double inte3 = (4 * inte1 - inte2) / 3;
    std::cout << "\nRichardson's extrapolation for Trapezoidal Method: " <<
inte3
              << " with error: " << fabs(100 * (true_value_trap - inte3) /
true_value_trap) << "\n";

    double inte4 = simpsons_rule(16);
    double inte5 = simpsons_rule(32);
    double inte6 = (16 * inte4 - inte5) / 15;
    std::cout << "Richardson's extrapolation for Simpson's 1/3 rule: " <<
inte6
              << " with error: " << fabs(100 * (true_value_simp - inte6) /
true_value_simp) << "\n";

    return 0;
}
```

Output Clear

```
ERROR!
/tmp/fiGre4MWOI.o
Trapezoidal Rule Results:
Segments: 1, Integral: 0.00000000, Error: 100.00000000
Segments: 2, Integral: 0.00000000, Error: 100.00000000
Segments: 4, Integral: 15.16326649, Error: 1.61777554
Segments: 8, Integral: 15.40142910, Error: 0.07253022
Segments: 16, Integral: 15.41195836, Error: 0.00421430
Segments: 32, Integral: 15.41256815, Error: 0.00025785
Segments: 64, Integral: 15.41260557, Error: 0.00001510
Segments: 128, Integral: 15.41260789, Error: 0.00000000

Simpson's 1/3 Rule Results:
Segments: 2, Integral: 0.00000000, Error: 100.00000000
Segments: 4, Integral: 20.21768866, Error: 31.17490756
Segments: 8, Integral: 15.48081663, Error: 0.44148591
Segments: 16, Integral: 15.41546811, Error: 0.01749653
Segments: 32, Integral: 15.41277142, Error: 0.00000000

Richardson's extrapolation for Trapezoidal Method: 15.41260480 with error: 0.00002009
Richardson's extrapolation for Simpson's 1/3 rule: 15.41564789 with error: 0.01866296


=== Code Execution Successful ===
```

x	F(x)	θ	F(x)cosθ		Ans		
0	0	0.5	0		139.93425		
5	9	1.4	1.5297		True Value		
10	13	0.75	9.512		129.52		
15	14	0.9	8.7025		Percentage Relative Error		
20	10.5	1.3	2.8087		8.04065009		
25	12	1.48	1.0881				
30	5	1.5	0.3537				