Assignment 5

Abhishek Agrahari
190123066

1

1.1 Code

```
% we have to calculate g(1) which means integral 0 to 1 e^(-x^2)
clc;
clear;
a = 0;
b = 1;
actual_area = 0.74682413;
err = @(a,b) abs(a-b);

fprintf(" N \t\t\t\t Rrule \t\t\t\t Trule \t\t\t\t Srule " + ...
    "\t\t\t\t Er \t\t\t\t Et \t\t\t\t Es\n")
for N = [50, 100, 200]
    h = (b-a)/N;
    Rectangle_tot = 0;
    Trapezoidal_tot = 0;
    Simpson_tot = 0;
    for i = 1:N
        a1 = a + (i-1)*h;
        a2 = a + i*h;
        Rectangle_tot = Rectangle_tot + left_rectangle_rule(a1, a2);
        Trapezoidal_tot = Trapezoidal_tot + trapezoidal_rule(a1, a2);
        Simpson_tot = Simpson_tot + simpson_rule(a1,a2);
    end
    fprintf("%3.0f \t\t %.10f \t\t %.10f \t\t %.10f \t\t" + ...
        " %.10f \t\t %.10f \t\t %.10f\n", ...
        N, Rectangle_tot, Trapezoidal_tot, Simpson_tot, ...
        err(actual_area, Rectangle_tot), ...
        err(actual_area, Trapezoidal_tot), ...
        err(actual_area, Simpson_tot));
end


function val = f(x)
    val = exp(-(x^2));
```

```
end

function val = left_rectangle_rule(a,b)
    val = (b-a)*f(a);
end

function val = trapezoidal_rule(a,b)
    val = ((b-a)*(f(a) + f(b)))/2;
end

function val = simpson_rule(a,b)
    val = ((b-a)*(f(a) + 4*f((a + b)/2)+ f(b)))/6;
end
```

1.2 Output

N	Rrule	Trule	Srule	Er	Et	Es
50	0.7531208128	0.7467996072	0.7468241329	0.0062966828	0.0000245228	0.0000000029
100	0.7499786043	0.7468180015	0.7468241328	0.0031544743	0.0000061285	0.0000000028
200	0.7484029014	0.7468226000	0.7468241328	0.0015787714	0.0000015300	0.0000000028

1.3 Note

- I have used Left rectangle rule.

1.4 Observations

- Error decreases as we go from Rectangle to Trapezoidal and Trapezoidal to Simpson rule.
    - This observation is in accordance with the order of their errors as given below -
        - Rectangle Rule : $O(h)$
        - Trapezoidal Rule : $O(h^2)$
        - Simpson Rule : $O(h^4)$
- Error decreases with increase in N.
    - This is because as N increases, h decreases which is proportional to the error.
- Actual value of the integral has 8 significant digits and value computed through Simpson rule is correct to 8 digits. Therefore Simpson rule produced the correct integral upto the desired decimal places.

2

2.1 Code

```
clc;
clear;
a = 0;
b = 1;
actual_area = 0.0808308690;
err = @(a,b) abs(a-b);

fprintf(" N \t\t\t    Trapezoidal \t   Corrected Trapezoidal \t  Err_Trapezoidal ...
 \t Err_Corrected_Trapezoidal\n")
for N = [1, 50, 100, 200]
    h = (b-a)/N;
    Trapezoidal_tot = 0;
    Corr_Trapezoidal_tot = 0;
    for i = 1:N
        a1 = a + (i-1)*h;
        a2 = a + i*h;
        Trapezoidal_tot = Trapezoidal_tot + trapezoidal_rule(a1, a2);
        Corr_Trapezoidal_tot = Corr_Trapezoidal_tot + corrected_trapezoidal_rule(a1,a2);
    end
    fprintf("%3.0f \t\t %.15f \t\t %.15f \t\t %.15f \t\t %.15f\n", ...
        N, Trapezoidal_tot, Corr_Trapezoidal_tot, ...
        err(actual_area, Trapezoidal_tot), err(actual_area, Corr_Trapezoidal_tot));
end

function val = f(x)
    val = (x^2)*exp(-2*x);
end

function val = f_prime(x)
    val = 2*exp(-2*x)*(x-x^2);
end

function val = trapezoidal_rule(a,b)
    val = ((b-a)*(f(a) + f(b)))/2;
end

function val = corrected_trapezoidal_rule(a,b)
    val = trapezoidal_rule(a,b) + ((b-a)^2)*(f_prime(a)-f_prime(b))/12;
end
```

2.2 Output

N	Trapezoidal	Corrected Trapezoidal	Err_Trapezoidal	Err_Corrected_Trapezoidal
1	0.067667641618306	0.067667641618306	0.013163227381694	0.013163227381694
50	0.080830893167599	0.080830893167599	0.000000024167599	0.000000024167599
100	0.080830895780054	0.080830895780054	0.000000026780054	0.000000026780054
200	0.080830895943348	0.080830895943348	0.000000026943348	0.000000026943348

2.3 Note

- N = 1 corresponds to non-composite method.
- In the observations section below I have used the following conventions -
  - * $f(x) = x^2.e^{-2x}$
  - * $a = 0$
  - * $b = 1$

2.4 Observations

- Error decreases with increase in N.
  - * This is because as N increases, h decreases which is proportional to the error.
- For all values of N, the value of integral computed through Trapezoidal and Corrected Trapezoidal rule is same.
  - * For N = 1, this is because error term given by the following expression, is zero.

$$Correction\ Term = \frac{(b-a)^2(f'(a)-f'(b))}{12}$$

    - ◦ Derivative of f(x) is $f'(x) = 2.e^{-2x}(x-x^2)$ and $f'(0) = f'(1)$
  - * For N > 1, error term given by the following expression, is zero.

$$Correction\ Term = \frac{h^2(f'(a)-f'(b))}{12}$$

    - ◦ This highlights the fact that even in the composite case, correction term involves derivative at ends only. Derivative terms for the points in the middle cancels out.

3

3.1 Code

```
clc;
clear;
fun = @(x) exp((x.^2)./(-2));
err = @(a,b) abs(a-b);
for m = [1,2]
    fprintf("\nFor m = %d\n", m)
    fprintf(" N \t\t\t\t Srule \t\t\t\t\t Error \t\t\t\t\t Relative Error\n")
    a = -m;
    b = m;
    actual_prob = integral(fun, a, b)*(1/sqrt(2*pi));
    for N = [1, 50, 100, 200]
        h = (b-a)/N;
        Simpson_tot = 0;
        for i = 1:N
            a1 = a + (i-1)*h;
            a2 = a + i*h;
            Simpson_tot = Simpson_tot + simpson_rule(a1,a2);
        end
        Req_prob = Simpson_tot*(1/sqrt(2*pi));
        fprintf("%3.0f \t\t %.15f \t\t %.15f \t\t %.15f\n", N, Req_prob, ...
         err(actual_prob, Req_prob), err(actual_prob, Req_prob)/actual_prob);
    end

    fprintf("Actual Value of Required Probability : %.15f \n", actual_prob);
end

function val = f(x)
    val = exp((x^2)/(-2));
end

function val = simpson_rule(a,b)
    val = ((b-a)*(f(a) + 4*f((a + b)/2)+ f(b)))/6;
end
```

3.2 Output

```
For m = 1
 N              Srule                  Error                  Relative Error
  1          0.693236856881339       0.010547364744253       0.015449724751491
 50          0.682689492997549       0.000000000860463       0.000000001260402
100          0.682689492190859       0.000000000053773       0.000000000078767
200          0.682689492140447       0.000000000003361       0.000000000004923
Actual Value of Required Probability : 0.682689492137086

For m = 2
 N              Srule                  Error                  Relative Error
  1          1.135834036421404       0.181334300317763       0.189978366110384
 50          0.954499733026886       0.000000003076756       0.000000003223422
100          0.954499735911592       0.000000000192050       0.000000000201205
200          0.954499736091643       0.000000000011998       0.000000000012570
Actual Value of Required Probability : 0.954499736103642
```

3.3 Note

- N = 1 corresponds to non-composite method.
- Relative error is calculated from the following formula -

$$RelativeError = \left| \frac{\textit{Actual Integral - Computed Integral}}{\textit{Actual Integral}} \right|$$

- Acutal Integral is calculated by an inbuilt function of matlab

```
integral(function_to_be_integrated, lower_limit, upper_limit)
```

- Value of the probability is computed using the following formula -

$$P(-m\sigma \leq x \leq m\sigma) = \frac{1}{\sqrt{2\pi}} \int_{-m}^{m} e^{\frac{-z^2}{2}} dz$$

3.4 Observations

- Error decreases with increase in N.
  - * This is because as N increases, h decreases which is proportional to the error.
- Relative error for m = 2 is greater than the relative error for m = 1.
  - * Therefore we can say that as the length of integration increases, value of error also increases.
- Error for non composite method (N = 1) is quite high.