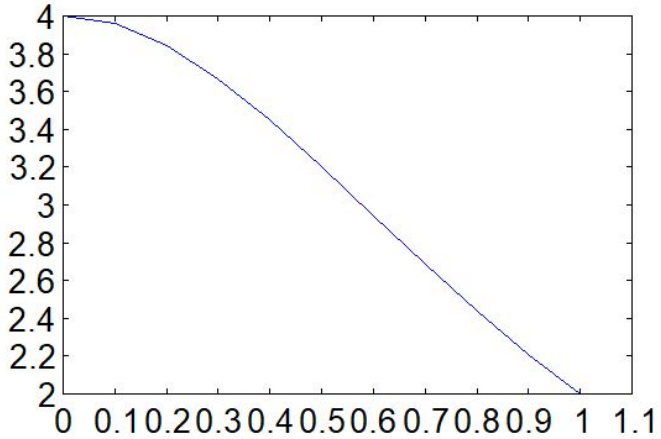
Computer Project 1
MTH 264
Prof. Kumnit Nong

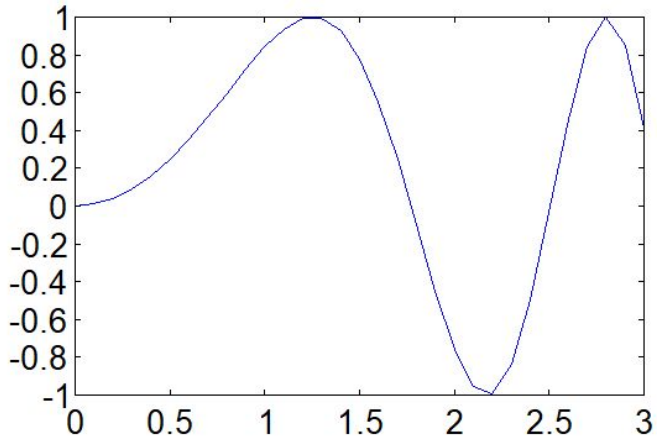
Kemal Ficici
Faruk Yaylagul

1. **Plot each of the given functions:**

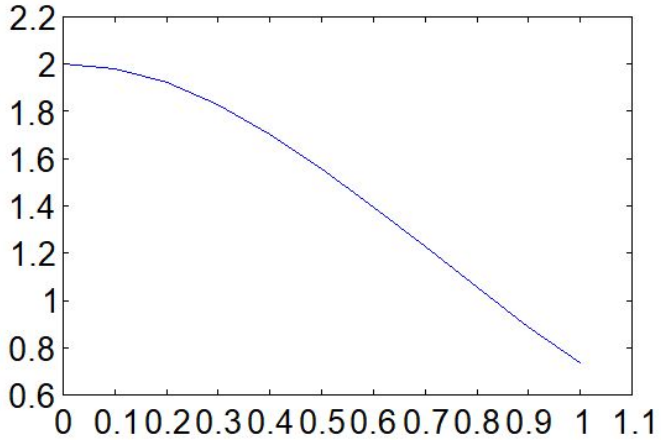
y = 4/(x^2 + 1)



y = sin(x^2)



y = e^(-x^2)



2. Evaluate the following integrals using the number of partitions N = 2, 10, 100, 1000, 10000. Use the command format long to display 12 decimal places, and store results in a table.

2a. $\int_0^1 \frac{4}{x^2+1} dx$

n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	3.60000000000	2.60000000000	3.16235294118	3.10000000000	3.13333333333
10	3.23992598891	3.03992598891	3.14242598500	3.13992598891	3.14159261394
100	3.15157598692	3.13157598692	3.14160098692	3.14157598692	3.14159265359
1000	3.14259248692	3.14059248692	3.14159273692	3.14159248692	3.14159265359
10000	3.14169265192	3.14149265192	3.14159265442	3.14159265192	3.14159265359

2b. $\int_0^3 sin(x^2) dx$

n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	1.16710979533	1.78528752319	-0.60904794771	1.47619865926	1.76220563640
10	0.66857599952	0.79221154509	0.79597119453	0.73039377230	0.78347809363
100	0.76697053616	0.77933409072	0.77376771044	0.77315231344	0.77356334815
1000	0.77294024906	0.77417660452	0.77356457695	0.77355842679	0.77356252698
10000	0.77350066812	0.77362430367	0.77356254739	0.77356248589	0.77356252689

2c. $\int_0^1 2e^{-x^2} dx$

n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	1.77880078307	1.14668022424	1.50919588754	1.46274050366	1.49436085782
10	1.55563364815	1.42920953638	1.49426175550	1.49242159226	1.49364989651
100	1.49995720852	1.48731479735	1.49365439698	1.49363600294	1.49364826579
1000	1.49428026356	1.49301602244	1.49364832694	1.49364814300	1.49364826562
10000	1.49371147645	1.49358505234	1.49364826624	1.49364826440	1.49364826562

3. Explain your discoveries and show any trends of solutions with respect to N. How many digits do you think you got right?

We compared the change in of the integral with respect to N, and noticed that our solutions converged with higher values of N. This is due to the higher number of partitions giving us a better approximation of the actual integral of the functions. However, certain methods

converged quicker than others. For instance, the trapezoid rule and the midpoint rules converged much quicker than the Upper and Lower sums, due to the fact that they provide closer approximations than the Upper and Lower sums'. However, the convergence rates of each of these methods were paled in comparison to Simpson's Rule.

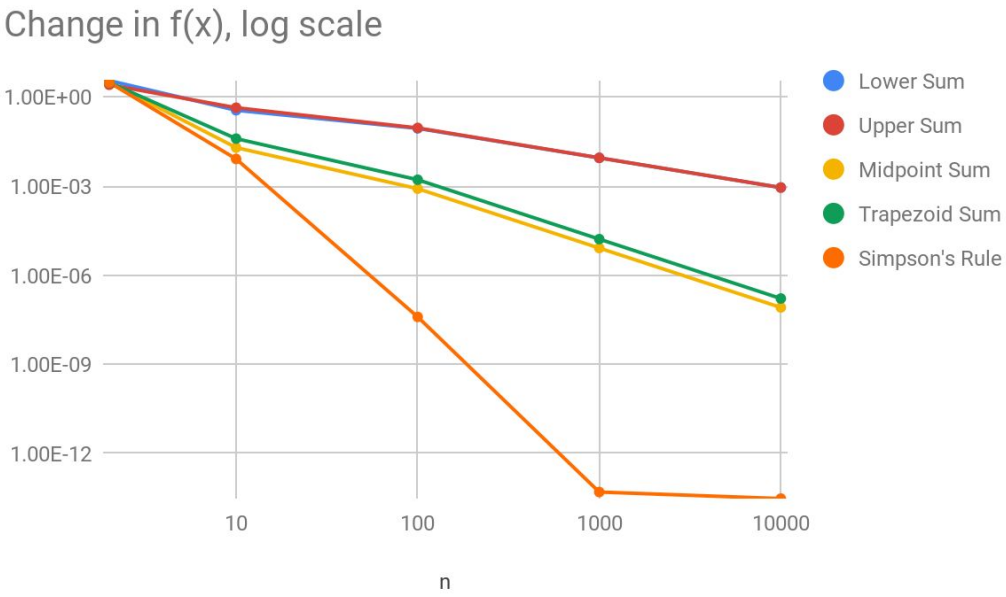
Simpson's rule uses parabolas to approximate portions of the function, allowing for very close approximations of the integral. In fact, it got over 12 digits correct for N=1000, according to our results.

We predicted the number of correct digits by looking at the change in the sums we got for different values of N. If digits remain the same as N increases, we assume that that digit is correct, since the difference between the actual value and the sum converges as N increases. For example, if the difference of f(x) between N1 and N2 is 5.00E-02, we will round the difference to 1.00E-01, and conclude that only every digit before the tenths place has is its correct value.

Here are our results for the number of correct digits produced by each method when N=1000:

Function	Lower	Upper	Midpoint	Trapezoidal	Simpson
$\int_0^1 \frac{4}{x^2+1} dx$	3	3	7	7	12
$\int_0^3 sin(x^2) dx$	3	3	6	5	10
$\int_0^1 2e^{-x^2} dx$	3	3	7	7	12

To determine the accuracy of N=10000, we'd need to calculate the approximate integral of the functions using a higher value of N. Our best prediction is that the number of correct digits when N=10000 are either greater than or equal to the correct digits when N=1000.


Change in f(x), log scale
1.00E+00
1.00E-03
1.00E-06
1.00E-09
1.00E-12
10
100
1000
10000
n
Lower Sum
Upper Sum
Midpoint Sum
Trapezoid Sum
Simpson's Rule

$\Delta\int_0^1 \frac{4}{x^2+1}dx$					
n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	3.60E+00	2.60E+00	3.16E+00	3.10E+00	3.13E+00
10	3.60E-01	4.40E-01	1.99E-02	3.99E-02	8.26E-03
100	8.84E-02	9.16E-02	8.25E-04	1.65E-03	3.97E-08
1000	8.98E-03	9.02E-03	8.25E-06	1.65E-05	5.02E-14
10000	9.00E-04	9.00E-04	8.25E-08	1.65E-07	3.02E-14
$\Delta\int_0^3 sin(x^2)dx$					
n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	1.17E+00	1.79E+00	-6.09E-01	1.48E+00	1.76E+00
10	4.99E-01	9.93E-01	1.41E+00	7.46E-01	9.79E-01
100	9.84E-02	1.29E-02	2.22E-02	4.28E-02	9.91E-03
1000	5.97E-03	5.16E-03	2.03E-04	4.06E-04	8.21E-07
10000	5.60E-04	5.52E-04	2.03E-06	4.06E-06	8.14E-11
$\Delta\int_0^1 2e^{-x^2}dx$					
n	Lower Sum	Upper Sum	Midpoint Sum	Trapezoid Sum	Simpson's Rule
2	1.78E+00	1.15E+00	1.51E+00	1.46E+00	1.49E+00
10	2.23E-01	2.83E-01	1.49E-02	2.97E-02	7.11E-04
100	5.57E-02	5.81E-02	6.07E-04	1.21E-03	1.63E-06
1000	5.68E-03	5.70E-03	6.07E-06	1.21E-05	1.63E-10
10000	5.69E-04	5.69E-04	6.07E-08	1.21E-07	3.00E-14

Code

Lower Sum:

```
clear
clc
format long

% Assigned Functions
f1 = @(x) (4/(x.^2 + 1));
f2 = @(x) (sin(x.^2));
f3 = @(x) (2 * exp(-1 * x.^2));


f = f1;
a=0;
b=1;

numBoxes = [2; 10; 100; 1000; 10000; ];

lowerSum = [];


for n = numBoxes;
    dx = (b-a)/ n;

    % Lower Sum
    x = a:dx:b-dx;
    lowerSum = [lowerSum; sum(f(x)) * dx];
end

% Display outputs
disp(' Number of Partitions: ')
disp(numBoxes)
disp('Lower Sum')
disp(lowerSum)
```

Upper Sum:

```
clear
clc
format long

% Assigned Functions
f1 = @(x) (4/(x.^2 + 1));
f2 = @(x) (sin(x.^2));
f3 = @(x) (2 * exp(-1 * x.^2));


f = f1;
a=0;
b=1;

numBoxes = [2; 10; 100; 1000; 10000; ];

upperSum = [];


for n = numBoxes;
    dx = (b-a)/ n;

    % Upper Sum
    x = a+dx:dx:b;
    upperSum = [upperSum; sum(f(x)) * dx];
end

% Display outputs
disp(' Number of Partitions: ')
disp(numBoxes)
disp('Upper Sum')
disp(upperSum)


```

Midpoint Sum:

```
clear
clc
format long

% Assigned Functions
f1 = @(x) (4/(x.^2 + 1));
f2 = @(x) (sin(x.^2));
f3 = @(x) (2 * exp(-1 * x.^2));


f = f1;
a=0;
b=1;

numBoxes = [2; 10; 100; 1000; 10000; ];

midSum = [];


for n = numBoxes;
    dx = (b-a)/ n;

    % Sum using MidPoint Rule
    x = a:dx:b-dx;
    midSum = [midSum; sum(f(x + (dx/2))) * dx]
end

% Display outputs
disp(' Number of Partitions: ')
disp(numBoxes)
disp('Midpoint Sum')
disp(midSum)


```

Trapezoid Sum:

```
clear
clc
format long

% Assigned Functions
f1 = @(x) (4/(x.^2 + 1));
f2 = @(x) (sin(x.^2));
f3 = @(x) (2 * exp(-1 * x.^2));


f = f1;
a=0;
b=1;

numBoxes = [2; 10; 100; 1000; 10000; ];

trapezoidSum = [];


for n = numBoxes;
    dx = (b-a)/ n;

    % Sum using Trapezoid Rule
    x = a+dx:dx:b-dx;
    trapezoidSum = [trapezoidSum; (f(a) + 2*sum(f(x)) + f(b) ) * dx / 2];
end

% Display outputs
disp(' Number of Partitions: ')
disp(numBoxes)
disp('Trapezoid Sum')
disp(trapezoidSum)

```

Simpson’s Rule:

```
clear
clc
format long

% Assigned Functions
f1 = @(x) (4/(x.^2 + 1));
f2 = @(x) (sin(x.^2));
f3 = @(x) (2 * exp(-1 * x.^2));


f = f1;
a=0;
b=1;

numBoxes = [2; 10; 100; 1000; 10000; ];

simpsonSum = [];


for n = numBoxes;
    dx = (b-a)/ n;

    % Sum Using Simpsons Rule
    x = a+dx :dx: b-dx;
    x1 = x(1:2:end); %Every even Interval
    x2 = x(2:2:end); %Every odd Interval
    simpsonSum = [simpsonSum; (((4*sum(f(x1))) + (2*sum(f(x2))) + f(a) + f(b))* (dx / 3))];
end

% Display outputs
disp(' Number of Partitions: ')
disp(numBoxes)
disp('Simpsons Rule')
disp(simpsonSum)

```

References:

- Calculus Textbook, p. 562 - Simpson's Rule