

MTH 264 Project II

Kemal Ficici
Faruk Yaylagul

November 4, 2018

1 Problems

Use the following methods to approximate definite integrals:

1. Simpson's Rule
2. Composite Midpoint Rule
3. Alternative Extended Simpson's Rule

1.1 Integral Approximation

Use the listed methods to approximate the following integrals. Find the minimum N partitions to yield 4 decimal places correctly.

Note: Only use even values of N , starting with $N=8$.

$a=0.0000000001$

(a) $\int_a^{\pi/2} \frac{x}{\sin(x)} dx$

(b) $\int_a^{\pi/2} \frac{e^x-1}{\sin(x)} dx$

(c) $\int_a^1 \frac{\arcsin x}{x} dx$

1.2 Arclength

Use the listed methods to compute the arclength of $f(x)$ between $[a, b]$. Find the minimum N partitions to yield 4 decimal places correctly.

(a) $f(x) = \frac{x}{\sin(x)}$ from $[a, \frac{\pi}{2}]$

(b) $f(x) = \frac{e^x-1}{\sin(x)}$ from $[a, \frac{\pi}{2}]$

(c) $f(x) = \frac{\arcsin x}{x}$ from $[a, 1]$

1.3 Volume

Use the listed methods to compute the volume of the solid generated by rotating $f(x)$ between $[a, b]$ from the previous problem along the x-axis. Find the minimum N partitions to yield 4 decimal places correctly.

2 Solutions

2.1 Integral Approximation

Integral approximation was done using the assigned methods. The table below lists the amount of N partitions required to reach 4 “correct” decimal places. There is no guarantee that the approximation methods converged on the correct solution, however we assume the digit to be “correct” if it hasn’t changed from the previous iteration with $N-2$ partitions.

Table 1: Minimum N to yield 4 ”correct” decimal places

	Simpson’s Rule	Comp. Mid.	Alt. Ext. Simpson
$\int_a^{\pi/2} \frac{x}{\sin(x)} dx$	10	12	10
$\int_a^{\pi/2} \frac{e^x-1}{\sin(x)} dx$	12	20	12
$\int_a^1 \frac{\arcsin x}{x} dx$	32	26	34

2.2 Arclength

The arclength formula is as follows:

$$\int_a^b \sqrt{1 + f'(x)^2} dx$$

We use the integral approximation methods used in the previous problem to approximate this integral.

Supplying the double derivatives for the Composite Midpoint Rule when approximating the integral for arclength proved tedious, so the double derivative was instead approximated.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \tag{1}$$

$$f''(x) \approx \frac{f'(x+h) - f'(x)}{h}$$

The table below lists the amount of N partitions required to reach 4 “correct” decimal places.

Table 2: Minimum N to yield 4 ”correct” decimal places

	Simpson’s Rule	Comp. Mid.	Alt. Ext. Simpson
$f(x) = x/\sin(x)$	10	16	10
$f(x) = \frac{e^x-1}{\sin(x)}$	96	30	46
$f(x) = \frac{\arcsin(x)}{x}$	552	290	556

2.3 Volume

The volume of a solid generated by rotating the area under a curve about the x-axis can be determined using the following method:

$$V = \pi \int_a^b R(x)^2 dx$$

Where $R(x)$ is the function defining the distance between the function and the axis of rotation.

The double derivative approximation method used in the previous problem was also used.

Using the integral approximation methods used in the previous problems, we can approximate the volume using the formulas. The table below lists the amount of N partitions required to reach 4 “correct” decimal places.

Table 3: Minimum N to yield 4 ”correct” decimal places

	Simpson’s Rule	Comp. Mid.	Alt. Ext. Simpson
$R(x) = x/\sin(x)$	14	30	24
$R(x) = \frac{e^x-1}{\sin(x)}$	26	70	24
$R(x) = \frac{a\sin(x)}{x}$	68	56	78

3 Code

3.1 Problem 1: Integral Approximation

3.1.1 Simpson's Rule

```
1 % Problem 1
2 % Simpson's rule
3
4 clear
5 clc
6 format long
7
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14 f = f3; % set which function you want to integrate
15 a = 0.0000000001;
16 b = 1; % pi/2, 1
17
18 simpson = @(x,dx) f(x) + 4*f(x + dx) + f(x + 2*dx);
19 n = 8;
20 H = 0;
21 Aold = 0;
22 dx = (b-a)/n;
23
24 for i = 1:2:n
25     xL = a+(i-1)*dx;
26     H = H+simpson(xL, dx);
27 end
28
29 Anew = H*dx /3;
30 error = 1;
31
32 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
33 while error != 0
34     % compare how many digits are "incorrect"
35     % Anew - Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
36     Aold = Anew;
37     n = n+2;
38     dx = (b-a)/n;
39     H=0;
40     for g = 1:2:n
41         xL = a+(g-1)*dx;
42         H = H+simpson(xL, dx);
43     end
44     Anew = H*dx/3;
45     error = abs(rounddown(Anew) - rounddown(Aold));
46 end
47
48 disp(n)
49 disp(Aold)
50 disp(Anew)
```

3.1.2 Composite Midpoint Rule

```
1 % Problem 1
2 % Composite Midpoint Rule
3
4 clear
5 clc
6 format long
7
8
9
10
11 % Assigned Functions
12 f1 = @(x) (x / sin(x));
13 f2 = @(x) ((exp(x)-1)/sin(x));
14 f3 = @(x) (asin(x)/x);
15
16 % double derivatives of functions
17 ddf1 = @(x) (csc(x)*(x*cot(x)^2 - 2*cot(x) + x*csc(x)^2));
18 ddf2 = @(x) (csc(x)*(exp(x) - 2*exp(x)*cot(x) + ((exp(x) - 1)*(cot(x)^2 + csc(x)^2))));
19 ddf3 = @(x) ((2*asin(x))/x^3 - 2/(x^2 * sqrt(1-x^2)) + 1/sqrt((1-x^2)^3));
20
21 % set which function you want to integrate.
22 f = f3;
23 ddf = ddf3;
24
25
26 n = 6; % n=8 once we get into the loop
27 a = 0.0000000001;
28 b = 1;% pi/2, 1
29
30 prev = 0;
31
32
33 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
34 error = 1;
35 while error > 0
36     n = n+2;
37     dx = (b-a)/n;
38
39     smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
40     bigsum = 0;
41     for g = a+dx:dx:b
42         bigsum = bigsum + f((2*g-dx)/2);
43     end
44     compositeMid = smallsum + dx*bigsum;
45
46
47     error = rounddown(compositeMid) - rounddown(prev);
48     if error > 0 % this is here so we can compare values manually(sanity check)
49         prev = compositeMid;
50     end
51 end
52
53
54 disp(n)
55 disp(prev)
56 disp(compositeMid)
```

3.1.3 Alternative Extended Simpson's Rule

```
1 % Problem 1
2 % Alternative Extended Simpson's Rule
3 %
4
5 clear
6 clc
7 format long
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14 % set the function you want to integrate
15 f = f3;
16 a = 0.0000000001;
17 b = 1; % pi/2, 1
18
19 n = 6; % n=8 once we get into the loop
20 error = 1;
21 prev = 0;
22
23
24 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
25 while error != 0
26     n = n+2;
27     dx = (b-a)/n;
28     smallsum = 17*f(a) + 59*f(a+dx) + 43*f(a+2*dx) + 49*f(a+3*dx) + 49*f(b-3*dx) + 43*f(b-2*dx) + ...
29         59*f(b-dx) + 17*f(b);
30     bigsum = 0;
31     for i = a+4*dx:dx:b-4*dx
32         bigsum = bigsum + f(i);
33     end
34     simpson = (dx/48) * (smallsum + 48*bigsum);
35     error = rounddown(simpson) - rounddown(prev);
36     if error != 0 % this is here so we can compare values manually(sanity check)
37         prev = simpson;
38     end
39 end
40
41
42 disp(n)
43 disp(prev)
44 disp(simpson)
```

3.2 Problem 2: Arclength

3.2.1 Simpson's Rule

```
1 % Problem 2
2 % Simpson's rule
3
4 clear
5 clc
6 format long
7
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14 % Derivatives of assigned functions
15 df1 = @(x) (1-x*cot(x))*csc(x);
16 df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
17 df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);
18
19
20 f = df1; % set which function you want to integrate
21 arclen = @(x) sqrt(1+(f(x))^2)
22
23 a = 0.00000001;
24 b = pi/2;%0.9999; % pi/2, 0.9999
25
26 simpson = @(x,dx) arclen(x) + 4*arclen(x + dx) + arclen(x + 2*dx);
27 n = 8;
28 H = 0;
29 Aold = 0;
30 dx = (b-a)/n;
31
32 for i = 1:2:n
33     xL = a+(i-1)*dx;
34     H = H+simpson(xL, dx);
35 end
36
37 Anew = H*dx /3;
38 error = 1;
39
40 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
41 while error != 0
42     % compare how many digits are "incorrect"
43     % Anew - Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
44     Aold = Anew;
45     n = n+2;
46     dx = (b-a)/n;
47     H=0;
48     for g = 1:2:n
49         xL = a+(g-1)*dx;
50         H = H+simpson(xL, dx);
51     end
52     Anew = H*dx/3;
53     error = abs(rounddown(Anew) - rounddown(Aold));
54 end
55
56 disp(n)
57 disp(Aold)
58 disp(Anew)
```


3.2.2 Composite Midpoint Rule

```
1 % Problem 2
2 % Composite Midpoint Rule
3
4 clear
5 clc
6 format long
7
8
9
10
11 % Assigned Functions
12 f1 = @(x) (x / sin(x));
13 f2 = @(x) ((exp(x)-1)/sin(x));
14 f3 = @(x) (asin(x)/x);
15
16 % derivatives of assigned functions
17 df1 = @(x) (1-x*cot(x))*csc(x);
18 df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
19 df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);
20
21
22
23
24 % set which function you want to integrate.
25 f = df2;
26
27 n = 6; % n=8 once we get into the loop
28 a = 0.00000001;
29 b = pi/2;% pi/2, 1
30 dx= 0;
31
32
33 h = 0.000001;
34 df = @(x) (f(x+h)-f(x))/(h); %simpler solution; we can approximate the derivative!
35
36 arclen = @(x) sqrt(1+(f(x)^2)); % change f(x) to df(x) if you want to approximate instead of ...
    supplying f'(x)
37
38 dfarclen = @(x) (arclen(x+h)-arclen(x))/(h);
39 ddf = @(x) (dfarclen(x+h)-dfarclen(x))/(h); % approximate double derivative
40
41
42 prev = 0;
43
44
45 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
46 error = 1;
47 while error != 0
48     n = n+2;
49     dx = (b-a)/n;
50     smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
51     bigsum = 0;
52     for g = a+dx:dx:b
53         bigsum = bigsum + arclen((2*g-dx)/2);
54     end
55     compositeMid = smallsum + dx*bigsum;
56     error = rounddown(prev) - rounddown(compositeMid);
57     if error != 0
58         prev = compositeMid;
59     end
60 end
61
62
63 disp(n)
64 disp(prev)
65 disp(compositeMid)
```

3.2.3 Alternative Extended Simpson's Rule

```
1 % Problem 2
2 % Alternative Extended Simpson's Rule
3 %
4
5 clear
6 clc
7 format long
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14 % Derivatives of assigned functions
15 df1 = @(x) (1-x*cot(x))*csc(x);
16 df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
17 df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);
18
19
20 f = df3; % set which function you want to integrate
21 arclen = @(x) sqrt(1+(f(x))^2)
22
23 a = 0.0000000001;
24 b = 0.9999; % pi/2, 0.9999
25
26 n = 6; % n=8 once we get into the loop
27 error = 1;
28 prev = 0;
29
30
31 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
32 while error != 0
33     n = n+2;
34     dx = (b-a)/n;
35     smallsum = 17*arclen(a) + 59*arclen(a+dx) + 43*arclen(a+2*dx) + 49*arclen(a+3*dx) ...
36         +49*arclen(b-3*dx) + 43*arclen(b-2*dx) + 59*arclen(b-dx) + 17*arclen(b);
37     bigsum = 0;
38     for i = a+4*dx:dx:b-4*dx
39         bigsum = bigsum + arclen(i);
40     end
41     simpson = (dx/48) * (smallsum + 48*bigsum);
42     error = rounddown(simpson) - rounddown(prev);
43     if error != 0 % this is here so we can compare values manually(sanity check)
44         prev = simpson;
45     end
46 end
47
48
49 disp(n)
50 disp(prev)
51 disp(simpson)
```

3.3 Problem 3: Volume

3.3.1 Simpson's Rule

```
1 % Problem 3
2 % Simpson's rule
3
4 clear
5 clc
6 format long
7
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14
15 f = f2; % set which function you want to integrate
16
17 a = 0.00000001;
18 b = pi/2;%0.9999; % pi/2, 0.9999
19
20 R = @(x) f(x)^2;
21 simpson = @(x,dx) R(x) + 4*R(x + dx) + R(x + 2*dx);
22 n = 8;
23 H = 0;
24 Aold = 0;
25 dx = (b-a)/n;
26
27 for i = 1:2:n
28     xL = a+(i-1)*dx;
29     H = H+simpson(xL, dx);
30 end
31
32 Anew =pi* H*dx /3;
33 error = 1;
34
35 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
36 while error != 0
37     % compare how many digits are "incorrect"
38     % Anew - Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
39     Aold = Anew;
40     n = n+2;
41     dx = (b-a)/n;
42     H=0;
43     for g = 1:2:n
44         xL = a+(g-1)*dx;
45         H = H+simpson(xL, dx);
46     end
47     Anew =pi* H*dx/3;
48     error = abs(rounddown(Anew) - rounddown(Aold));
49 end
50
51 disp(n)
52 disp(Aold)
53 disp(Anew)
```

3.3.2 Composite Midpoint Rule

```
1 % Problem 3
2 % Composite Midpoint Rule
3
4 clear
5 clc
6 format long
7
8
9
10
11 % Assigned Functions
12 f1 = @(x) (x / sin(x));
13 f2 = @(x) ((exp(x)-1)/sin(x));
14 f3 = @(x) (asin(x)/x);
15
16 % set which function you want to integrate.
17 f = f3;
18
19 n = 6; % n=8 once we get into the loop
20 a = 0.00000001;
21 b = 1;% pi/2, 1
22 dx= 0;
23
24
25 h = 0.000001;
26 R = @(x) f(x)^2; % change f(x) to df(x) if you want to approximate instead of supplying f'(x)
27 df = @(x) (R(x+h)-R(x))/(h); %simpler solution; we can approximate the derivative!
28
29
30
31 ddf = @(x) (df(x+h)-df(x))/(h); % approximate double derivative
32
33
34 prev = 0;
35
36
37 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
38 error = 1;
39 while error != 0
40     n = n+2;
41     dx = (b-a)/n;
42     smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
43     bigsum = 0;
44     for g = a+dx:dx:b
45         bigsum = bigsum + R((2*g-dx)/2);
46     end
47     compositeMid = pi*(smallsum + dx*bigsum);
48     error = rounddown(prev) - rounddown(compositeMid);
49     if error != 0
50         prev = compositeMid;
51     end
52 end
53
54
55 disp(n)
56 disp(prev)
57 disp(compositeMid)
```

3.3.3 Alternative Extended Simpson's Rule

```
1 % Problem 3
2 % Alternative Extended Simpson's Rule
3 %
4
5 clear
6 clc
7 format long
8
9 % Assigned Functions
10 f1 = @(x) (x / sin(x));
11 f2 = @(x) ((exp(x)-1)/sin(x));
12 f3 = @(x) (asin(x)/x);
13
14 f = f2; % set which function you want to integrate
15
16
17 a = 0.0000000001;
18 b = pi/2; % pi/2, 0.9999
19
20 n = 6; % n=8 once we get into the loop
21 error = 1;
22 prev = 0;
23 R = @(x) f(x)^2;
24
25 rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
26 while error != 0
27     n = n+2;
28     dx = (b-a)/n;
29     smallsum = 17*R(a) + 59*R(a+dx) + 43*R(a+2*dx) + 49*R(a+3*dx) + 49*R(b-3*dx) + 43*R(b-2*dx) + ...
30         59*R(b-dx) + 17*R(b);
31     bigsum = 0;
32     for i = a+4*dx:dx:b-4*dx
33         bigsum = bigsum + R(i);
34     end
35     simpson = pi * (dx/48) * (smallsum + 48*bigsum);
36     error = rounddown(simpson) - rounddown(prev);
37     if error != 0 % this is here so we can compare values manually(sanity check)
38         prev = simpson;
39     end
40 end
41
42
43 disp(n)
44 disp(prev)
45 disp(simpson)
```