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.

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

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
$f(x) = \frac{asin(x)}{x}$	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
$R(x) = \frac{asin(x)}{x}$	68	56	78

3 Code

3.1 Problem 1: Integral Approximation

3.1.1 Simpson's Rule

```
% Problem 1
% Simpson's rule

clear
clc
format long


% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

f = f3; % set which function you want to integrate
a = 0.0000000001;
b = 1; % pi/2, 1

simpson = @(x,dx) f(x) + 4*f(x + dx) + f(x + 2*dx);
n = 8;
H = 0;
Aold = 0;
dx = (b-a)/n;

for i = 1:2:n
    xL = a+(i-1)*dx;
    H = H+simpson(xL, dx);
end

Anew = H*dx /3;
error = 1;

rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ~= 0
    % compare how many digits are "incorrect"
    % Anew - Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
    Aold = Anew;
    n = n+2;
    dx = (b-a)/n;
    H=0;
    for g = 1:2:n
        xL = a+(g-1)*dx;
        H = H+simpson(xL, dx);
    end
    Anew = H*dx/3;
    error = abs(rounddown(Anew) - rounddown(Aold));
end

disp(n)
disp(Aold)
disp(Anew)
```

3.1.2 Composite Midpoint Rule

```
% Problem 1
% Composite Midpoint Rule

clear
clc
format long




% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% double derivatives of functions
ddf1 = @(x) (csc(x)*(x*cot(x)^2 -2*cot(x)+ x*csc(x)^2));
ddf2 = @(x) (csc(x)*(exp(x) - 2*exp(x)*cot(x) + ((exp(x) - 1)*(cot(x)^2 + csc(x)^2))));
ddf3 = @(x) ((2*asin(x))/x^3 - 2/(x^2 * sqrt(1-x^2)) + 1/sqrt(( 1-x^2)^3));

% set which function you want to integrate.
f = f3;
ddf = ddf3;


n = 6; % n=8 once we get into the loop
a = 0.0000000001;
b = 1;% pi/2, 1

prev = 0;


rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
error = 1;
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;

    smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
    bigsum = 0;
    for g = a+dx:dx:b
        bigsum = bigsum + f((2*g-dx)/2);
    end
    compositeMid = smallsum + dx*bigsum;


    error = rounddown(compositeMid) - rounddown(prev);
    if error ≠ 0 % this is here so we can compare values manually(sanity check)
        prev = compositeMid;
    end
end


disp(n)
disp(prev)
disp(compositeMid)
```

3.1.3 Alternative Extended Simpson's Rule

```
% Problem 1
% Alternative Extended Simpson's Rule
%

clear
clc
format long

% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% set the function  you want to integrate
f = f3;
a = 0.0000000001;
b = 1; % pi/2, 1

n = 6; % n=8 once we get into the loop
error = 1;
prev = 0;


rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;
    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) + ...
        59*f(b-dx) + 17*f(b);
    bigsum = 0;

    for i = a+4*dx:dx:b-4*dx
        bigsum = bigsum + f(i);
    end
    simpson = (dx/48) * (smallsum + 48*bigsum);
    error = rounddown(simpson) - rounddown(prev);
    if error ≠ 0 % this is here so we can compare values manually(sanity check)
        prev = simpson;
    end
end


disp(n)
disp(prev)
disp(simpson)
```

3.2 Problem 2: Arclength

3.2.1 Simpson's Rule

```
% Problem 2
% Simpson's rule

clear
clc
format long


% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% Derivatives of assigned functions
df1 = @(x) (1-x*cot(x))*csc(x);
df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);


f = df1; % set which function you want to integrate
arclen = @(x) sqrt(1+(f(x))^2)

a = 0.00000001;
b = pi/2;%0.9999; % pi/2, 0.9999

simpson = @(x,dx) arclen(x) + 4*arclen(x + dx) + arclen(x + 2*dx);
n = 8;
H = 0;
Aold = 0;
dx = (b-a)/n;

for i = 1:2:n
    xL = a+(i-1)*dx;
    H = H+simpson(xL, dx);
end

Anew = H*dx /3;
error = 1;

rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ≠ 0
    % compare how many digits are "incorrect"
    % Anew -  Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
    Aold = Anew;
    n = n+2;
    dx = (b-a)/n;
    H=0;
    for g = 1:2:n
        xL = a+(g-1)*dx;
        H = H+simpson(xL, dx);
    end
    Anew = H*dx/3;
    error = abs(rounddown(Anew) - rounddown(Aold));
end

disp(n)
disp(Aold)
disp(Anew)
```

3.2.2 Composite Midpoint Rule

```
% Problem 2
% Composite Midpoint Rule

clear
clc
format long




% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% derivatives of assigned functions
df1 = @(x) (1-x*cot(x))*csc(x);
df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);




% set which function you want to integrate.
f = df2;

n = 6; % n=8 once we get into the loop
a = 0.00000001;
b = pi/2;% pi/2, 1
dx= 0;


h = 0.000001;
df = @(x) (f(x+h)-f(x))/(h); %simpler solution; we can approximate the derivative!

arclen = @(x) sqrt(1+(f(x)^2)); % change f(x) to df(x) if you want to approximate instead of ...
    supplying f'(x)

dfarclen = @(x) (arclen(x+h)-arclen(x))/(h);
ddf = @(x) (dfarclen(x+h)-dfarclen(x))/(h); % approximate double derivative


prev = 0;


rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
error = 1;
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;
    smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
    bigsum = 0;
    for g = a+dx:dx:b
        bigsum = bigsum + arclen((2*g-dx)/2);
    end
    compositeMid = smallsum + dx*bigsum;
    error = rounddown(prev) - rounddown(compositeMid);
    if error ≠ 0
        prev = compositeMid;
    end
end


disp(n)
disp(prev)
disp(compositeMid)
```

3.2.3 Alternative Extended Simpson's Rule

```
% Problem 2
% Alternative Extended Simpson's Rule
%

clear
clc
format long

% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% Derivatives of assigned functions
df1 = @(x) (1-x*cot(x))*csc(x);
df2 = @(x) (exp(x) + cot(x) -exp(x)*cot(x))*(csc(x));
df3 = @(x) ((x/(sqrt(1-x^2)))-asin(x))/(x^2);


f = df3; % set which function you want to integrate
arclen = @(x) sqrt(1+(f(x))^2)

a = 0.0000000001;
b = 0.9999; % pi/2, 0.9999

n = 6; % n=8 once we get into the loop
error = 1;
prev = 0;


rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;
    smallsum = 17*arclen(a) + 59*arclen(a+dx) + 43*arclen(a+2*dx) + 49*arclen(a+3*dx) ...
        +49*arclen(b-3*dx) + 43*arclen(b-2*dx) + 59*arclen(b-dx) + 17*arclen(b);
    bigsum = 0;

    for i = a+4*dx:dx:b-4*dx
        bigsum = bigsum + arclen(i);
    end
    simpson = (dx/48) * (smallsum + 48*bigsum);
    error = rounddown(simpson) - rounddown(prev);
    if error ≠ 0 % this is here so we can compare values manually(sanity check)
        prev = simpson;
    end
end


disp(n)
disp(prev)
disp(simpson)
```

3.3 Problem 3: Volume

3.3.1 Simpson's Rule

```
% Problem 3
% Simpson's rule

clear
clc
format long


% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);


f = f2; % set which function you want to integrate

a = 0.00000001;
b = pi/2;%0.9999; % pi/2, 0.9999

R = @(x) f(x)^2;
simpson = @(x,dx) R(x) + 4*R(x + dx) + R(x + 2*dx);
n = 8;
H = 0;
Aold = 0;
dx = (b-a)/n;

for i = 1:2:n
    xL = a+(i-1)*dx;
    H = H+simpson(xL, dx);
end

Anew =pi*  H*dx /3;
error = 1;

rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ≠ 0
    % compare how many digits are "incorrect"
    % Anew -  Aold > 0.0001 doesn't work since 1.00012 and 1.00009 would come out to be true
    Aold = Anew;
    n = n+2;
    dx = (b-a)/n;
    H=0;
    for g = 1:2:n
        xL = a+(g-1)*dx;
        H = H+simpson(xL, dx);
    end
    Anew =pi*  H*dx/3;
    error = abs(rounddown(Anew) - rounddown(Aold));
end

disp(n)
disp(Aold)
disp(Anew)
```

3.3.2 Composite Midpoint Rule

```
% Problem 3
% Composite Midpoint Rule

clear
clc
format long




% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

% set which function you want to integrate.
f = f3;

n = 6; % n=8 once we get into the loop
a = 0.00000001;
b = 1;% pi/2, 1
dx= 0;


h = 0.000001;
R = @(x) f(x)^2; % change f(x) to df(x) if you want to approximate instead of supplying f'(x)
df = @(x) (R(x+h)-R(x))/(h); %simpler solution; we can approximate the derivative!



ddf = @(x) (df(x+h)-df(x))/(h); % approximate double derivative


prev = 0;


rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
error = 1;
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;
    smallsum = (((b - a) * dx^2) / 24) * ddf((a+b)/2);
    bigsum = 0;
    for g = a+dx:dx:b
        bigsum = bigsum + R((2*g-dx)/2);
    end
    compositeMid = pi*(smallsum + dx*bigsum);
    error = rounddown(prev) - rounddown(compositeMid);
    if error ≠ 0
        prev = compositeMid;
    end
end


disp(n)
disp(prev)
disp(compositeMid)
```

3.3.3 Alternative Extended Simpson's Rule

```
% Problem 3
% Alternative Extended Simpson's Rule
%

clear
clc
format long

% Assigned Functions
f1 = @(x) (x / sin(x));
f2 = @(x) ((exp(x)-1)/sin(x));
f3 = @(x) (asin(x)/x);

f = f2; % set which function you want to integrate


a = 0.0000000001;
b = pi/2; % pi/2, 0.9999

n = 6; % n=8 once we get into the loop
error = 1;
prev = 0;
R = @(x) f(x)^2;

rounddown = @(x) floor(x * 10000); % modify this to set how many "correct" decimal places you want
while error ≠ 0
    n = n+2;
    dx = (b-a)/n;
    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) + ...
        59*R(b-dx) + 17*R(b);
    bigsum = 0;

    for i = a+4*dx:dx:b-4*dx
        bigsum = bigsum + R(i);
    end
    simpson = pi * (dx/48) * (smallsum + 48*bigsum);
    error = rounddown(simpson) - rounddown(prev);
    if error ≠ 0 % this is here so we can compare values manually(sanity check)
        prev = simpson;
    end
end


disp(n)
disp(prev)
disp(simpson)
```