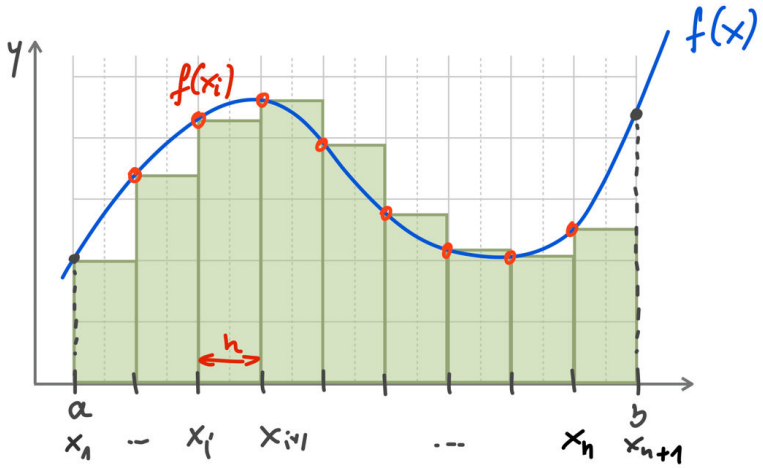
Numerical exploration of integration techniques

There exist multiple techniques for integration of data, some of which you may have heard about (e.g. rectangle rule, trapezoidal rule, Simpson method). In this project, you will compare the three methods by running an experiment called a convergence study.

Rectangle rule

Sometimes also called endpoint rule, assumes that we divide the integration interval into n sub-divisions and compute area of each rectangle given by the length of the sub-division $h = \frac{b-a}{n}$ and the value of the function (or data) at each point x_i.

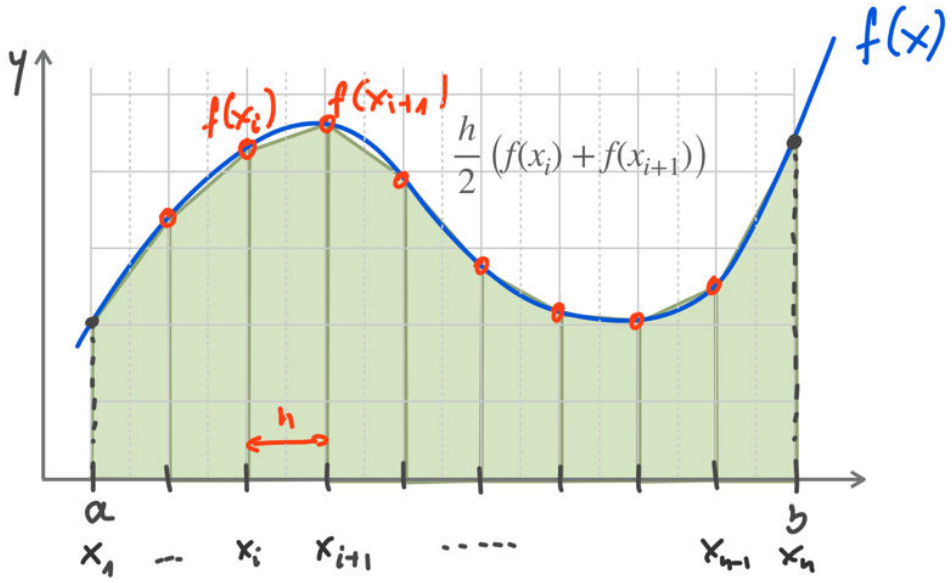


The sum of the area of the rectangles is then the approximation to the integral of the function in the interval (a,b):

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_i) \cdot h$$

Trapezoidal rule

The endpoint rule is not the only numerical procedure to integrate a function. Another idea is the trapezoidal rule. The algorithm is essentially the same as for the endpoint rule, but this time we sum up the areas of trapezoids spanned by basis formed by the function values at two consecutive points x_i, x_{i+1} in the interval $[a, b]$, and the length of the interval h. The figure below depicts this idea:



And the approximation becomes:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \frac{1}{2} h(f(x_{i+1}) + f(x_i))$$

Simpson rule

The final integration method considered here will be Simpson rule, which can be expressed as follows:

$$\int_a^b f(x)dx \approx \frac{1}{3} h \sum_{i=1}^{n/2} (f(x_{2i-1}) + 4f(x_{2i}) + f(x_{2i+1}))$$

Task 1

Consider a function $f(x) = \sin(x)$ on an interval $x \in [0, 2\pi]$. Define a set of $n + 1 = 21$ equi-spaced points on that interval such that $x_1 = 0$ and $x_{21} = 2\pi$ (use function `linspace()`). For each of those points compute the value of the function $f_i = f(x_i) = \sin(x_i)$.

We know that

$$\int_0^{2\pi} \sin(x)dx = 0$$

Use all three methods to compute the same integral and confirm that you get correct answer for each of the methods. You may want to create a function for each method. Each function would accept the values f_i and coordinates x_i and compute the value of the definite integral of this data.

Rectangle Method

```
A = rectangular(f,x) %call function
```

```
A =
   0.645874891023264
```

Trapazoid

```
A1 = Trapazoid(f,x) %call function
```

```
A1 =
   0.489506025912460
```

Simpson

```
A2 = Simpson(f,x) %call function
```

```
A2 =
   0.465644195692440
```

Task 2

You have probably noticed that you do not get exact integral in Task 1 for all the methods. It is typical in numerical methods that they provide approximate answers. The answers, however, get better as we improve the aproximation. In the case of integration, we can use smaller intervals, which will lead to better approximated area under the curve.

Let's integrate a function

$$f(x) = \frac{2}{1+x^2}$$

on the interval $x \in [0, 1]$.

Define a range of number of intervals $n = [10, 100, 1000, 10000, 100000]$. For each value of n, first create a set of $n+1$ equi-spaced points $x = [x_1, x_2, \ldots, x_{n+1}]$ and evaluate the function on those points such that $f_i = f(x_i) = \frac{2}{1+x_i^2}$. We know that

$$\int_0^1 \frac{2}{1+x^2} dx = \frac{\pi}{2}$$

which allows us to define error of the approximation as

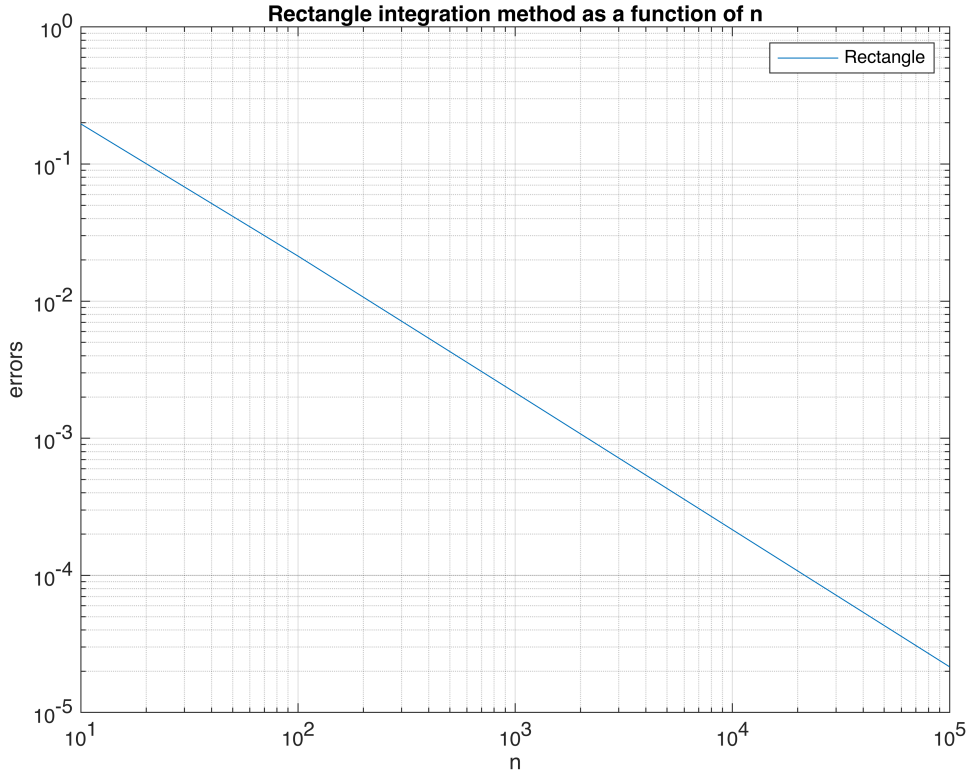
$$error = |I_{approx} - \pi/2|$$

where I_{approx} is the integral approximated by a numerical method, and $|.|$ indicates absolute value.

Create a plot of the error of each integration method as a function of n. Plot results of all three integration methods in one plot. You will want to use `loglog()` instead of `plot()` to make sure the axis have a logarithmic scale.

```
format long
n=10;% number of intervals
x = linspace(0,1,n);% Generate n number of spaces between 0 and 1
f = @(x) 2./(1+x.^2);% equation
A= Task2(f,x,n); % function
error= abs(A-pi/2);% approximate - absolute value

n=100;
A= Task2(f,x,n);
error1= abs(A-pi/2);

n=1000;
A= Task2(f,x,n);
error2= abs(A-pi/2);

n=10000;
A= Task2(f,x,n);
error3= abs(A-pi/2);

n=100000;
A= Task2(f,x,n);
error4= abs(A-pi/2);


n1 = [10, 100, 1000, 10000, 100000];%range of number of intervals of n
errors = [error, error1, error2, error3, error4];
```

```
loglog(n1, errors)% plot
legend('Rectangle')
xlabel('n')
ylabel('errors')
title('Rectangle integration method as a function of n' )
grid on
```




```
m = 10;
x = linspace(0,1,m);
f = @(x) 2./(1+x.^2);
A1 = Task2Trapazoid(f,x,m);
ert = abs(A1-pi/2)
```

```
ert =
   0.147178236598183
```

```
m1 = 100;
x = linspace(0,1,m1);
A2 = Task2Trapazoid(f,x,m1);
ert1 = abs(A2-pi/2)
```

```
ert1 =
   0.016335464383009
```

```
m1 = 1000;
x = linspace(0,1,m1);
A3 = Task2Trapazoid(f,x,m1);
ert2 = abs(A3-pi/2)
```

```
ert2 =
   0.001651350093272
```

```
m1 = 10000;
x = linspace(0,1,m1);
```

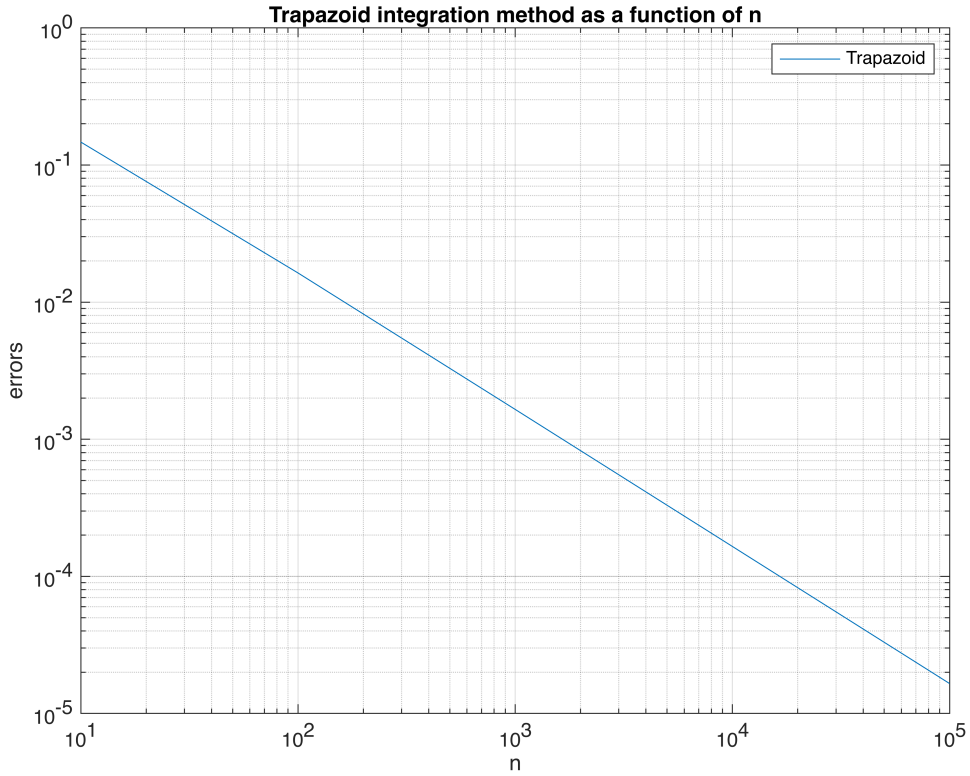
```
A4 = Task2Trapazoid(f,x,m1);
ert3 = abs(A4-pi/2)
```

```
ert3 =
     1.653148114935110e-04
```

```
m1 = 100000;
x = linspace(0,1,m1);
A5 = Task2Trapazoid(f,x,m1);
ert4 = abs(A5-pi/2)
```

```
ert4 =
     1.653328095141404e-05
```

```
m1 = [10, 100, 1000, 10000, 100000];
erts = [ert, ert1, ert2, ert3, ert4];

loglog(m1,erts)% plot
legend('Trapazoid')
xlabel('n')
ylabel('errors')
title('Trapazoid integration method as a function of n' )
grid on
```




```
l = 10;
x = linspace(0,1,l);
A2 = Task2Simpson(f,x,l);
```

```
errts = abs(A2-pi/2)
```

```
errts =
   0.139603645988444
```

```
l = 100;
x = linspace(0,1,l);
A2 = Task2Simpson(f,x,l);
errts1 = abs(A2-pi/2)
```

```
errts1 =
   0.015575564467299
```

```
l = 1000;
x = linspace(0,1,l);
A2 = Task2Simpson(f,x,l);
errts2 = abs(A2-pi/2)
```

```
errts2 =
   0.001575359778598
```

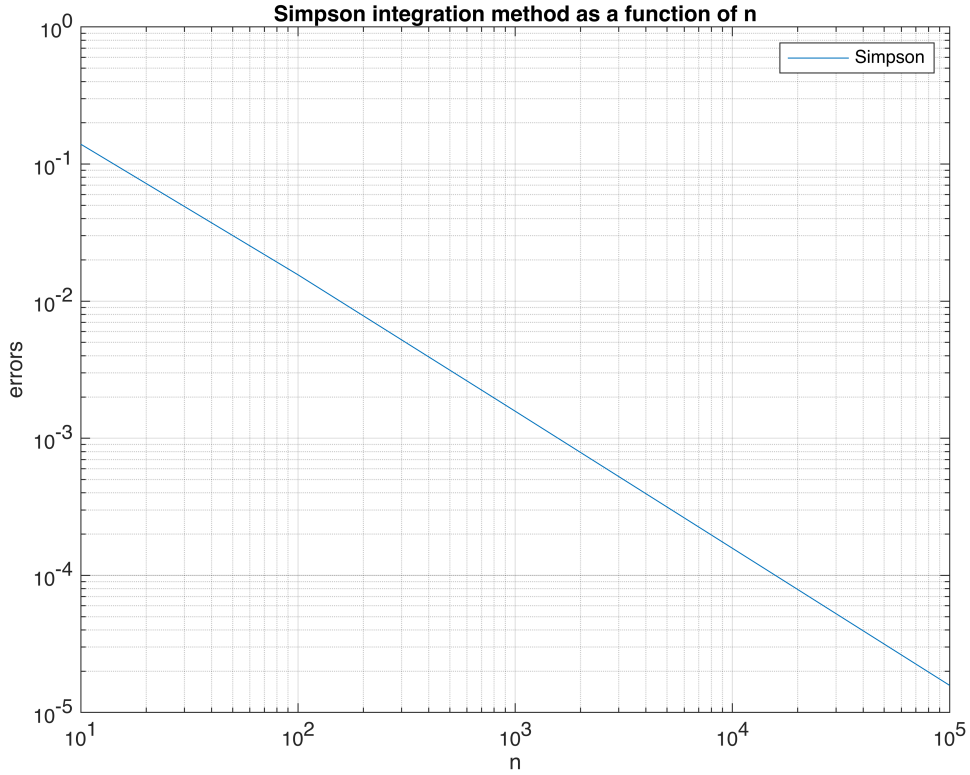
```
l = 10000;
x = linspace(0,1,l);
A2 = Task2Simpson(f,x,l);
errts3 = abs(A2-pi/2)
```

```
errts3 =
     1.577157799930173e-04
```

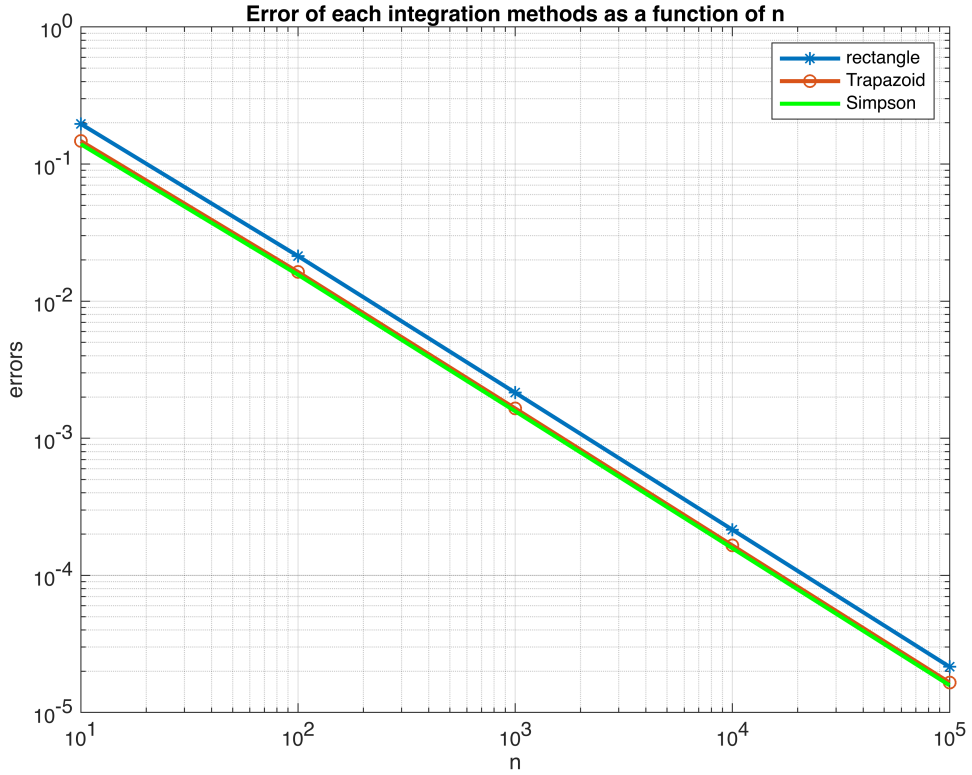
```
l = 100000;
x = linspace(0,1,l);
A2 = Task2Simpson(f,x,l);
errts4 = abs(A2-pi/2)
```

```
errts4 =
     1.577337780123145e-05
```

```
l1 = [10, 100, 1000, 10000, 100000];
errtss = [errts, errts1, errts2, errts3, errts4];

loglog(l1,errtss)% plot
legend('Simpson')
xlabel('n')
ylabel('errors')
title('Simpson integration method as a function of n' )
grid on
```




```
loglog(n1, errors,'*-',LineWidth=2), hold on
loglog(m1,erts,'o-',LineWidth=2), hold on
loglog(l1,errtss,'g-',LineWidth=2), hold off

legend('rectangle', 'Trapazoid', 'Simpson')
xlabel('n')
ylabel('errors')
title('Error of each integration methods as a function of n' )
grid on
```



What can you conclude from this plot regarding those three numerical methods? Which one would you consider the most accurate?

The more discretized the area under the curve becomes, the better the aproximation. This implies much smaller (h) values, in this case the Simpson method has the lowest error.

Feel free to increase n further. Do you see something strange happening?

The greater we increse the n value in the denominator the closer the value gets to the exact value.