Lecture 5
Integration

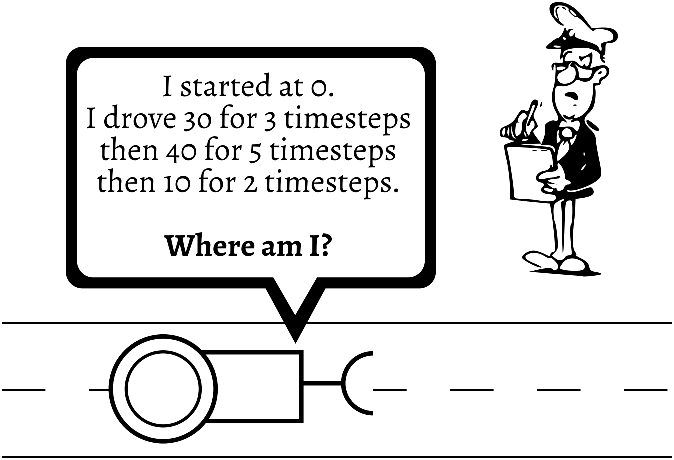
Jan Tekülve

jan.tekuelve@ini.rub.de

Computer Science and Mathematics
Preparatory Course

19.10.2020

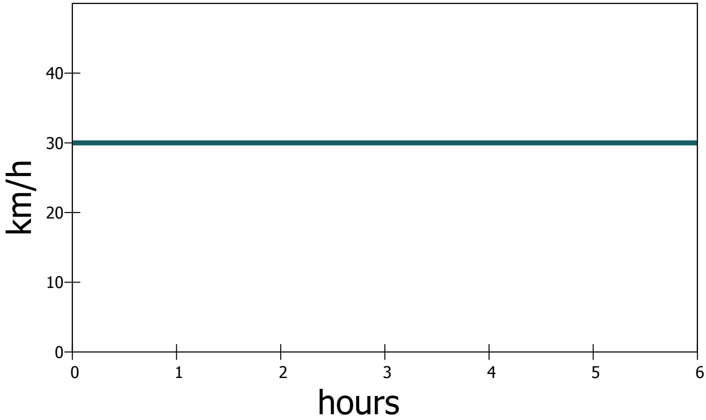
Reverting Differentiation



Overview

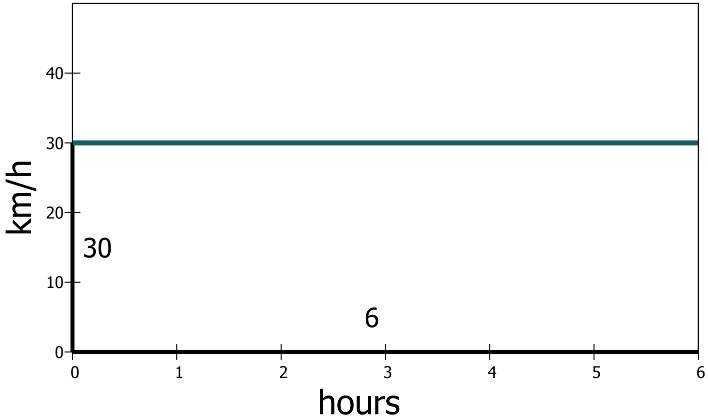
From Velocity to Position

You drove 30 km/h for 6 hours. How far did you drive?



From Velocity to Position

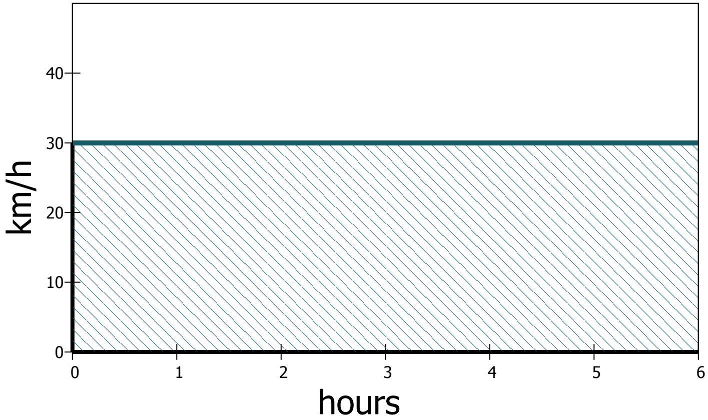
You drove 30 km/h for 6 hours. How far did you drive?



$$30\frac{\text{km}}{\text{h}} * 6\text{h} = 180\text{km}$$

From Velocity to Position

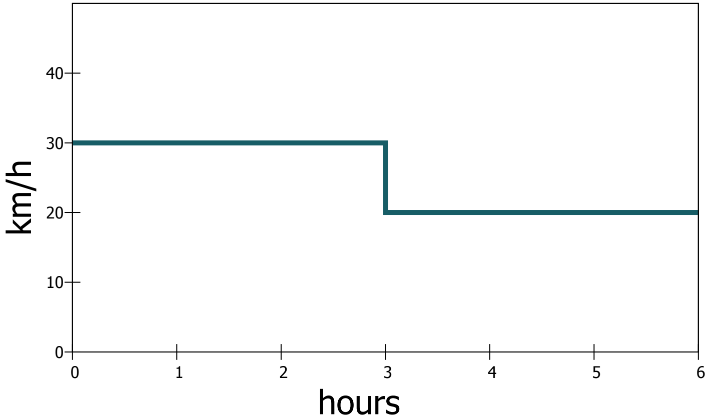
You drove 30 km/h for 6 hours. How far did you drive?



$$30\frac{\text{km}}{\text{h}} * 6\text{h} = 180\text{km}$$

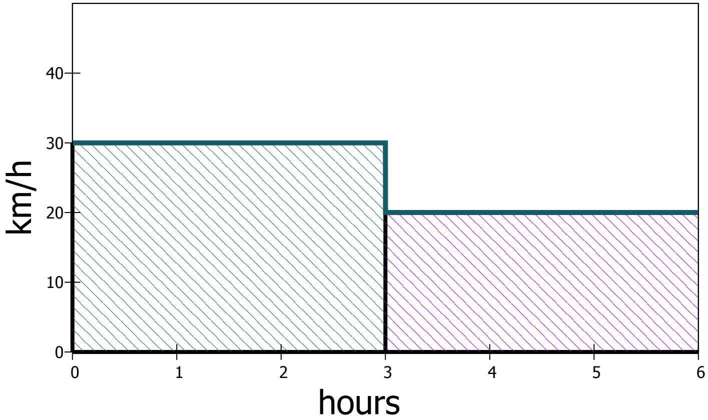
From Velocity to Position

Let's say you slowed down for the last 3 hours. How far did you get?



From Velocity to Position

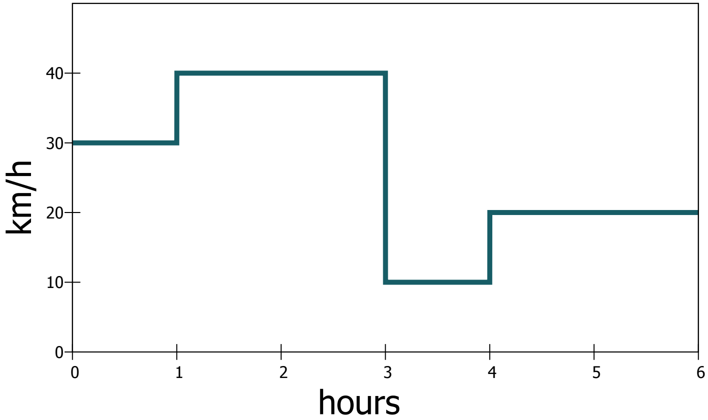
Let's say you slowed down for the last 3 hours. How far did you get?



$$30\frac{\text{km}}{\text{h}} * 3\text{h} + 20\frac{\text{km}}{\text{h}} * 3\text{h} = 140\text{km}$$

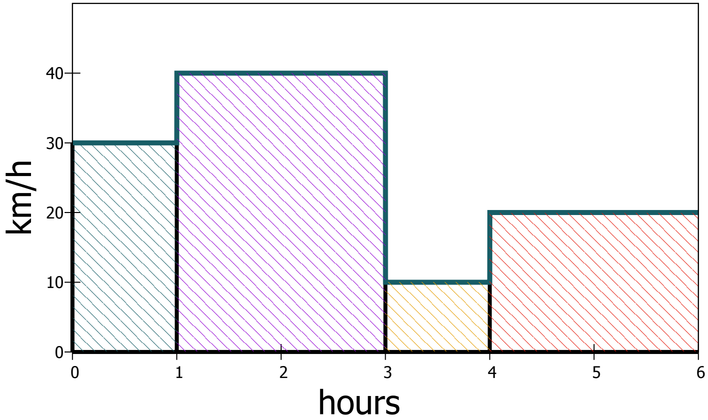
From Velocity to Position

What if you mixed it up to not get bored?



From Velocity to Position

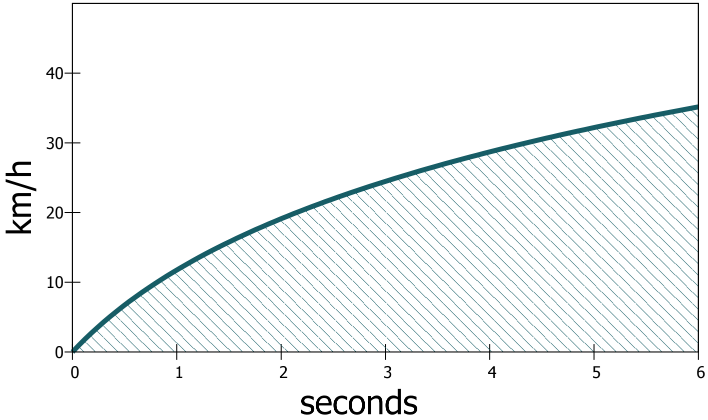
What if you mixed it up to not get bored?



$$30\frac{\text{km}}{\text{h}} * 1\text{h} + 40\frac{\text{km}}{\text{h}} * 2\text{h} + 10\frac{\text{km}}{\text{h}} * 1\text{h} + 20\frac{\text{km}}{\text{h}} * 2\text{h} = 160\text{km}$$

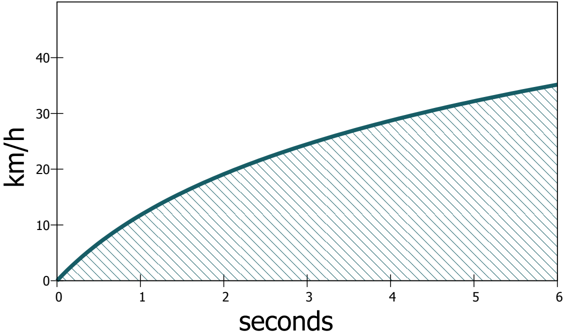
From Velocity to Position

But how about something realistic?



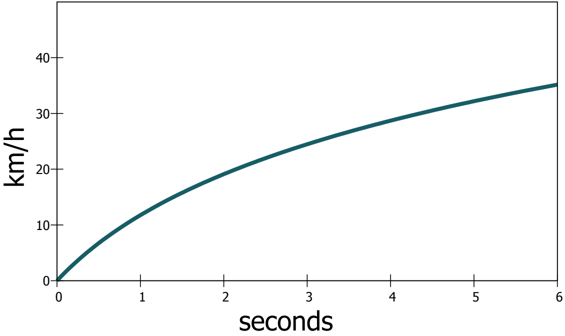
Approximation

- Not all areas can be calculated with rectangles



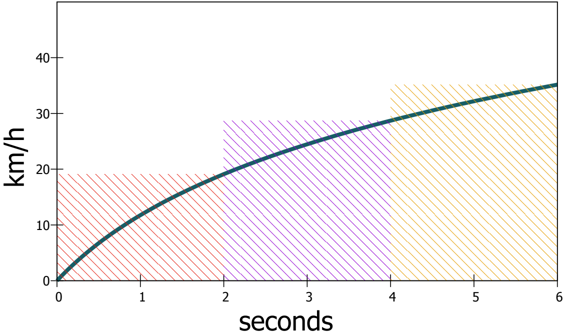
Approximation

- Not all areas can be calculated with rectangles
- One can however **approximate** them



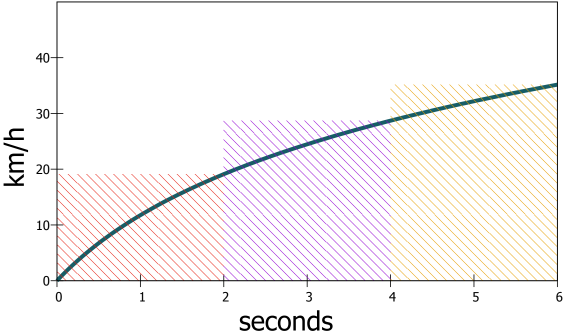
Approximation

- Not all areas can be calculated with rectangles
- One can however **approximate** them



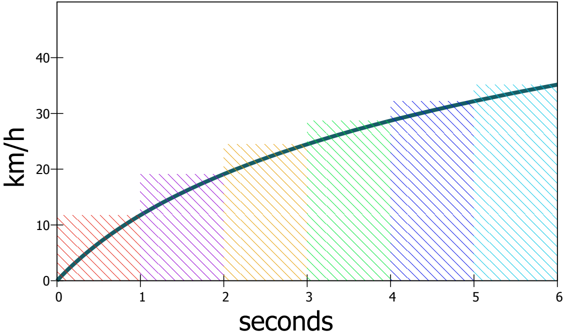
Approximation

- Not all areas can be calculated with rectangles
- One can however **approximate** them
- The more rectangles the better the approximation becomes

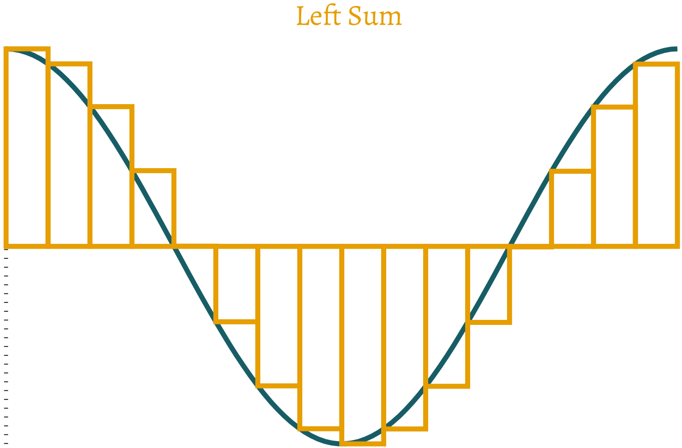


Approximation

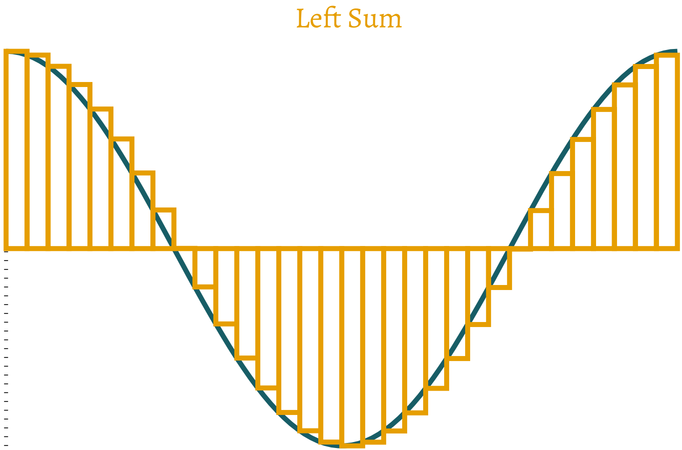
- Not all areas can be calculated with rectangles
- One can however **approximate** them
- The more rectangles the better the approximation becomes



Riemann Sums



Riemann Sums



Riemann Sums

Left and Right Sum

- For an interval $[x_i, x_{i+1}]$ and a function f the functions

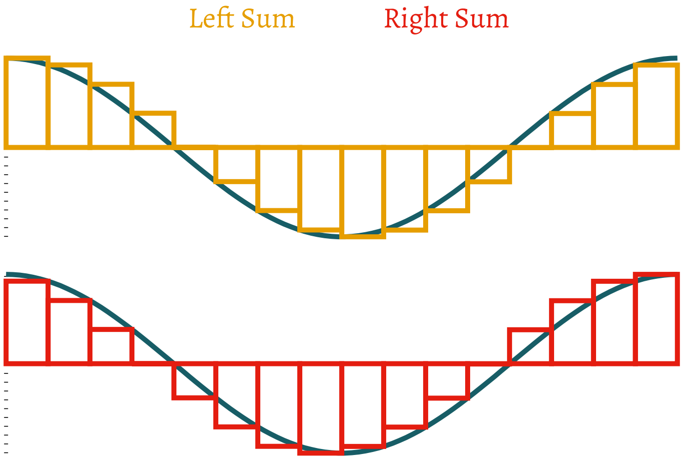
$$\text{Left}(f, [x_i, x_{i+1}[) = f(x_i) \text{ and } \text{Right}(f, [x_i, x_{i+1}]) = f(x_{i+1})$$

  are defined to return the leftmost or rightmost value of the function in the interval.

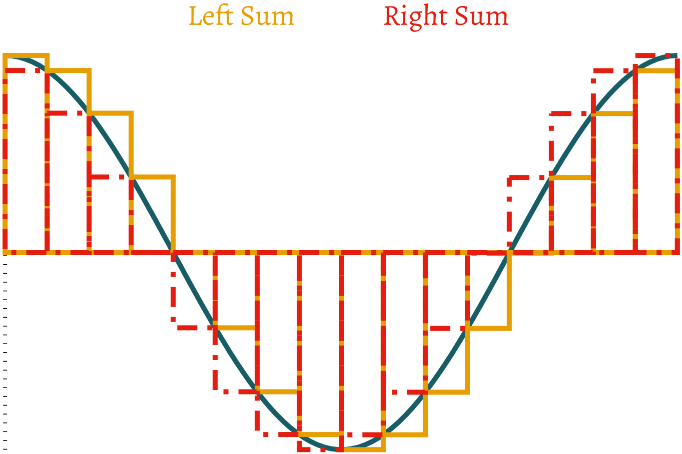
- **Left and Right Sum** are defined as the Sums of Left and Right across whole partitioned interval $(x_i)_{i \in [a,b]}$ with partition length Δx

$$I_L = \sum_i^n \text{Left}(f, x_i, x_{i+1})\Delta x \text{ and } I_R = \sum_i^n \text{Right}(f, x_i, x_{i+1})\Delta x$$

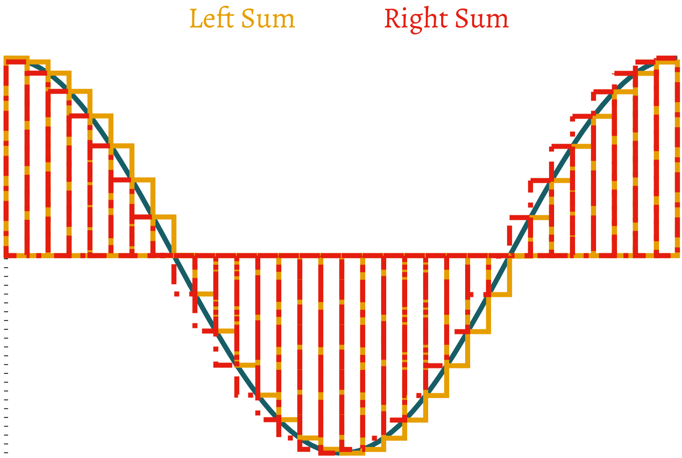
Left and Right Sum



Left and Right Sum



Left and Right Sum



Estimation of the True Area

- Left and Right Sums for a partition $(x_i)_{i\in[a,b]}$ give us an estimate of the area A

$$I_L \leq A \leq I_R,$$

if the function in the interval $[a, b]$ is increasing and

$$I_R \leq A \leq I_L,$$

if the function in the interval $[a, b]$ is decreasing.

Midpoint Method

Calculating Midpoints

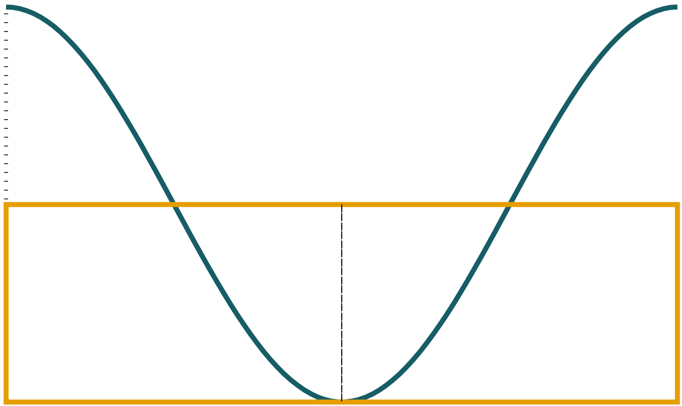
Another way of approximating an integral with finite sums is the **Midpoint Method**, which uses the function value in the middle of a given interval $[x_i, x_{i+1}]$

$$\mathrm{Mid}(f, [x_i, x_{i+1}]) = f(\frac{x_i + x_{i+1}}{2})$$

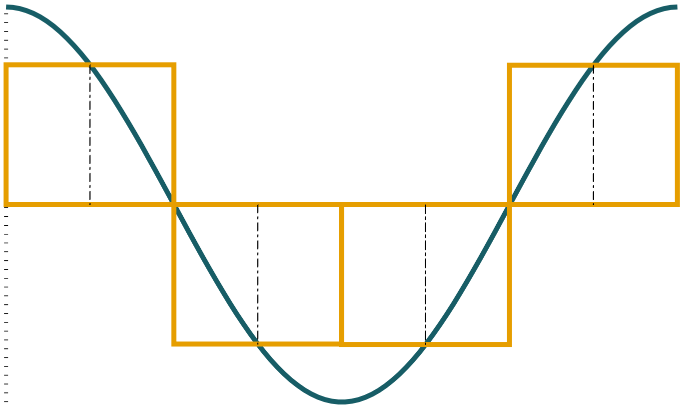
The sum of Midpoints also yields an estimation of the area under the curve

$$I_M = \sum_i^n Mid(f, [x_i, x_{i+1}])\Delta x$$

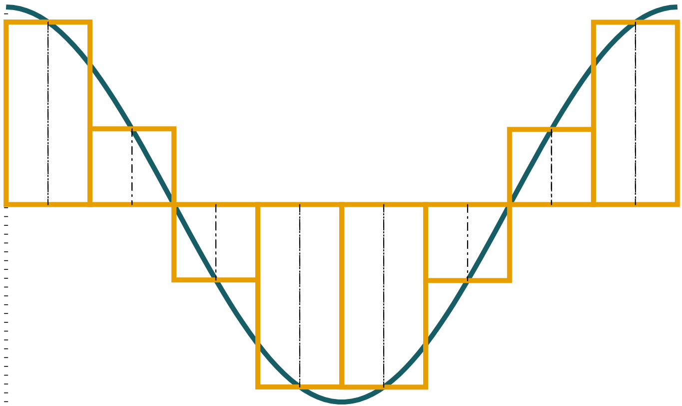
Midpoint Sums



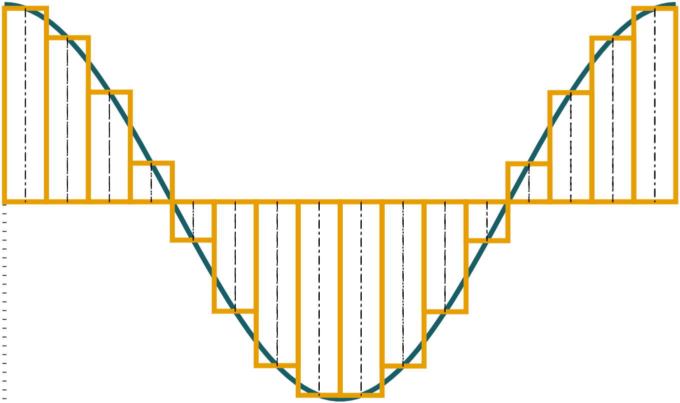
Midpoint Sums



Midpoint Sums



Midpoint Sums



From Sums to Integrals

Midpoint Sum: $\sum_i^n Mid(f, [x_i, x_{i+1}])\Delta x$

The more elements n, the smaller Δx and the better our approximation.

From Sums to Integrals

Midpoint Sum: $\sum_i^n Mid(f, [x_i, x_{i+1}])\Delta x$

The more elements n, the smaller Δx and the better our approximation.

What if n becomes ∞ and Δx becomes infinitely small?

From Sums to Integrals

Midpoint Sum: $\sum_i^n Mid(f, [x_i, x_{i+1}])\Delta x$

The more elements n, the smaller Δx and the better our approximation.

What if n becomes ∞ and Δx becomes infinitely small?

Definite Integral

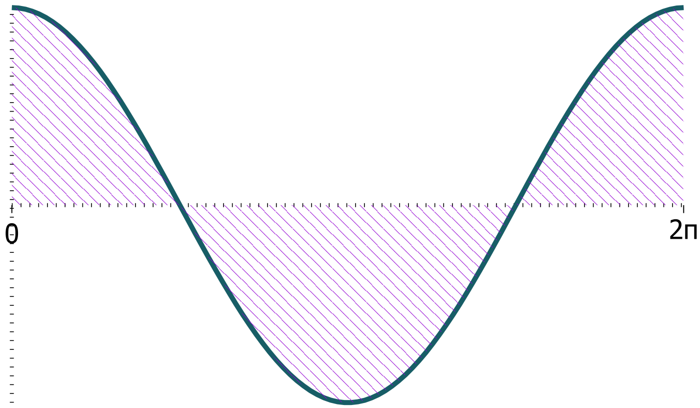
The **definite integral** of a function $f(x)$ between the **lower boundary** a and the **upper boundary** b

$$\int_a^b f(x)dx$$

is defined as the size of the area between f and the x-axis inside the boundaries. Areas above the x-Axis are considered positive and areas below negative.

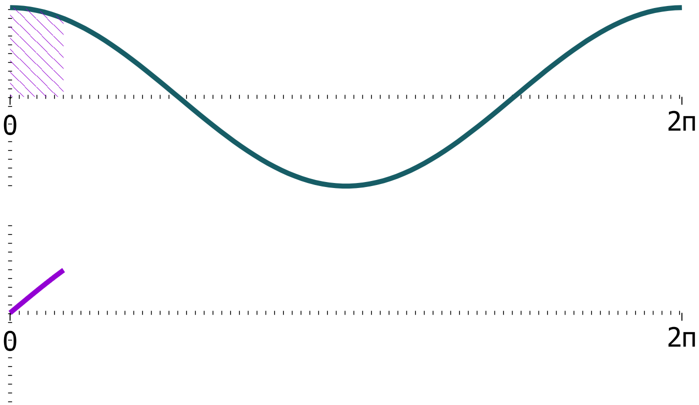
Integral as Area

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x)$



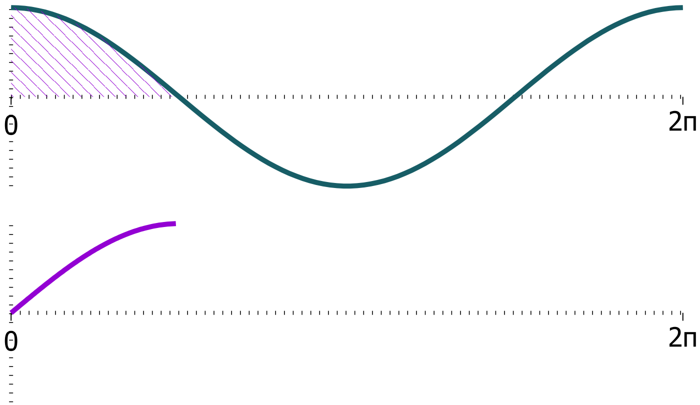
Integral as Function

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x)$



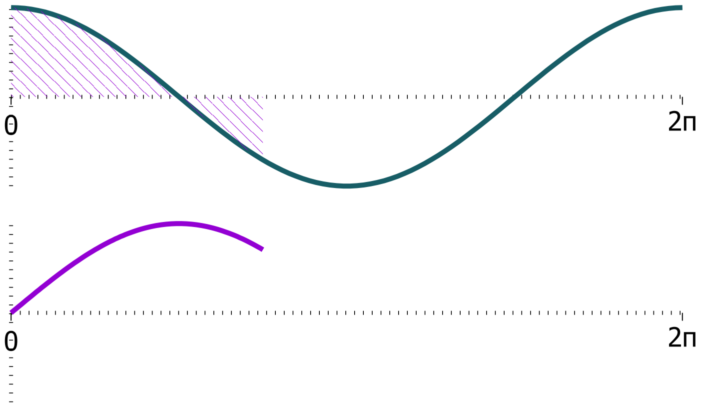
Integral as Function

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x)$



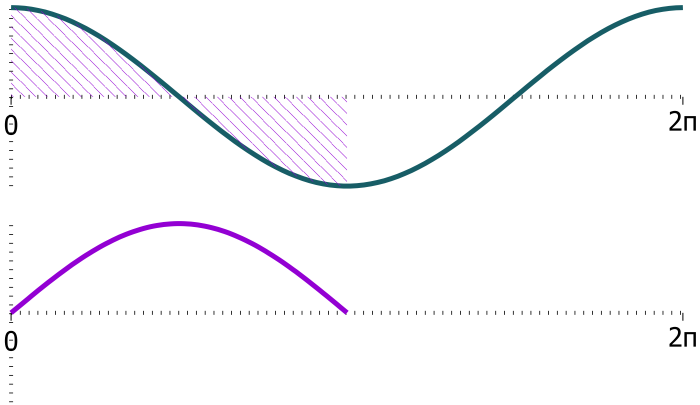
Integral as Function

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x)$



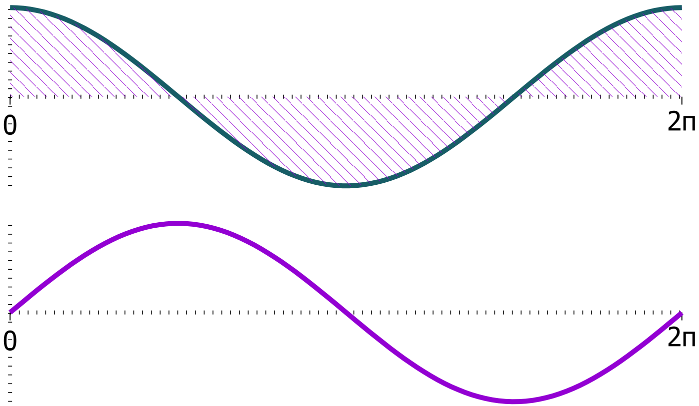
Integral as Function

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x)$



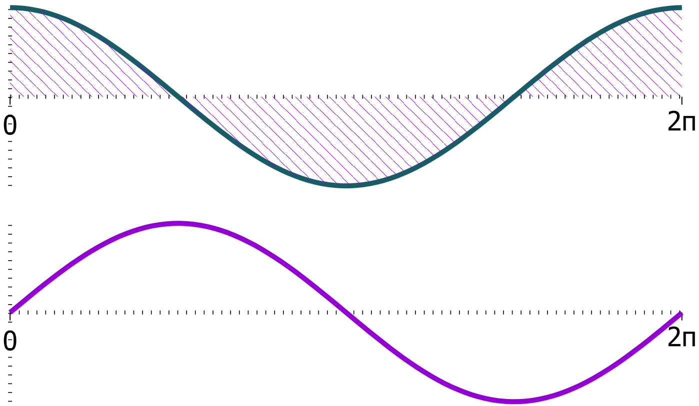
Integral as Function

$f(x) = cos(x) \qquad \int_0^{2\pi} cos(x)$



Integral as Function

$f(x) = cos(x)$ $\int_0^{2\pi} cos(x) = sin(2\pi)$



The Antiderivative

Definition

If f is a function with domain $[a, b] \to \mathbb{R}$ and there is a function F, which is differentiable in the interval $[a, b]$ with the property that

$$F'(x) = f(x),$$

then F is considered the **antiderivative** of f

The Antiderivative

Definition

If f is a function with domain $[a, b] \to \mathbb{R}$ and there is a function F, which is differentiable in the interval $[a, b]$ with the property that

$$F'(x) = f(x),$$

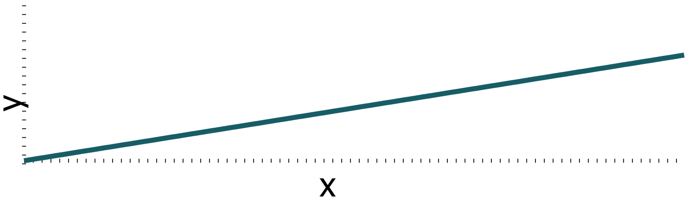
then F is considered the **antiderivative** of f

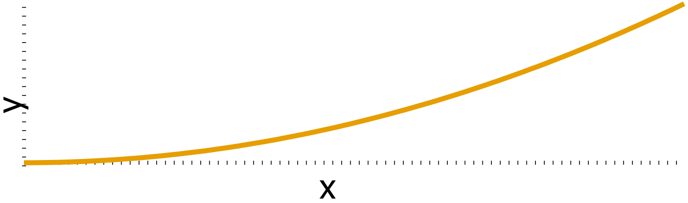
Properties of the antiderivative

- Differentiation removes constants, because of that an antiderivative is described by a family of functions $F(x) + c$
- Unlike with differentiation there are no fixed rules to compute an antiderivative from a given f

A function and its antiderivative

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





Calculating with Integrals

Fundamental Theorem of Calculus

If f is integrable and continuous in $[a, b]$. Then the following holds for each antiderivative F of f

$$\int_a^b f(x)dx = \int_a^b F'(x)dx = [F(x)]_a^b = F(b) - F(a)$$

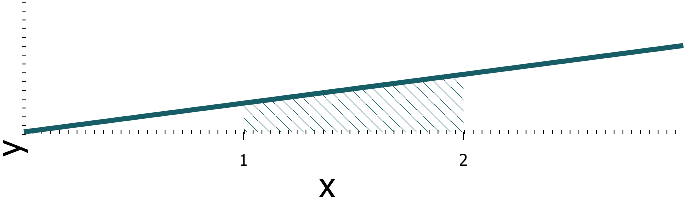
Example:

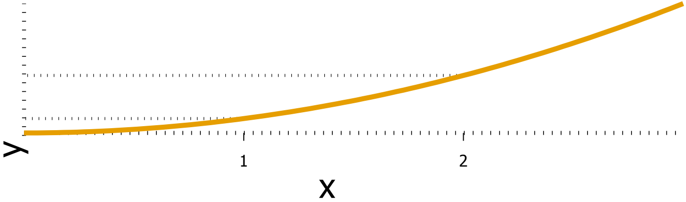
- Area under $f(x)$ between values 1 and 2

$$\int_1^2 xdx = \left[\frac{1}{2}x^2\right]_1^2 = \frac{1}{2}2^2 - \frac{1}{2}1^2 = 1.5$$

Integral as area underneath a function

$f(x) = x \qquad F(x) = \frac{1}{2}x^2 \qquad \int_1^2 f(x)dx = F(2) - F(1)$





The Integral as Linear Operator

Integration Rules

- **Summation**

$$\int_a^b f(x) + g(x) = \int_a^b f(x) + \int_a^b g(x)$$

The Integral as Linear Operator

Integration Rules

- **Summation**

$$\int_a^b f(x) + g(x) = \int_a^b f(x) + \int_a^b g(x)$$

- **Scalar Multiplication**

$$\int_a^b cf(x) = c\int_a^b f(x)$$

The Integral as Linear Operator

Integration Rules

- **Summation**

$$\int_a^b f(x) + g(x) = \int_a^b f(x) + \int_a^b g(x)$$

- **Scalar Multiplication**

$$\int_a^b cf(x) = c\int_a^b f(x)$$

- **Boundary Transformations**

$$\int_a^b f(x) + \int_b^c f(x) = \int_a^c f(x) \quad \wedge \quad \int_a^b f(x) = -\int_b^a f(x)$$

Improper Integrals

Infinite Intervals

It is possible to calculate the area in infinitely large intervals. Intervals with an infinite boundary are called **Improper Integrals**

$$\int_a^\infty f(x)dx = \lim_{b\to\infty} \int_a^b f(x)dx$$

Example:

- Convergent improper integral

$$\int_1^\infty x^{-2}dx = \lim_{b\to\infty} \int_1^b x^{-2}dx = \lim_{b\to\infty} \left[-x^{-1}\right]_1^b = \lim_{b\to\infty} \left(-b^{-1} + 1\right) = 1$$

Numerical Approximation

- It is not trivial to find the antiderivative to a given function or a given dataset

Numerical Approximation

- It is not trivial to find the antiderivative to a given function or a given dataset
- Instead of calculating the Integral the area beneath a curve may be approximated, by splitting the area into sub-areas

Numerical Approximation

- It is not trivial to find the antiderivative to a given function or a given dataset
- Instead of calculating the Integral the area beneath a curve may be approximated, by splitting the area into sub-areas
- That seems familiar!

Numerical Approximation

- It is not trivial to find the antiderivative to a given function or a given dataset
- Instead of calculating the Integral the area beneath a curve may be approximated, by splitting the area into sub-areas
- That seems familiar!

Partioning an Interval

Let $(x_i)_{i\in[a,b]}$ be a sequence of n increasing numbers in $[a, b]$ with fixed distance h between x_i and $x_i + 1$ for all x_i.

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

(Simple) Numerical Integration

- We can understand Numerical Integration as the opposite of Numerical Differentiation

(Simple) Numerical Integration

- We can understand Numerical Integration as the opposite of Numerical Differentiation
- Thus instead of subtracting following elements, we will add to the previous element

(Simple) Numerical Integration

- We can understand Numerical Integration as the opposite of Numerical Differentiation
- Thus instead of subtracting following elements, we will add to the previous element
- This is equivalent to the left Riemann Sums

(Simple) Numerical Integration

- We can understand Numerical Integration as the opposite of Numerical Differentiation
- Thus instead of subtracting following elements, we will add to the previous element
- This is equivalent to the left Riemann Sums

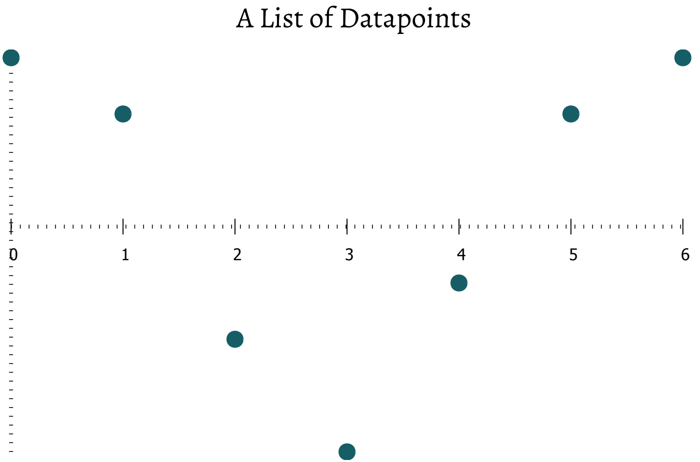
(Simple) Numerical Differentiation

The set $\mathbb{I}$ describes the computable domain of f in the given context. It is possible to calculate function value $f(x_i)$, where $x_i \in \mathbb{I}$.

$$F(x_i) \approx F(x_{i-1}) + (f(x_i) \cdot (x_i - x_{i-1})),$$

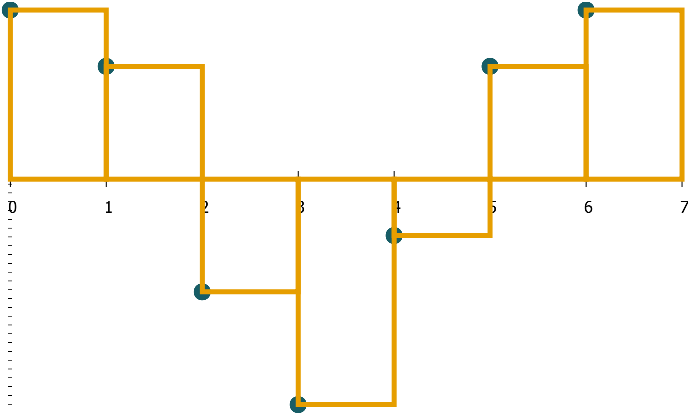
where x_{i-1} is the smallest negative distance from x_i in $\mathbb{I}$.

Graphical Example



Graphical Example

A List of Datapoints



Integrating a List of Datapoints

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	1	2	3	4	5	6
$f(x_i)$	3	2	-2	-4	-1	2	3

Integrating a List of Datapoints

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	1	2	3	4	5	6
$f(x_i)$	3	2	-2	-4	-1	2	3

- The distance between each point is $\Delta x = 1$.
  The area underneath each point is therefore $1 * f(x_i)$

Integrating a List of Datapoints

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	1	2	3	4	5	6
$f(x_i)$	3	2	-2	-4	-1	2	3

- The distance between each point is $\Delta x = 1$.
  The area underneath each point is therefore $1 * f(x_i)$

- The integrated position for $x_3 = 3$ and startpoint $s = 2$ equals:

$$F(x_3) = s + 1f(x_0) + 1f(x_1) + 1f(x_2) + 1f(x_3) = 2 + 3 + 2 + (-2) + (-4) = 1$$

- The position at time-step $x_3 = 3$ is 1

Changing the Precision

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	0.5	1	1.5	2	2.5	3
$f(x_i)$	3	2.5	2	0	-2	-2.5	-4

Changing the Precision

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	0.5	1	1.5	2	2.5	3
$f(x_i)$	3	2.5	2	0	-2	-2.5	-4

- The distance between each point is $\Delta x = 0.5$.
  The area underneath each point is therefore $0.5 * f(x_i)$

Changing the Precision

- From a sensor we receive the following velocity values $f(x_i)$:

i	0	1	2	3	4	5	6
x_i	0	0.5	1	1.5	2	2.5	3
$f(x_i)$	3	2.5	2	0	-2	-2.5	-4

- The distance between each point is $\Delta x = 0.5$.
  The area underneath each point is therefore $0.5 * f(x_i)$

- The integrated position for $x_6 = 3$ and startpoint $s = 2$ equals:

$$\begin{aligned} F(x_6) &= s + 0.5f(x_0) + 0.5f(x_1) + 0.5f(x_2) \\ &\quad + 0.5f(x_3) + 0.5f(x_4) + 0.5f(x_5) + 0.5f(x_6) \\ &= 2 + 1.5 + 1.25 + 1 + 0 + (-1) + (-1.25) + (-2) = 1.5 \end{aligned}$$

- The position at time-step $x_6 = 3$ is 1.5

Reading Files

- Opening a file

```
fileObject = open("file.txt", "r")
#The option r stands for read
```

- Reading the file contents

```
#readlines creates a list containing each line
lines = fileObject.readlines()
for line in lines:
    print(line)
```

- Close the file after usage:

```
fileObj.close()#This can be done right after readlines()
```

Details on Strings

- Useful string operations

```
#Strip removes the new-line character '\n'
line = line.strip()
#Split tokenizes the string at the given character
line = line.split(" ")# 'Hello you' to ['Hello','you']
line = line.split("o")# 'Hello you' to ['Hell',' y','u']
line = line.replace("l","b")# 'Hello you' to 'Hebbo you'
```

Task Template

- Download the archive *task_template_5.zip* from the course homepage. Extract it into a folder of your choice.
- The archive contains *task_51.py*, *student_code_51.py*, *task_52.py*, *student_code_52.py*, and *velocity_series.txt*.
- You only need to edit code in the *student_code* files.

Explain Task Template!

Tasks

1. Implement the function *area(a,b)*, which returns the area between x-axis and function curve $f(x)$ in the interval a, b.
   - The functions $f(x)$ and $F(x)$ are implemented already and you can call them in your code
   - Run *task51.py* to verify your result and plot an example area.
   - Given $F(x) = 4x^3 + 2x^2$ calculate $f(x)$ on a piece of paper.
   - Implement both functions $F(x)$ and $f(x)$ in *student_code* instead of the given ones. Test using *task51.py*.
2. Calculate the derivative of $F(x) = sin(x)$ and verify the result via numerical integration.
   - Implement the function *numerical_integration* using the formula for (simple) numerical integration.
   - A list of y-values, a start point and a step size are given through the function argument.
   - Use a for loop through all y-values of function $f(x)$ and use them to calculate the values $F(x)$. Append them to the list *y_values_F*.
   - Play around with the *precision* parameter of the *task52.py*. What happens?

File Reading Task

3.1* Write a script that opens *velocity_series.txt* from the course page, reads its contents and stores them as a list of floating values.

- Use *file.readlines*() to receive a list of strings containing each line
- Extract the velocity in each line by applying the *split*() method in a for-loop
- In the loop typecast the velocity into a float and append it to a second list

3.2* Copy your function *numerical_integration* from the previous task and calculate the resulting series of positions.

- Choose a step size and a starting point of your choice
- If that works read the step size and the starting point from the file *velocity_series.txt*. You might need to change your code from 3.1 to achieve this.