# Math  Physics Lab Documentation

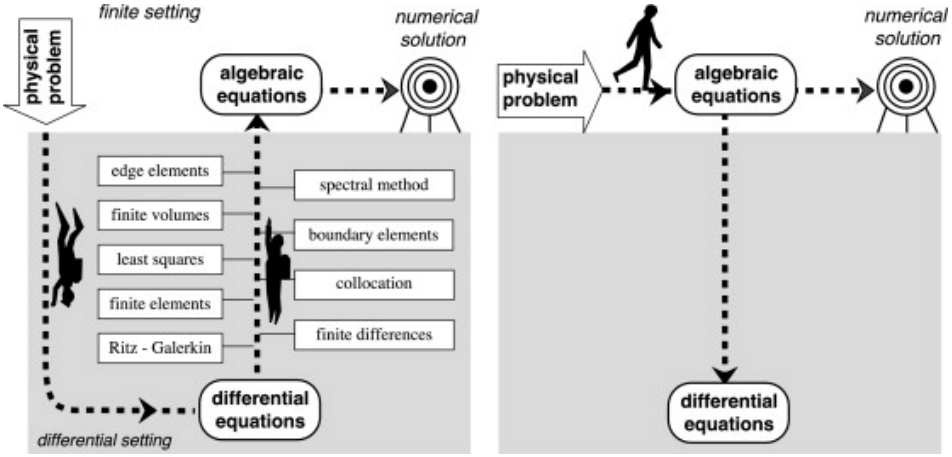
***Release 0.1.2***

**Enzo Tonti**

April 25, 2016

Contents

The professor Enzo Tonti's Math & Physics Lab project aims to provide a series of small software programs implementing basic math and physics equations.




prof. Enzo Tonti *[A1]* at the University of Trieste, Italy, pioneered in the early '80s the use of personal computers for scientific computing inspiring generation of future computational scientists.

We hope with this project to continue to inspire young student interests in math and physics and to become the next generation of computational scientists.

CHAPTER 1

# Features

- Collection of software programs implementing basic math and physics equations

CHAPTER 2

# Contribute

- Documentation: https://github.com/enzotonti/mathlab/tree/master/doc
- Issue Tracker: https://github.com/enzotonti/mathlab/issues
- Source Code: https://github.com/enzotonti/mathlab

CHAPTER 3

# Content

## 3.1 About

This section describes what Math & Physics Lab project is about.

## 3.2 Install

This section covers the basics of how to download and install the Math & Physics Lab project.

**Contents:**



### 3.2.1 Installing from source

Clone the Math & Physics Lab project from GitHub repository:

```
git clone https://github.com/enzotonti/mathlab.git mathlab
```

then:

```
cd mathlab
python setup.py install
```

## 3.3 API reference

**project Modules:**

### 3.3.1 `mathlab.geometry`

Collection of geometry tutorials

**Functions:**

| | |
|---|---|
| *point_line*(xa, ya, xb, yb, xp, yp) | Given a line r passing through two points A and B and assigned a point P, this function dete |
| *gravity_center*(points) | To do: complete documentation |
| *electric_dipole*(charge_location) | To do: complete documentation |

mathlab.geometry.**point_line**(*xa*, *ya*, *xb*, *yb*, *xp*, *yp*)
Given a line r passing through two points A and B and assigned a point P, this function determines the projection of P on the line, evaluates its distance from the line and it indicates whether the projection is internal to the segment AB.

**Parameters** **xa, ya, xb, yb, xp, yp** (*float*) – Coordinate (X, Y) of A, B and P.

**Returns** *distance* – Distance between P and the line

mathlab.geometry.**gravity_center**(*points*)
To do: complete documentation

**Parameters** **points** – Coordinate ....

mathlab.geometry.**electric_dipole**(*charge_location*)
To do: complete documentation

**Parameters** **charge_location** – Coordinate ....

### 3.3.2 mathlab.differential

Collection of differential equation tutorials

#### Functions:

| | |
|---|---|
| *function_05*(parameter_01, parameter_02, ...) | Function description. |
| *function_06*(parameter_01, parameter_02, ...) | Function description. |

mathlab.differential.**function_05**(*parameter_01*, *parameter_02*, *parameter_03*)
Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

mathlab.differential.**function_06**(*parameter_01*, *parameter_02*, *parameter_03*)
Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

### 3.3.3 mathlab.module_03

Module for describing .....

#### Functions:

| | |
|---|---|
| *function_01*(parameter_01, parameter_02, ...) | Function description. |
| *function_02*(parameter_01, parameter_02, ...) | Function description. |

mathlab.module_03.**function_01**(*parameter_01*, *parameter_02*, *parameter_03*)
Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

mathlab.module_03.**function_02**(*parameter_01*, *parameter_02*, *parameter_03*)
Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

### 3.3.4 mathlab.module_04

Module for describing .....

#### Functions:

| | |
|---|---|
| *function_03*(parameter_01, parameter_02, ...) | Function description. |
| *function_04*(parameter_01, parameter_02, ...) | Function description. |

mathlab.module_04.**function_03**(*parameter_01*, *parameter_02*, *parameter_03*)
Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

mathlab.module_04.**function_04**(*parameter_01*, *parameter_02*, *parameter_03*)

Function description.

**Parameters**

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

**Returns** *return_01* – Description.

## 3.4 Examples

We provide Jupyter Notebooks for all the Math & Physics Lab functions.

To run these examples on your computer install and run Jupyter Notebook

### 3.4.1 Distance between a point and a line

Given a line r passing through two points A and B and assigned a point P, this function determines the projection of P on the line, evaluates its distance from the line and it indicates whether the projection is internal to the segment AB.

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: import mathlab
```

importing the Math and Physics Lab project

equation describing a line between two points A-B:

$$x = x_{\mathrm{A}} + s * (x_{\mathrm{B}} - x_{\mathrm{A}})$$

$$y = y_{\mathrm{A}} + s * (y_{\mathrm{B}} - y_{\mathrm{A}})$$

with:

$$0 \leq s \leq 1$$

equation of a line passing through P and orthogonal to A-B:

$$x = x_{\mathrm{P}} + t * sin(\alpha)$$

$$y = y_{\mathrm{P}} + t * cos(\alpha)$$

with:

$$sin(\alpha) = (y_{\mathrm{B}} - y_{\mathrm{A}})/L$$

$$cos(\alpha) = (x_{\mathrm{B}} - x_{\mathrm{A}})/L$$

and

$$L = \sqrt{(x_{\mathrm{B}} - x_{\mathrm{A}})^2 + (y_{\mathrm{B}} - y_{\mathrm{A}})^2}$$

instersection between the two line:

$$x_A + s * (x_B - x_A) = x_P + t * sin(\alpha)$$

$$y_A + s * (y_B - y_A) = y_P - t * cos(\alpha)$$

system of equation:

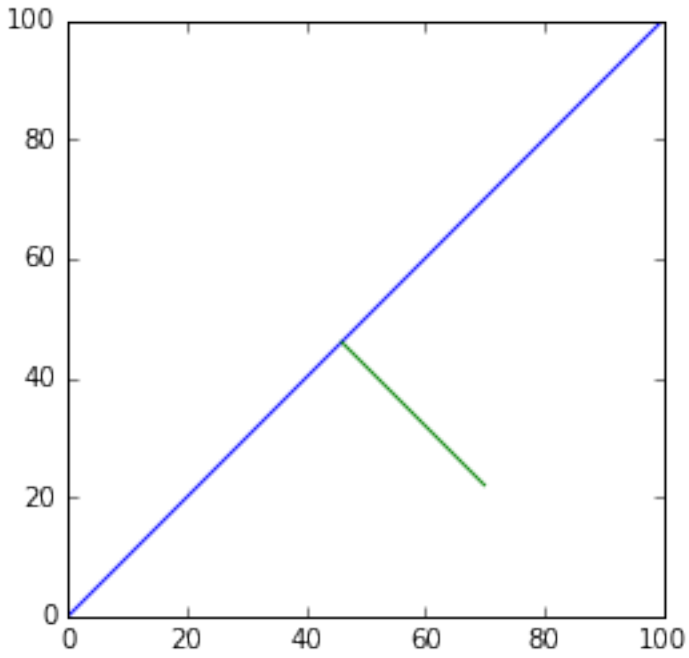
$$s * (x_B - x_A) - t * sin(\alpha) = x_P - x_A$$

$$s * (y_B - y_A) + t * cos(\alpha) = y_P - y_A$$

$$x_A = 0; y_A = 0; x_B = 100; y_B = 110; x_P = 70; y_P = 22$$

```
In [4]: result = mathlab.point_line(0, 0, 100, 100, 70, 22)
```

```
projection point:  46.0 46.0
distance:  -33.941125497
```




```
In [ ]:
```

### 3.4.2 Center of Gravity

Given n points defining a polygon it calculates the center of mass.

```
In [1]: %pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

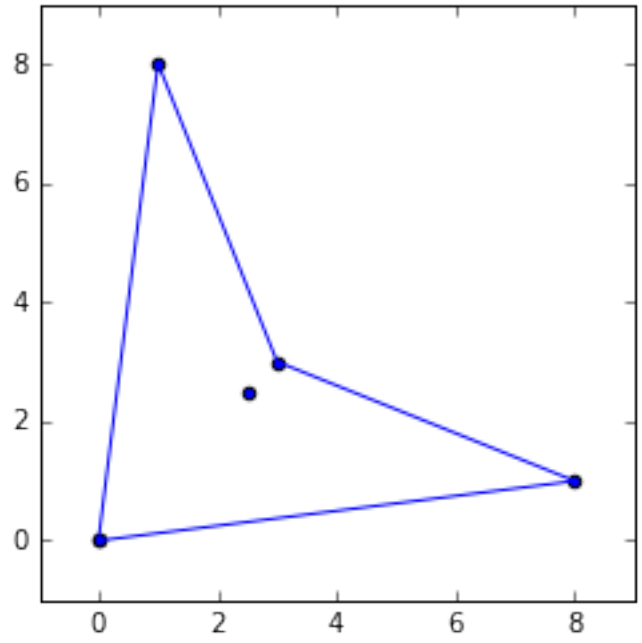
```
In [2]: import mathlab
```

importing the Math and Physics Lab project

To Do: write equations/comments/

```
In [3]: points = [[0,0], [8,1], [3,3], [1,8], [0,0]]
        mathlab.gravity_center(points)
```

```
x[] [0, 8, 3, 1, 0]
y[] [0, 1, 3, 8, 0]
area[]: [0.0, 10.5, 10.5, 0.0]
total area:  21.0
```




```
In [ ]:
```

### 3.4.3 Electric Dipole

To complete.

```
In [1]: %pylab inline
```

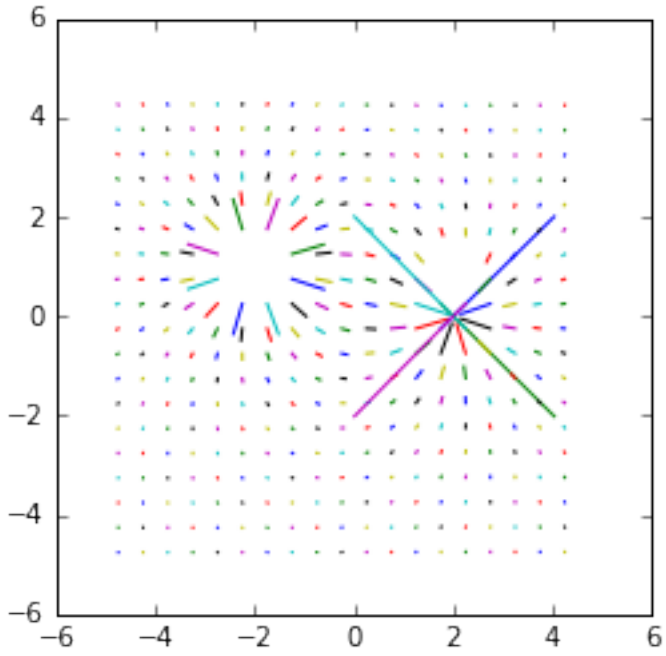
```
Populating the interactive namespace from numpy and matplotlib
```

```
In [2]: import mathlab
```

importing the Math and Physics Lab project

To Do: write equations/comments/

```
In [4]: charge_location = [[-2.0, 1.0], [2.0, 0]]
        mathlab.electric_dipole(charge_location)
```




```
In [ ]:
```

## 3.5 Credits

### 3.5.1 Citations

We kindly request that you cite the following article *[A1]* if you use the Math & Physics Lab project.

### 3.5.2 References

# Bibliography

[A1] Enzo Tonti. Why starting from differential equations for computational physics? *Journal of Computational Physics*, 257, Part B():1260 – 1290, 2014. Physics-compatible numerical methods. URL: http://www.sciencedirect.com/science/article/pii/S0021999113005548, doi:http://dx.doi.org/10.1016/j.jcp.2013.08.016.

[B1] Enzo Tonti. Why starting from differential equations for computational physics? *Journal of Computational Physics*, 257, Part B():1260 – 1290, 2014. Physics-compatible numerical methods. URL: http://www.sciencedirect.com/science/article/pii/S0021999113005548, doi:http://dx.doi.org/10.1016/j.jcp.2013.08.016.

## m

## E



## F



## G



## M



## P