Fourth Year Laboratory

E5 - Simulation of Control Systems Using C++

Amr Keleg

Faculty of Engineering, Ain Shams University

December 29, 2020

Contact: `amr_mohamed@live.com`

Outline

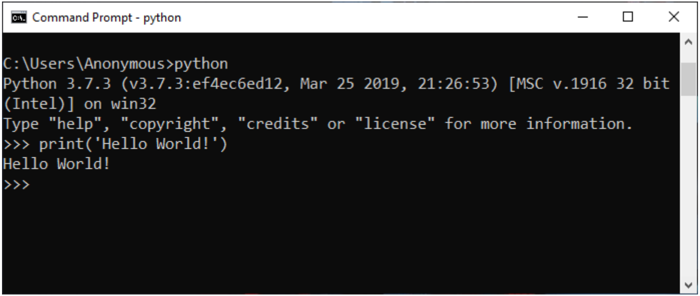
- Codeblocks with mingw compiler
  http://www.codeblocks.org/downloads/26

- Codeblocks with mingw compiler
  `http://www.codeblocks.org/downloads/26`
- Python 3.x with pip
  `https://www.python.org/downloads/windows/`
  - Use the installer option

- Codeblocks with mingw compiler
  `http://www.codeblocks.org/downloads/26`
- Python 3.x with pip
  `https://www.python.org/downloads/windows/`
  - Use the installer option
- Install matplotlib
  - `pip install matplotlib`

- Codeblocks with mingw compiler
  `http://www.codeblocks.org/downloads/26`
- Python 3.x with pip
  `https://www.python.org/downloads/windows/`
  - Use the installer option
- Install matplotlib
  - `pip install matplotlib`
- Make sure `python` and `pip freeze` can be executed on a normal command prompt

- Codeblocks with mingw compiler
  `http://www.codeblocks.org/downloads/26`
- Python 3.x with pip
  `https://www.python.org/downloads/windows/`
  - Use the installer option
- Install matplotlib
  - `pip install matplotlib`
- Make sure `python` and `pip freeze` can be executed on a normal command prompt
- Download the experiment codes from
  `https://drive.google.com/file/d/1b2I-x7tNHYapKHRRoxjeD8qYFmonk6fg`



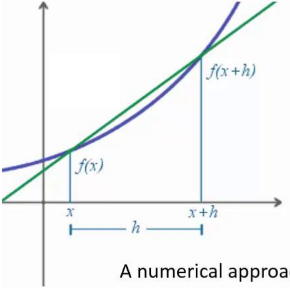
Try importing matplotlib by writing down:

```
>>> import matplotlib
```

Outline

Outline

Numerical Differentiation



The derivative of a function $y = f(x)$ is a measure of how y changes with x.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

A numerical approach to the derivative of a function $y = f(x)$ is:

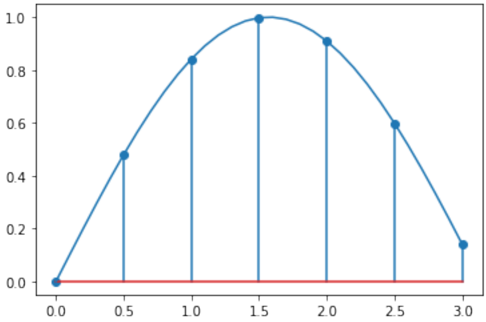
$$\frac{dy}{dx} = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

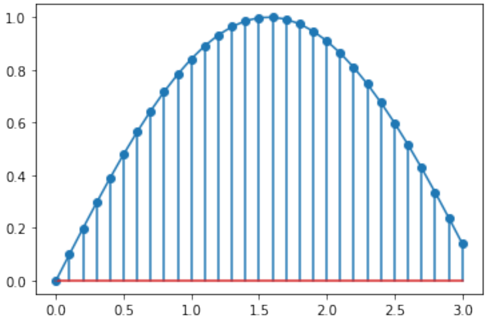
Outline

Header file (Differentiator.h)

```
#ifndef SIMULATION_DIFFERENTIATOR_H
#define SIMULATION_DIFFERENTIATOR_H
class Differentiator {
    public:
        Differentiator(double samplingTime);
        void Input(double data);
        double Output();
        ~Differentiator() {}
    private:
        double ts, output , input , oldInput;
};
#endif //SIMULATION_DIFFERENTIATOR_H
```

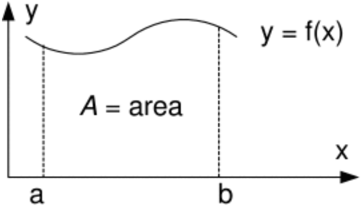
```
#include "Differentiator.h"

// Constructor
Differentiator::Differentiator(double samplingTime) {
    ts = samplingTime;
    oldInput = input = 0;
    output = 0;
}

void Differentiator::Input(double data) {
    input = data;
    output = (input - oldInput) / ts;
    oldInput = input;
}

double Differentiator::Output() {
    return output;
}
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

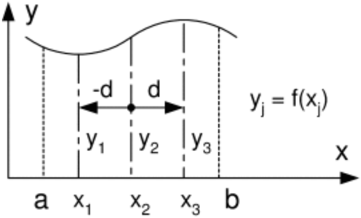

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

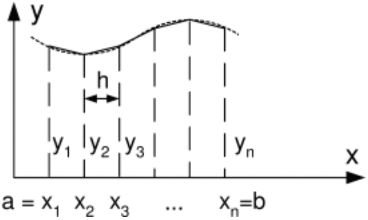
Outline

Numerical Integration



$$A = \int_a^b y\,dx \approx \sum_j w_j\, f(x_j)$$





Outline

Make a C++ program that simulate 100 seconds of the closed loop shown below with 0.01 sec sampling time:

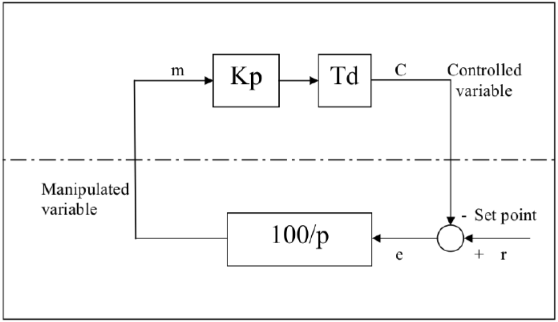


Figure 5.7: **Control Loop**

Assume that:
Kp = 1
Td = 1 sec
R = 50%
P = 150
B = 50% (The bias when the error is zero.)

Make a C++ program that simulate 100 seconds of the closed loop shown below with 0.01 sec sampling time:

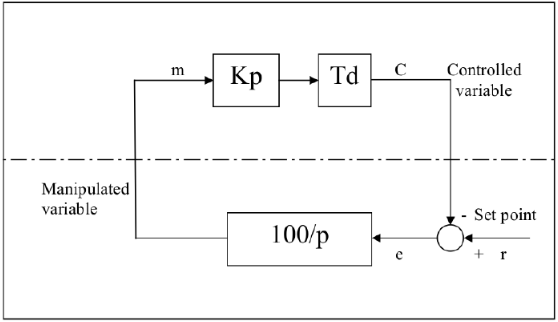


Figure 5.7: **Control Loop**

Assume that:
Kp = 1
Td = 1 sec
R = 50%
P = 150
B = 50% (The bias when the error is zero.)

```cpp
#include "Plot.h"
#include "Deadtime.h"

int main() {
  Plot plt;

  double samplingTime = 0.01,
         Td = 1,
         P = 150,
         B = 50,
         r = 50.0,
         c = 0,
         m = 0,
         e = 0;

  Deadtime deadtime(samplingTime, Td);

  for (double t = 0; t < 100; t += samplingTime) {
    e = r - c;
    m = (100.0 / P) * e + B;
    deadtime.Input(m);
    c = deadtime.Output();
    plt.addPoint(t, {r, c});
  }

  plt.show();
}
```