Pan African University
Institute of Water
and Energy Sciences

EEUEM21 Textbook

Applied Computation In Energy Engineering

Prof. Lotfi BAGHLI

MCF at Université de Lorraine - ENSEM

EEUEM21 Textbook

Applied Computation In Energy Engineering

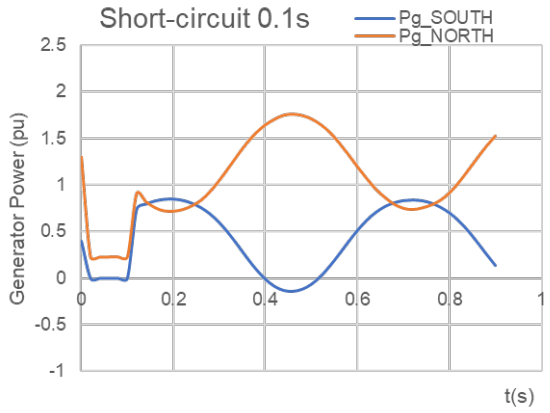
Prof. Lotfi BAGHLI

MCF at Université de Lorraine - ENSEM

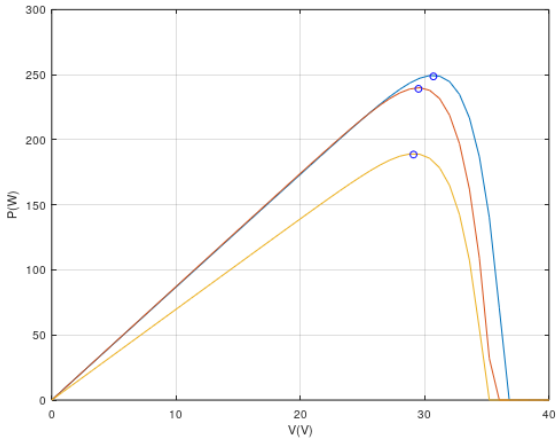
Context of the course

This course will build a solid background in computation methods to solve problems in electrical and energy engineering.

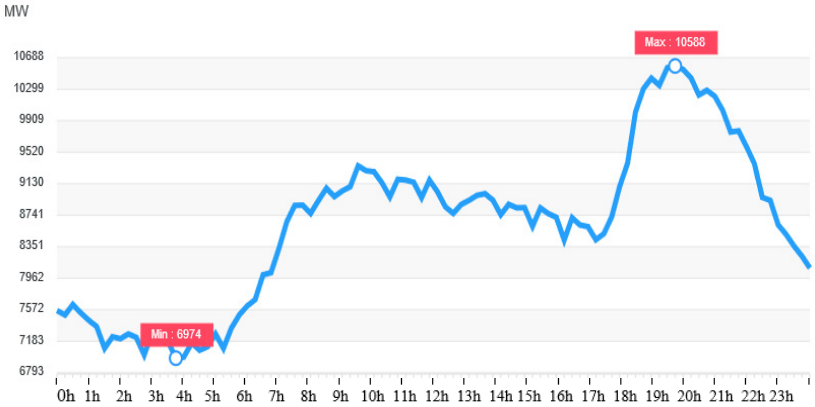
It will provide students with essential concepts of programming, numerical methods, prediction, optimization, machine learning, modelling and control needed to implement frontier technologies applications in energy systems and power grids.



Power swing between synchronous generators during a fault



Maximum Power Point Tracking for a PV system



Daily load curve in a power system

Illustration of Applied computation In Energy Engineering

1. Introduction to energy and computation 8
1.1. Introduction 8
1.2. Current and future research directions 8
1.2.1. Technology to ensure the operating 8
1.1. References 10
2. Programming for Energy Engineering 12
2.1. Introduction 12
2.2. Course content 13
2.3. Summary 14
3. Language and tools 15
3.1. Languages 16
3.2. Octave 17
3.2.1. Installation of Octave 17
3.3. C/C++ 19
3.4. Installation of CLion and tools 20
3.5. Installation of VSCode and tools 22
3.6. PHP, HTML and MySQL 24
3.7. Summary 24
4. Language basics 25
4.1. Learning a programming language 25
4.2. Using Help 27
4.2.1. C/C++ 27
4.2.2. Octave 28
4.3. Program structure 29
4.3.1. C/C++ 29
4.3.2. Using C++ 33
4.4. for statement 33
4.4.1. In C 33
4.4.2. In C++ 34
4.4.3. In Octave 36
4.5. if statement 36
4.5.1. In C++ 36
4.5.2. In Octave 38

4.6. Console Read – Write 38
4.6.1. In C++ 38
4.6.2. In C 38
4.6.3. In Octave 39
4.7. File Read – Write 39
4.7.1. In C++ 39
4.7.2. In Octave 40
4.8. Arrays 41
4.8.1. In C/C++ 41
4.8.2. In C++ 42
4.8.3. In C 42
4.8.4. In Octave 42
4.9. Plot data 43
4.9.1. In C/C++ 43
4.9.2. In Octave 44
4.10. Summary 45
5. Numerical methods and Application 46
5.1. Fixed-point theorem 46
5.1.1. Algorithm 46
5.1.2. Program 47
5.2. Newton-Raphson 49
5.2.1. Algorithm 49
5.2.2. Program 50
5.2.3. Assessment: Assignment 51
5.3. Gauss-Seidel 53
5.3.1. Algorithm 53
5.3.2. Program 54
5.3.3. Assessment: Assignment 56
5.4. Numerical integration 58
5.4.1. Euler Algorithm 58
5.4.2. Program 59
5.4.3. Runge Kutta 4 Algorithm 59
6. Application of numerical methods 61

6.1. Introduction 61
6.2. Simulation of a DC Motor 61
6.3. Assignment, Simulation of a DC Motor 65
6.4. Current control of a DC Motor 67
6.5. Assignment, Current control of a DC Motor 71
6.6. PWM harmonics cancellation 72
6.7. Assignment, PWM harmonics cancellation 76
6.8. Sun position and PV panel orientation 77
6.8.1. Optimal PV panel orientation 78
6.8.2. Example for Tlemcen: 80
6.9. Maximum Power Point Tracking of a PV panel 81
6.9.1. Model of a PV panel, MPPT algorithm 81
6.9.2. Program 84
6.9.3. Activity: Assessment 86
6.10. Assignement: BMS battery emulator 87
6.11. Summary 88
7. Load flow in a power grid 89
7.1. Introduction 89
7.1. Power definition 91
7.2. Power transfer 91
7.3. Power grid elements 91
7.4. Gauss-Seidel algorithm 93
7.5. Load Flow – GS Application 97
7.6. Newton-Raphson algorithm 99
7.7. Load Flow – NR Application 103
7.8. DC Load Flow method 103
7.9. Optimal Load Flow 104
7.10. Summary 106
8. Transient stability of a power grid 107
8.1. Introduction 108
8.2. Power grid elements 108
8.3. Simulation algorithm 115
8.4. Transient stability – Application 115

8.5. Summary 121
9. Artificial Intelligence (AI) and metaheuristics 122
9.1. Introduction 124
9.2. Fuzzy Logic 124
9.2.1. Fuzzification 125
9.2.2. Inference 125
9.2.3. Defuzzification 127
9.3. Fuzzy controller 128
9.4. Neural network 131
9.4.1. Perceptron 131
9.4.2. Radial Basis Function (RBF) 133
9.5. Applications 135
9.1. Recurrent Neural Networks 140
9.2. The Problem of Long-Term Dependencies 140
9.3. LSTM Networks 141
9.4. How to implement the models using open source software libraries 143
9.5. Metaheuristics and Optimization 143
9.6. Genetic Algorithm (GA) 148
9.6.1. Organizational chart. 149
9.6.2. GA code 150
9.7. Particle Swarm Optimization (PSO) 152
9.8. Practical application, example 1 158
9.9. Practical application, example 2 162
9.10. Practical application, example 3 163
9.11. PSO vs GA 165
9.12. Summary 166
10. Machine Learning 167
10.1. Introduction 167
10.2. Machine learning categories 168
10.3. Supervised Learning 168
10.4. Unsupervised Learning 169
10.5. Choosing the right algorithm 169
10.6. Usage of Machine Learning 170

10.7. Machine Learning Challenges 171
10.8. Training a Model to Classify Physical Activities 172
10.8.1. Step One: Load the Data 173
10.8.2. Step Two: Preprocess the Data 173
10.8.3. Step Three: Derive Features 174
10.8.4. Step Four: Build and Train the Model 175
10.8.5. Step Five: Improve the Model 177
10.9. Applying Unsupervised Learning 178
10.9.1. Common Hard Clustering Algorithms 179
10.9.2. Common Soft Clustering Algorithms 182
10.9.3. Improving Models with Dimensionality Reduction 184
10.9.4. Common Dimensionality Reduction Techniques 185
10.9.5. Using Principal Component Analysis 185
10.9.6. Using Factor Analysis 186
10.9.7. Using Nonnegative Matrix Factorization 186
10.9.8. Next Steps 187
10.10. Application: Daily load curves analysis and clustering 188
10.1. Linear regression models and the least squares method 191
10.1.1. Introduction example 191
10.1.2. Why regression 193
10.1. Application: Fitting Daily load curves 193
11. Control and planning 196
11.1. Control of energy production systems 196
12. Current and future research directions 200
12.1. Smart grid, intelligent buildings, generation planning and other innovations 200
13. Case studies: Group activity 203
13.1. Introduction 203
13.2. Study content 204
13.3. Case study 1: smart energy meter 205
13.4. Case study 2: DC small connected micro grids 207
13.5. Summary 209

1. Introduction to energy and computation

Introduction to energy and computation

1.1. Introduction

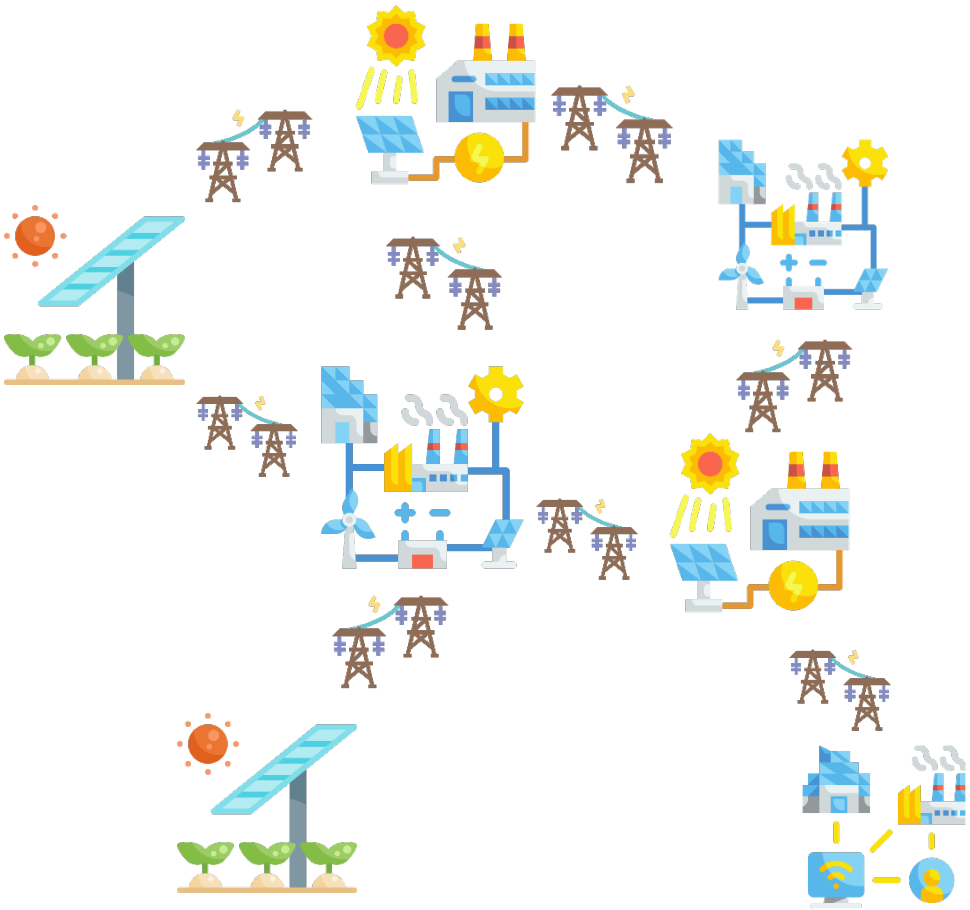
The world of energy is changing with the massive introduction of renewable We are moving from traditional centralized energy production systems to distributed ones.
TSO: Transmission System Operator

1.2. Current and future research directions

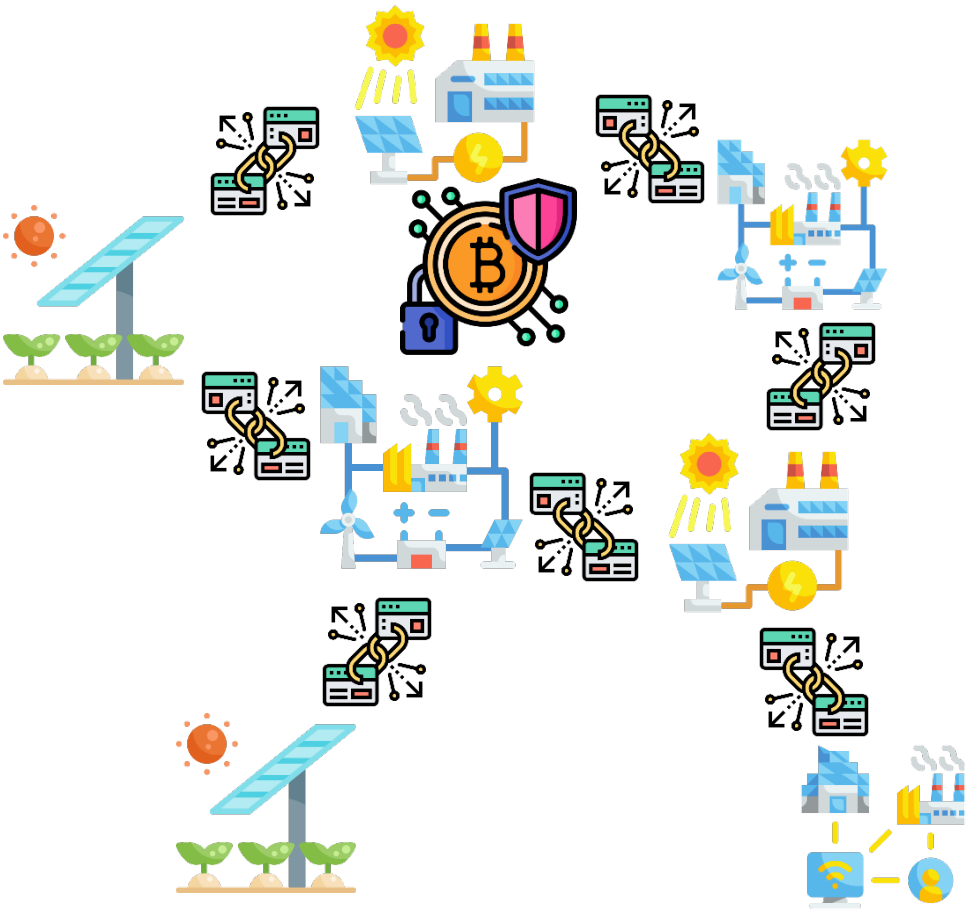
- Aggregation of electrical Microgrids
- Smartgrids, what is the difference with Microgrids
- Intelligent buildings, low carbon footprint, high energy quality (efficiency, losses, insulation,…), secured connections, data monitoring, auto driven by the DSO

1.2.1. Technology to ensure the operating

- Control of DC/AC interconnection converters to the AC loop
- Stability, resilience, degraded regime
- Grid forming, grid following, hybrid mode
- Load profile, PV power, wind
- Distributed optimum transport, Load flow, Transient Stability, L. Baghli
- Project LEAP-RE H2020 MG-FARM, 04/2022 – 05/2025, 1.12 M€, L. Baghli, S. Pierfederici



- Communication and transactions of energy packets
- Multi-layer communication protocol
- Independent of TSOs / DSOs (Transmission/Distribution Operators), decentralized and non-proprietary (IEEE, etc.)
- P2P transaction security (blockchain)
- IoT devices of prosumers at the microgrid level, energy exchange in degraded operation



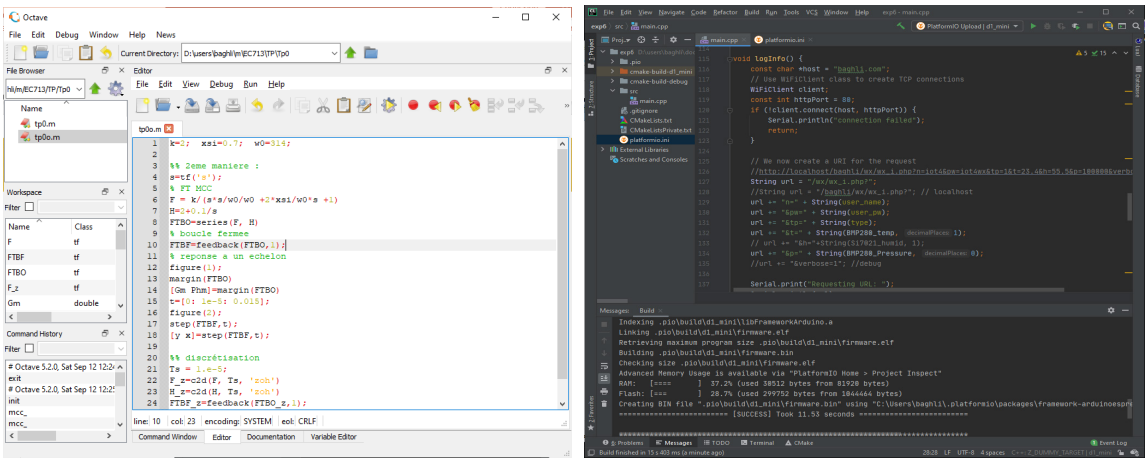
1.1. References

https://fr.mathworks.com/help/?s_tid=gn_supp
https://fr.mathworks.com/help/matlab/language-fundamentals.html
https://fr.mathworks.com/help/simulink/getting-started-with-simulink.html
Getting started with Octave, https://octave.org/doc/v4.2.0/Getting-Started-with-Mex_002dFiles.html
Documentation on Octave https://octave.org/doc/v5.2.0/
Octave Documentation https://octave.org/doc/v5.2.0/The-for-Statement.html#The-for-Statement
https://cplusplus.com/doc/tutorial/
W. H. Press et Al., Numerical recipes in C++, Cambridge University Press, 3rd Ed. 2002, https://e-maxx.ru/bookz/files/numerical_recipes.pdf

Scheid, Francis J., Schaum's outline of theory and problems of numerical analysis, Schaum's outline series, McGraw-Hill, 1989
J. Arrillaga, C. P. Arnold , Computer Analysis of Power Systems, John Wiley & Sons, 1990, http://eword.yolasite.com/resources/51621752-Computer-Analysis-of-Power-Systems.pdf
Stagg, G.W. and El-Abiad, A.h., “Computer Methods in Power System Analysis,” Mc Graw Hill, 1968
L. Baghli, G. Didier, S. Bendali, J. Leveque, An Open Source Real-Time Power System Simulator with HIL, ICEN 2010, Sidi Bel Abbes, 28-29 October, 2010.
G. Andersson, Modelling and Analysis of Electric Power Systems, ETHZ, 2008

2. Programming for Energy Engineering

Programming for Energy Engineering



Have a presentation of the course and his content
We will see why we choose 2 programming languages
and why it is important to master them
Why it is important to do the implementation by yourself

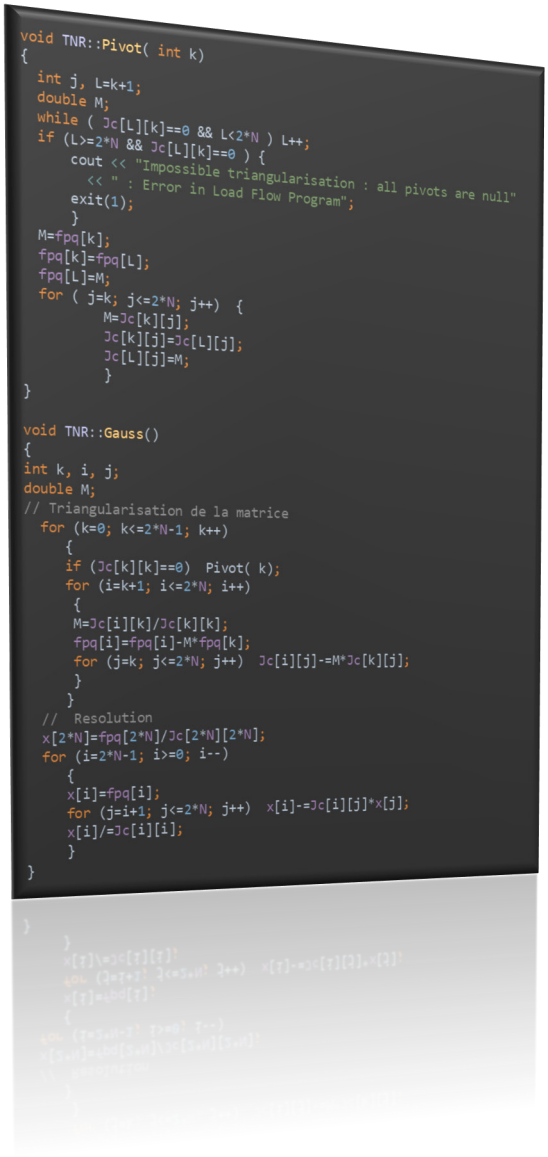
2.1. Introduction

Today everything is digitally controlled
Home automation, administrations, bots that mimic human action and thinking, Internet of Things (IoT), energy systems, cars, thermal and electric ones, industries.
Wi-Fi routers are indoor every where
4G and 5G are meshing the overall outdoor space

We can buy PV systems and conceive the control of the systems by ourselves or buy an already working one, perhaps not optimized the way we want it to behave.
It requires brain and development

Africa has a role to play in this new technology field
Asia played a big role 20 years ago when launching their electronics industry
Old teaching curricula are not adequate
Change the way we learn and how to mix consolidated matter
Students need lot of programming skills if they want to be performant
An average knowledge in programming is enough to deal with energy control and optimization algorithms.

However, we need a good knowledge of energy, power systems and electronics.
Then we can add the knowledge of the modelling, the control methods, in order to do the prediction, the optimization and control of energy.
There is a huge amount of information and study to address concerning applied computation in energy.
We must make choices and chose what are more required than others!



```
void TNR::Pivot( int k)
{
  int j, L=k+1;
  double M;
  while ( Jc[L][k]==0 && L<2*N ) L++;
  if (L>=2*N && Jc[L][k]==0 ) {
      cout << "Impossible triangularisation : all pivots are null"
        << " : Error in Load Flow Program";
      exit(1);
      }
  M=fpq[k];
  fpq[k]=fpq[L];
  fpq[L]=M;
  for ( j=k; j<=2*N; j++)  {
          M=Jc[k][j];
          Jc[k][j]=Jc[L][j];
          Jc[L][j]=M;
          }
}

void TNR::Gauss()
{
int k, i, j;
double M;
// Triangularisation de la matrice
  for (k=0; k<=2*N-1; k++)
     {
     if (Jc[k][k]==0)  Pivot( k);
     for (i=k+1; i<=2*N; i++)
      {
      M=Jc[i][k]/Jc[k][k];
      fpq[i]=fpq[i]-M*fpq[k];
      for (j=k; j<=2*N; j++)  Jc[i][j]-=M*Jc[k][j];
      }
     }
  //  Resolution
  x[2*N]=fpq[2*N]/Jc[2*N][2*N];
  for (i=2*N-1; i>=0; i--)
     {
     x[i]=fpq[i];
     for (j=i+1; j<=2*N; j++)  x[i]-=Jc[i][j]*x[j];
     x[i]/=Jc[i][i];
     }
}
```

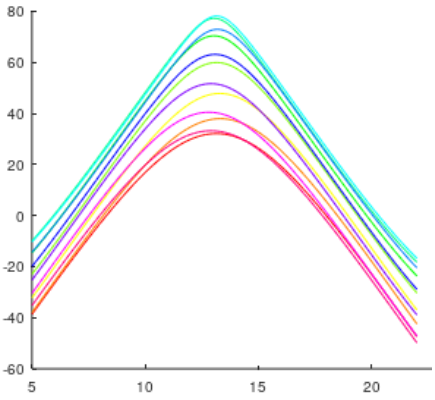
How to pick the right tools?

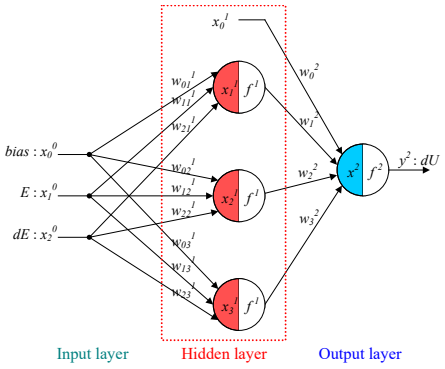
2.2. Course content

In this course, we will begin by choosing the programming languages
Why C/C++ and MATLAB/Octave
With what tools (Integrated Development Environment)
The choice is very important and should be made by a specialist in order to help the student to learn fast, easily and efficiently
The choices ten years ago are different from the choices to be made nowadays and will certainly be different in 10 years

We give some basics of programming using these 2 languages, the compiled C/C++ for Windows and the free Octave .m scripts
For these languages, CLion and Octave are the best free IDE available.
They are very powerful while being free
It is mandatory to test the code by yourself, modify it, put breakpoints, analyze intermediate results…

The numerical methods are also presented as well as how we program them in C++ and in Octave
We present applications from my personal experience and developed programs:
Simulation and control of a DC Motor, PWM harmonics cancellation, load flow and transient stability of a power grid, daily load curves, Wind and PV energy production, …
We also present some new methods to optimize or solve problems, based on Artificial Intelligence (AI) and metaheuristics



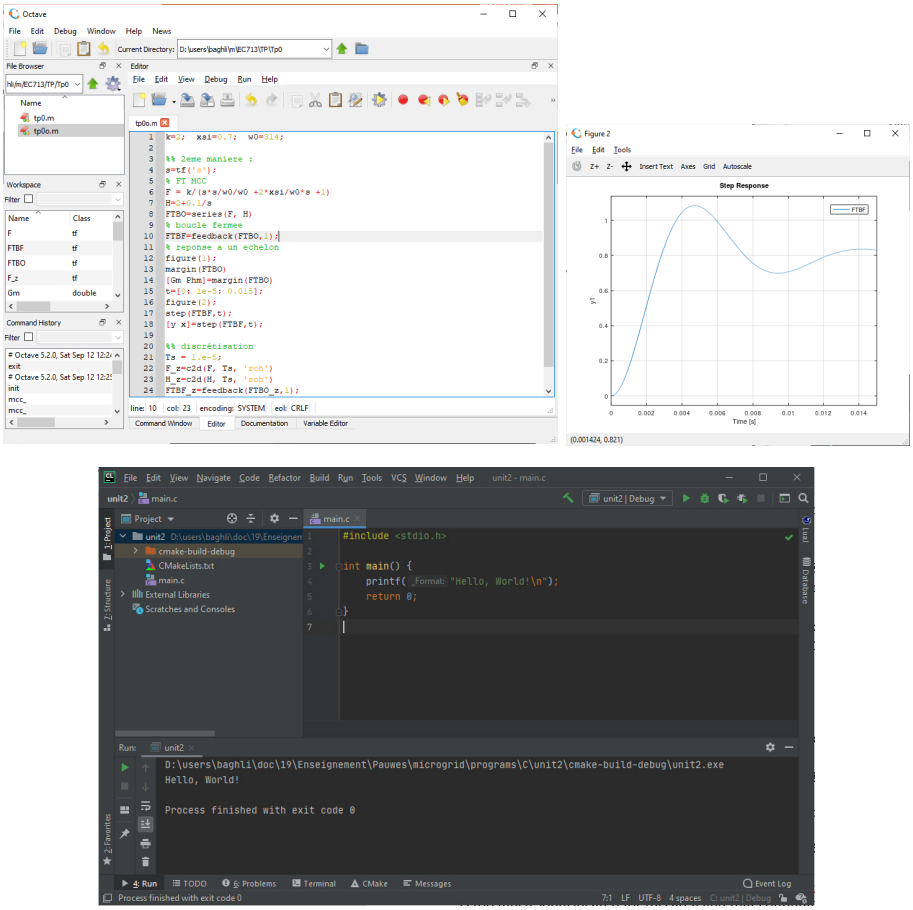


2.3. Summary

This introduction gives an overview of the course content
It shows how it is important to learn programming and
Programing must be practiced individually and intensively

3. Language and tools

Language and tools



Have a presentation of C/C++ language and MATLAB
Explore different Integrated Development Environment (IDE)
Install IDE, compilers and plugin for microcontrollers
Set up your PC to run further examples and exercises

References
Matlab, Eugeniy E. Mikhailov, Programming with MATLAB for Scientists, A Beginner's Introduction, 2017 by CRC Press, Taylor & Francis Group, ISBN: 978-1-4987-3831-6
Octave, https://octave.org/doc/v4.2.0/Getting-Started-with-Mex_002dFiles.html
CLion and MinGW C/C++ compiler, https://www.jetbrains.com/help/clion/quick-tutorial-on-configuring-clion-on-windows.html#MinGW

Installation of CLion for ESP32 (optional, for information only),
https://www.youtube.com/watch?v=OXoTYAOz1GM&list=PLXYd8IyLhtrG0OX2qWBzYkdjslJc6TzzX&index=2

3.1. Languages

Programming Languages
- Instructions in a specific syntax to implement an algorithm
- Used to perform tasks, compute results

Pros and cons
- Compilation takes time, a small change implies recompilation
- Compile to get an executable
- Compiled programs are target dependent
- Compiled program's execution is faster than an interpreted program
- Interpreted languages can be run as scripts (independent commands)
- Faster to learn and to develop with interpreted languages

Choice for this course
- C/C++
- MATLAB

Classification

Compiled languages:
- C/C++
- C#
- Delphi / Pascal
- Visual BASIC

Interpreted languages:
- Python
- PHP
- JAVA
- JavaScript
- MATLAB
- Octave
- Wolfram
- Basic

Array languages:
- MATLAB
- Octave
- Wolfram
- Fortran 90

The Target is the processor that will run the program

For MATLAB, Octave
- Target is the PC (x86…) with an Operating System (OS) as Windows, Mac OS or Linux

For C/C++

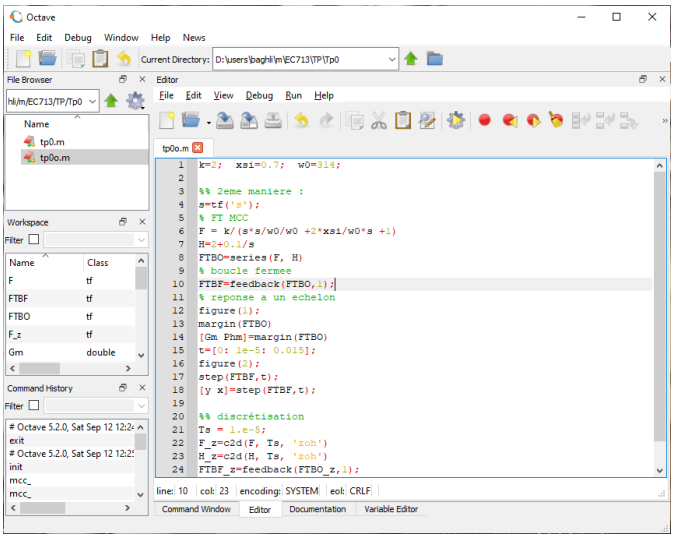
Depending on the (cross) compiler used, we can target different devices:

- Target is the PC which will run the (compiled) executable using the OS (Windows, Mac OS or Linux)
- Target is the uC (microcontroller, DSP, DSC). It has no OS. The compiled program is the only one running on the processor when Flashed (uploaded to its Flash memory).

Examples

- Mingw-w64 is a C/C++ compiler for a PC running Windows OS
- Visual Studio C++ is a C/C++ compiler for a PC running Windows OS
- MPLAB XC16 is a C compiler for Microchip dspic 33FJ microcontrollers

3.2. Octave



Why

Scientific libraries to manipulate arrays (Matrix), transfer functions

For Scientists (Master, PhD studies and beyond)

Easy to manipulate data and plot graphs

Pros and cons

Process and compute slower than pure C/C++ coded algorithm

Easier than C/C++

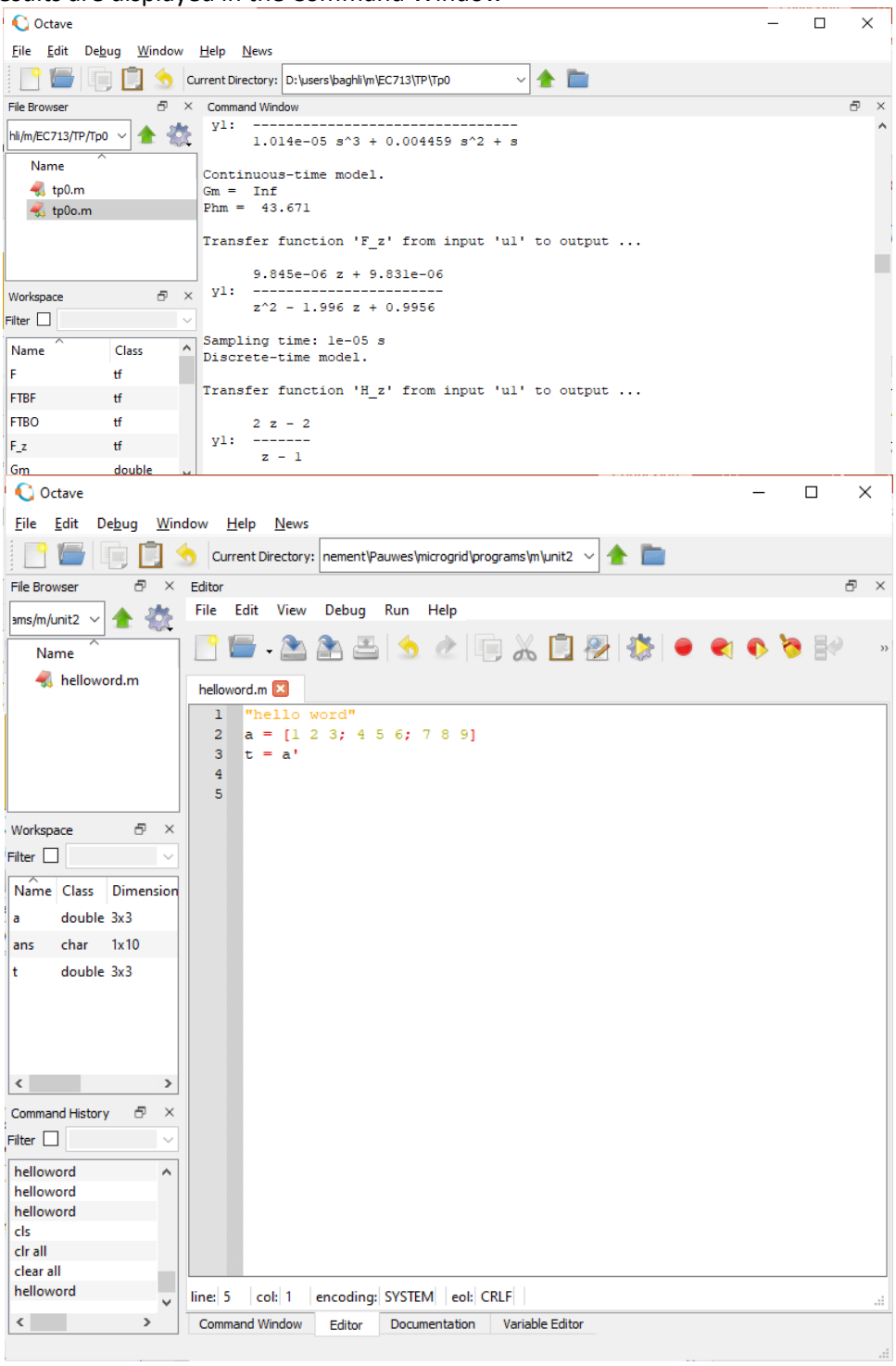
Can include compiled C/C++ or Fortran files to speed-up some calculus, ref: https://octave.org/doc/v4.2.0/Getting-Started-with-Mex_002dFiles.html

3.2.1. Installation of Octave

Octave installation link
https://www.gnu.org/software/octave/download.html#ms-windows
Documentation link
https://octave.sourceforge.io/docs.php
Manual link
https://octave.org/doc/v5.2.0/
Try

- Activate the Command Window by clicking on the Tab
- In the Command Window
  write 2+3*5 and hit enter
- The answer will be displayed
- Load and run “helloword.m” that is available on

- In the Command Window, you type the instructions to be immediately executed or call functions
- In the Editor, you edit the .m scripts
- Pressing F5 or clicking on Run
- Load and run the script “helloword.m” that is available on “Save File and Run” execute the program being edited
- The results are displayed in the Command Window



3.3. C/C++

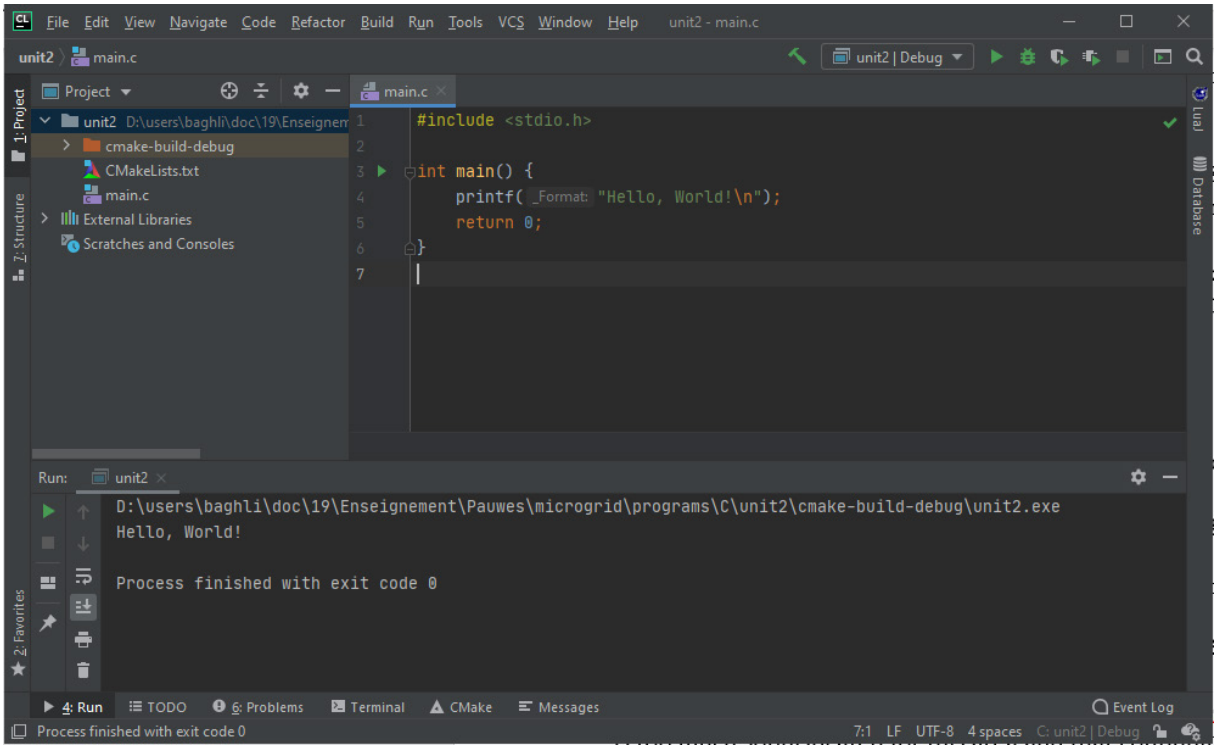
Why

- Need an Integrated Development Environment (IDE) to handle C/C++ projects
- Jetbreans makes the most powerful and intuitive IDEs (CLion, Android Studio, PhpStorm, PyCharm). They cover many languages with the same interface
- Syntax highlighting, debugging, multiple compilers and targets, plugins
- Free for students and teachers

Use CLion and target

- C/C++ for Windows using Mingw-w64 compiler

https://www.jetbrains.com/help/clion/quick-tutorial-on-configuring-clion-on-windows.html#MinGW



Pros and cons

- Universal language used for all targets, very efficient
- C/C++ is the fastest to run, gives most compact executables
- Optimized for execution speed and program size
- Easy to switch from C/C++ to other languages. The opposite is not true
- The widest offer available for a programming language
- Compilers and IDE are available on all platforms (Windows, MacOS, Linux)
- Multiple targets executable (PC with different OS, uC)
- Executables are specific to targets
- Changes in the program source needs recompilation (time consuming)
- IDE are user friendly compared to line command compilers
- One place to edit, compile, flash, run, debug

```c
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

3.4. Installation of CLion and tools

CLion

- Install and register student version of CLion

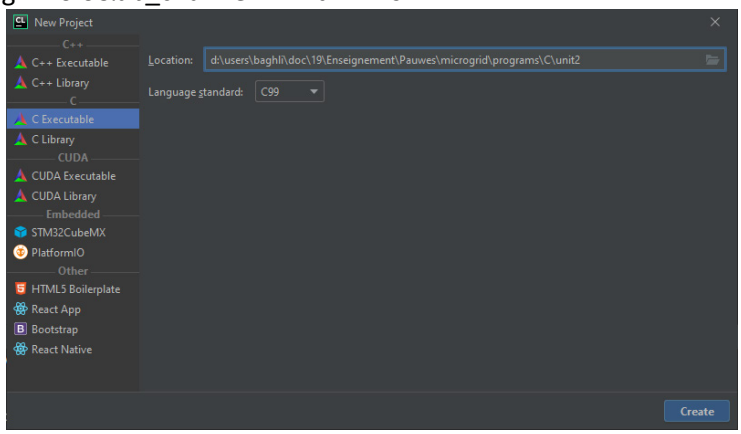
https://www.jetbrains.com/clion/download/#section=windows
https://www.jetbrains.com/community/education/#students

Mingw-w64 compiler

- Download the C/C++ compiler and install it using the explanations provided in this video

https://www.youtube.com/watch?v=9PAglZlRolo&ab_channel=AmitThinks

- Configure and install it as explained for Clion



https://www.jetbrains.com/help/clion/quick-tutorial-on-configuring-clion-on-windows.html#MinGW

- In the preferences as explained above, CLion will detect Mingw-w64 compiler
- Install the C/C++ Single File Execution https://plugins.jetbrains.com/plugin/8352-c-c--single-file-execution/

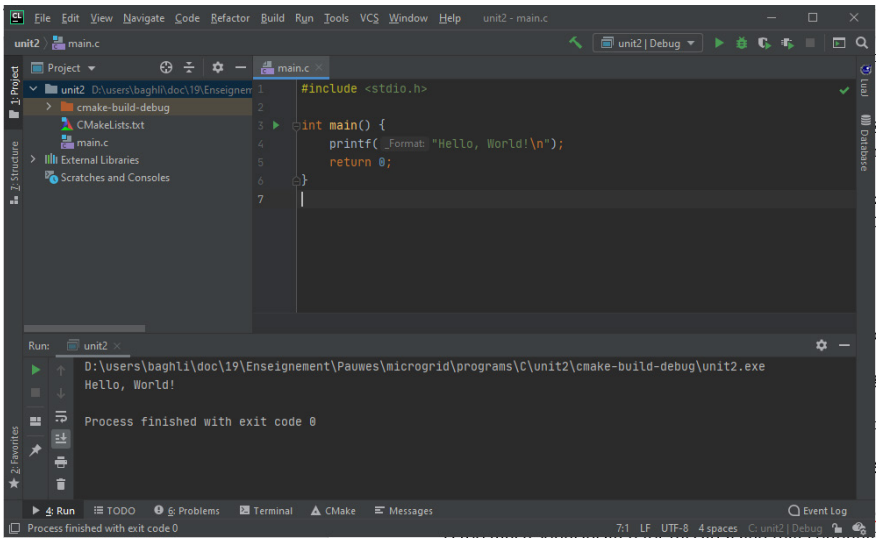
Try

- Try a new project, choose C Executable and the directory to save the project “unit2” and the source file (by default main.c)
- The default code is

Run

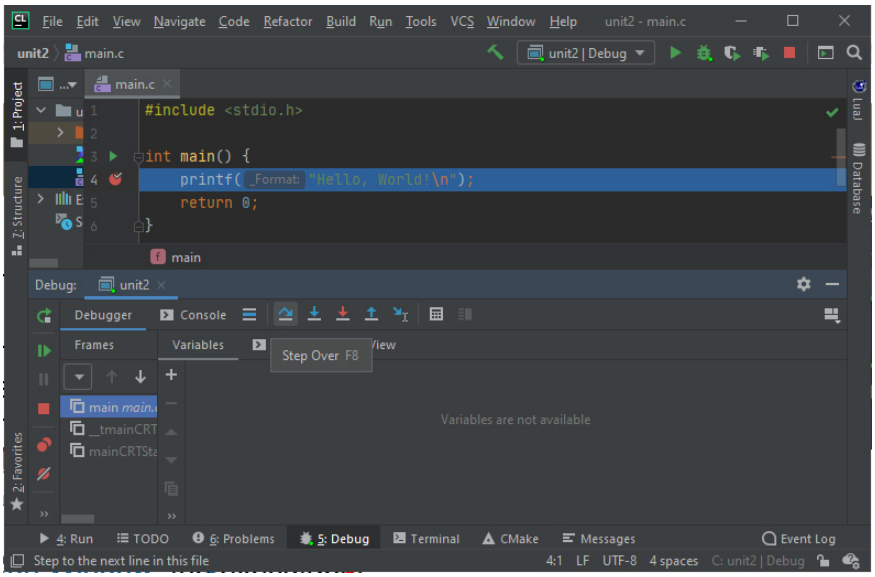
- Press the Play button to Compile and Run the program
- The CMake Window indicates the compilation progress and eventual errors

- In the Run Window, the executable “unit2.exe” is called and the outputted text “`Hello, World!`” is captured and displayed



Debug

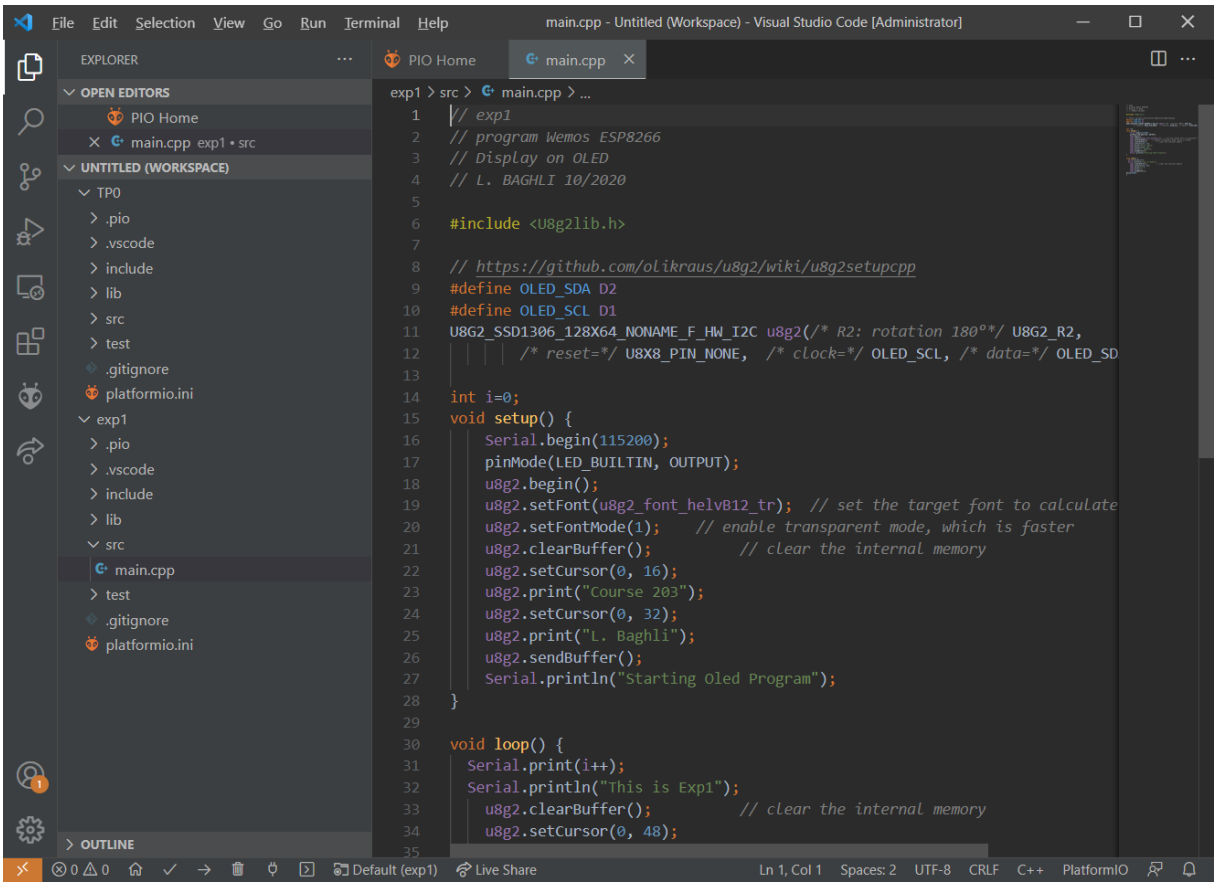
- Click on the left border of the line 4 printf to display a red circle indicating a Breakpoint
- Press the Debugging button to Compile and start Debugging the program
- The program runs and stops right before executing the line 3
- There is no display of the message "`Hello, World!`"
- Press Step Over or press F8 to execute line by line
- More details in unit 3



3.5. Installation of VSCode and tools

VSCode
https://code.visualstudio.com/download

It is and IDE as CLion is, perhaps more known because of Microsoft


File Edit Selection View Go Run Terminal Help
main.cpp - Untitled (Workspace) - Visual Studio Code [Administrator]
EXPLORER
OPEN EDITORS
PIO Home
main.cpp exp1 • src
UNTITLED (WORKSPACE)
TP0
.pio
.vscode
include
lib
src
test
.gitignore
platformio.ini
exp1
.pio
.vscode
include
lib
src
main.cpp
test
.gitignore
platformio.ini
OUTLINE
PIO Home
main.cpp
exp1 > src > main.cpp > ...
// exp1
// program Wemos ESP8266
// Display on OLED
// L. BAGHLI 10/2020
#include <U8g2lib.h>
// https://github.com/olikraus/u8g2/wiki/u8g2setupcpp
#define OLED_SDA D2
#define OLED_SCL D1
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(/* R2: rotation 180°*/ U8G2_R2,
/* reset=*/ U8X8_PIN_NONE, /* clock=*/ OLED_SCL, /* data=*/ OLED_SD
int i=0;
void setup() {
Serial.begin(115200);
pinMode(LED_BUILTIN, OUTPUT);
u8g2.begin();
u8g2.setFont(u8g2_font_helvB12_tr); // set the target font to calculate
u8g2.setFontMode(1); // enable transparent mode, which is faster
u8g2.clearBuffer(); // clear the internal memory
u8g2.setCursor(0, 16);
u8g2.print("Course 203");
u8g2.setCursor(0, 32);
u8g2.print("L. Baghli");
u8g2.sendBuffer();
Serial.println("Starting Oled Program");
}
void loop() {
Serial.print(i++);
Serial.println("This is Exp1");
u8g2.clearBuffer(); // clear the internal memory
u8g2.setCursor(0, 48);
Default (exp1)
Live Share
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
C++
PlatformIO

3.6. PHP, HTML and MySQL

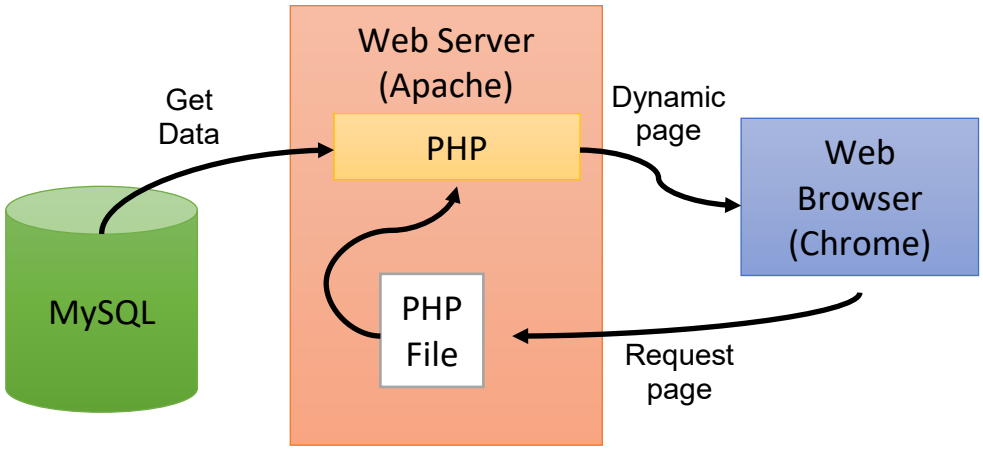
PHP, HTML and MySQL are used to interact with web stored data and to display them
HTML, HyperText Markup Language

It is the standard markup language to create Web pages

HTML pages are stored on a server running a specified software called web server (Apache is a free web server software)

HTML page can be dynamically generated using PHP
The Web server runs PHP. It is an interpreter that will load the PHP code, parse it and then execute it

It gets data from a database (MySQL is free and widely used) in order to prepare a dynamic page that is sent (served) to the client web browser that made the request

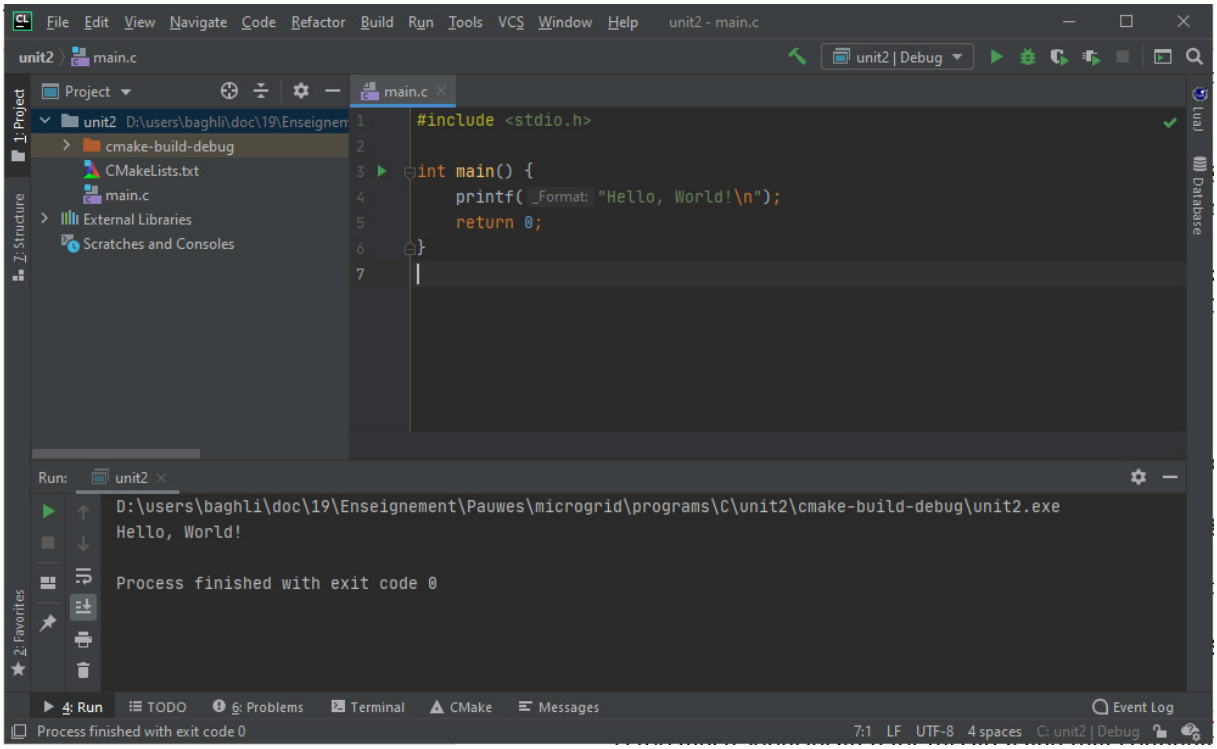


3.7. Summary

IDE are software tools to edit, compile, execute, (flash) and debug the programs
There are compiled and interpreted languages

4. Language basics

Language basics



Have syntax elements of C/C++ language and MATLAB / Octave
Explore loops, conditions, console and file read write, arrays, plot results

References (APA style)
C++ Tutorials and help http://www.cplusplus.com/doc/tutorial/control/
C++ shell (online compiler) http://cpp.sh
C++ Tutorials https://www.w3schools.com/cpp/default.asp
C Tutorial https://www.javatpoint.com/c-programming-language-tutorial
Octave Documentation https://octave.org/doc/v5.2.0/The-for-Statement.html#The-for-Statement
Learning C++ https://www.codecademy.com/catalog/language/c-plus-plus
C in 20 hours (french), http://c.developpez.com/tutoriels/20-heures/c-20-heures.pdf

4.1. Learning a programming language

It is better to learn well a useful programing language than know approximately many of them

C/C++ is universal, powerful and appropriate for Windows, Linux, MacOS targets as well as for all microcontrollers

It is the best choice for microcontrollers, IoT and embedded systems developers
It is also the best one in term of computational speed (fast) and program size (small footprint).
MATLAB and its free clone Octave is the reference for scientists and engineers
Be rigorous in syntax helps in fast learning.
An IDE like CLion corrects syntax typo and is of great help to concentrate on what is central: the algorithm
Make your code clean and readable. Adopt naming conventions
Compiled languages:

- C/C++
- C#
- Delphi / Pascal
- Visual BASIC

Interpreted languages:

- Python
- PHP
- JAVA
- JavaScript
- MATLAB
- Octave
- Wolfram
- Basic

Array languages:

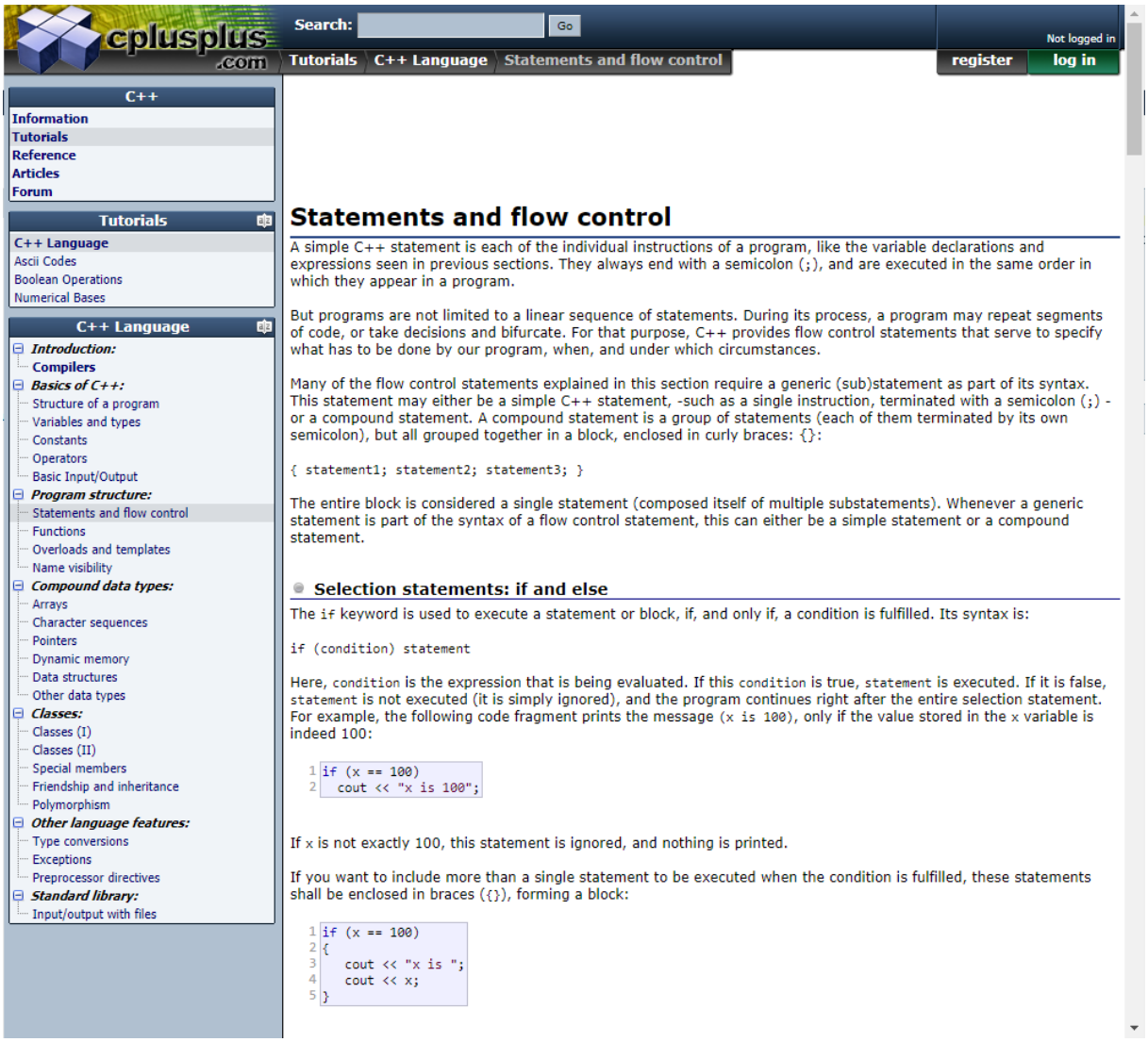
- MATLAB
- Octave
- Wolfram
- Fortran 90

Learning by your self, use the online help, tutorials and forums
Practice, practice and practice
Try solving problems
Enjoy programming!

4.2. Using Help

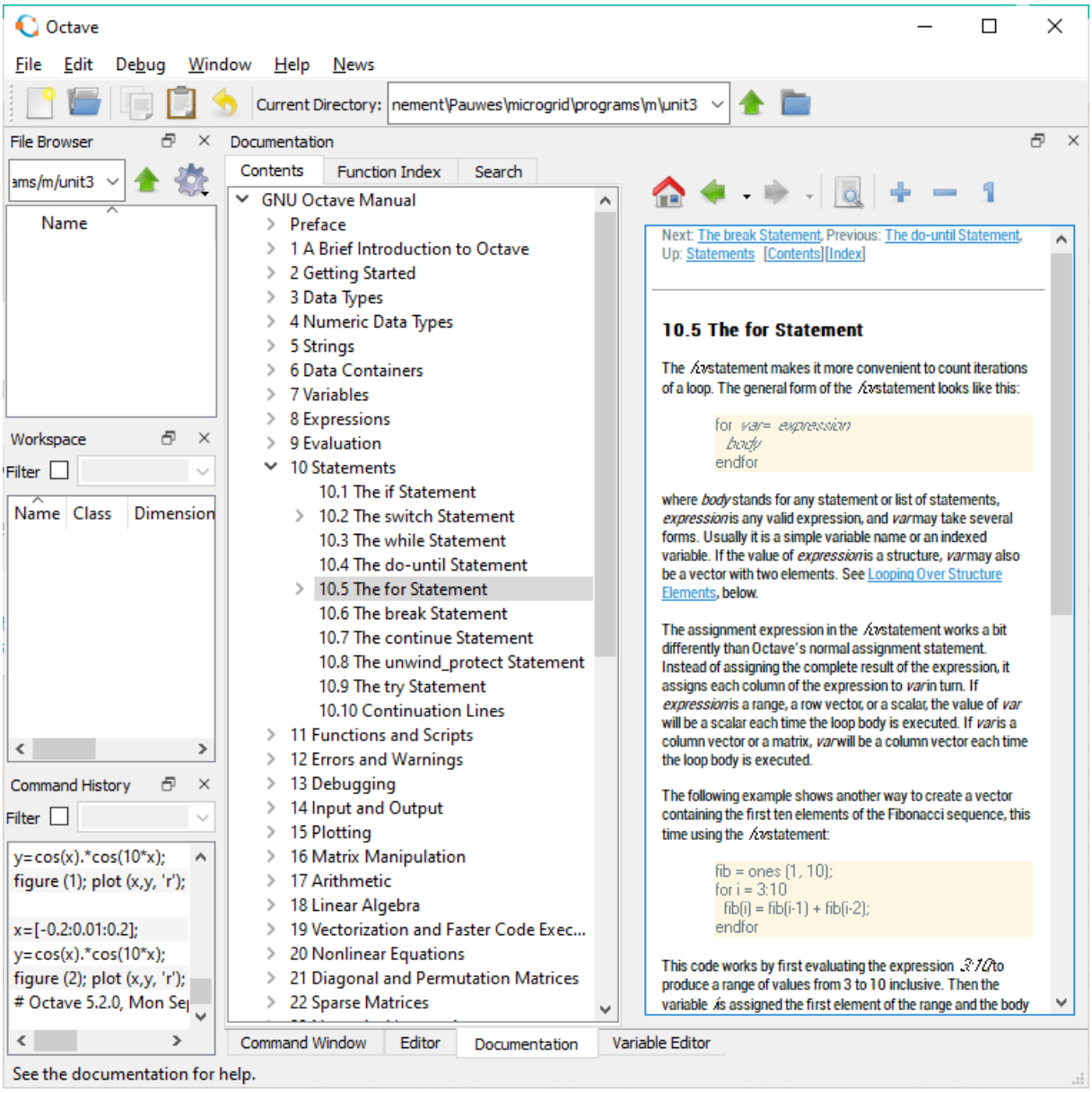
4.2.1. C/C++

- External PDF document, ref: http://c.developpez.com/tutoriels/20-heures/c-20-heures.pdf
- External online help, ref: http://www.cplusplus.com/doc/tutorial/control/
- https://www.w3schools.com/cpp/cpp_for_loop.asp
- https://www.javatpoint.com/c-loop
- https://c.developpez.com/cours/20-heures/



4.2.2. Octave

- Internal documentation, in the Tab Documentation in Ocatve IDE
- External PDF document, ref: Programming with MATLAB for Scientists A Beginner's Introduction by Eugeniy E. Mikhailov (z-lib.org)
- External online help, ref: https://octave.org/doc/v5.2.0/The-for-Statement.html#The-for-Statement



4.3. Program structure

4.3.1. C/C++

```cpp
// my first program in C++
#include <iostream>

int main() {
    std::cout << "Hello, World!";
    return 0;
}
```

Line 1: `// my first program in C++`
Two slash signs indicate that the rest of the line is a comment (not compiled)

Line 2: `#include <iostream>`
Lines beginning with a hash sign (#) are directives to the preprocessor. This one instructs the preprocessor to include a section of standard C++ code, known as header iostream, that allows to perform standard input and output operations to the screen

Line 4: `int main()`
This line initiates the declaration of a function. A function is a group of code statements which are given a name (here "main“). Function will be discussed later
The function named main is a special function in all C/C++ programs; it is the first function called when the program is run.

Lines 5 and 7: { and } indicates the beginning and the end of a function definition

Line 6: `std::cout << "Hello, World";`
This line is a C++ statement. A statement is an expression that can actually produce some effect. Statements are executed in the same order that they appear within a function's body.

The semicolon (;) marks the end of the statemen
All C++ statements must end with a semicolon character

Executing C/C++ programs online
Using only the Web browser

- http://cpp.sh/
- https://ideone.com/
- https://repl.it/languages/c

Better use CLion with its compiler or other C/C++ IDE

C/C++ Keywords

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

Reserved words that cannot be used as a variable name, constant name, function name

Variable

It is a name of the memory location. It is used to store data. Its value can be changed, and it can be reused many times

Variables must be declared before used

Local variables are declared inside the definition of a function and are know only there

Global variables are declared outside the function. Any function can change the value of the global variable. It is available to all the functions

Types	Data Types
Basic Data Type	int, char, float, double
Derived Data Type	array, pointer, structure, union
Enumeration Data Type	enum
Void Data Type	void

Types

Common types are integer, double, characters. Can be modified by:
unsigned, short, long,
Strings in C are char arrays with \0 additional terminated character
In C++, we prefer use the Class string http://www.cplusplus.com/reference/string/string

```
int a;
double b = 2.33;
char c;
```

```
int value=20; //global variable
void function1(){
    int x=10; //local variable
}
```

C++ classes

Classes are an expanded concept of data structures: like data structures, they can contain data members but also functions members.

```
class Rectangle {
    int width, height;
public:
    void set_values (int,int);
    int area (void);
} rect;
```

https://www.javatpoint.com/keywords-in-c
https://www.javatpoint.com/variables-in-c

Octave/Matlab functions

```
function [Vs, Is, Ps] = plotpv(Suns,Tc)
function I = PVNR(V,G,T)
```

C/C++ functions

The program is structured in segments of code that perform individual tasks. They are called functions

type name (parameter1, parameter2, ...) { statements }

The function moy takes 2 parameters of type double
and return a value of type double

```
#include <iostream>
using namespace std;

double moy(double a, double b) {
    double r;
    r = (a + b) / 2.;
    return r;
}

int main() {
    double z;
    z = moy(18, 15.5);
    cout << "The result is " << z;
}
```

Like the variables, the functions must be declared before being used
If the definition (corps of the function) appears after being called in the sequence, the declaration must be done.
The prototype of a function can be declared without actually defining the function completely, giving just enough details to allow the types involved in a function call to be known.
Note the ; at the end of the declaration

```
#include <iostream>
using namespace std;
double moy(double a, double b);

int main() {
```

```
    double z;
    z = moy(18, 15.5);
    cout << "The result is " << z;
}

double moy(double a, double b) {
    double r;
    r = (a + b) / 2.;
    return r;
}
```

C/C++ functions

By default, the arguments are passed by value

To allow the modification of the passed variables, we must pass the arguments by reference

```
#include <iostream>

using namespace std;

double moy(double a, double b) {
    double r;
    r = (a + b) / 2.;
    return r;
}

void doubleit(double &a, double &b) {
    a *= 2;
    b *= 2;
    return;
}

int main() {
    double z;
    z = moy(18, 15.5);
    cout << "The result is " << z << endl;
    double x = 3.5, y = 4.5;
    doubleit(x, y);
    cout << "new values are  x=" << x << " y=" << y << endl;
}
```

In C, the syntax is more complex, since we pass the addresses of the variables

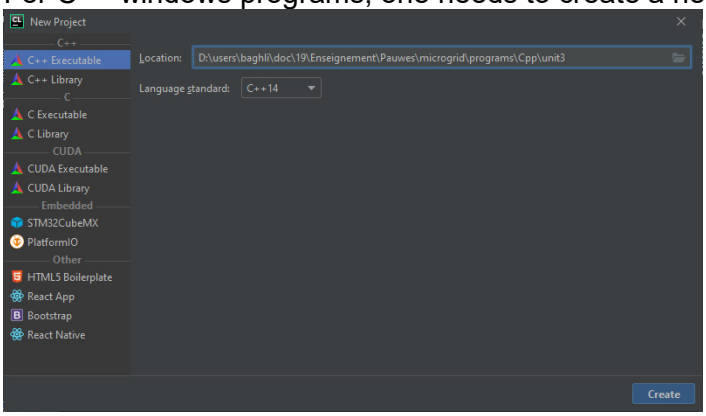
```
void doubleit(double *a, double *b) {
    *a *= 2;
    *b *= 2;
    return;
}
```

```
int main() {
    double x = 3.5, y = 4.5;
    doubleit(&x, &y);
}
```

http://www.cplusplus.com/doc/tutorial/functions/

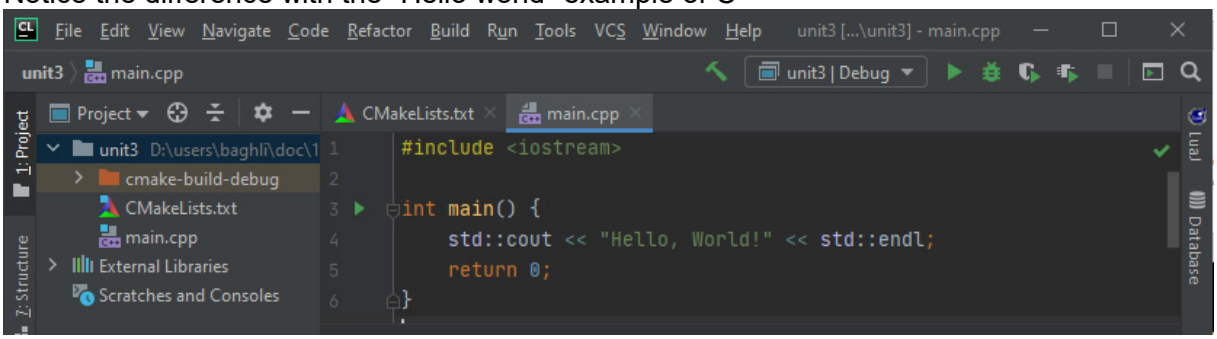
4.3.2. Using C++

For C++ windows programs, one needs to create a new project as illustrated



For C windows programs, it is explained in the previous Unit of the Course. Choose C Executable

Notice the difference with the “Hello world” example of C



C++ uses standard stream output cout

```
std::cout << "Hello, World!" << std::endl;
```

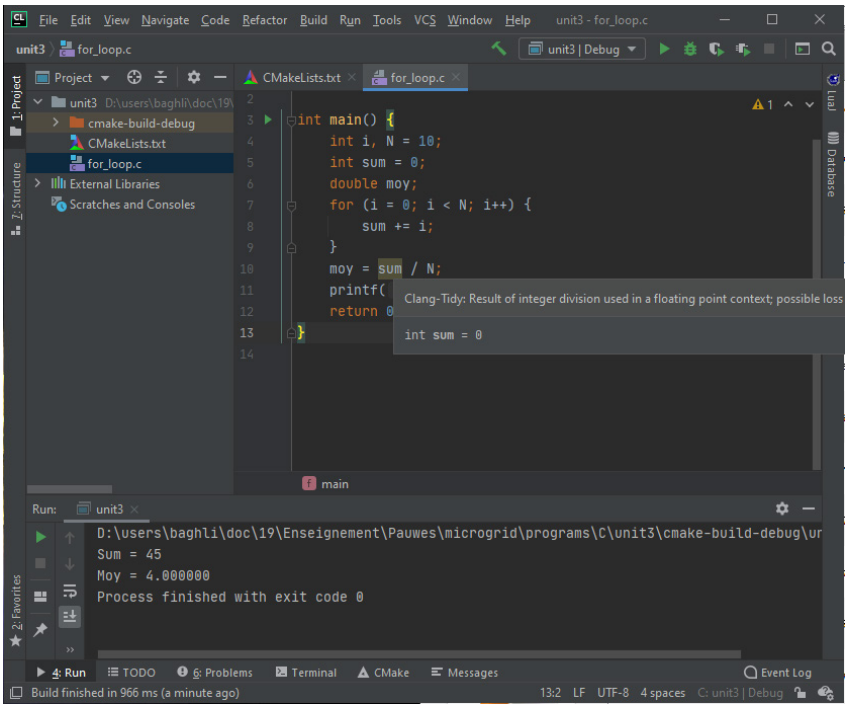
4.4. for statement

4.4.1. In C

Type this program that computes the means of the 10 first numbers

```
int main() {
    int i, N = 10;
    int sum = 0;
    double moy;
```

```
    for (i = 0; i < N; i++) {
        sum += i;
    }
    moy = sum / N;
    printf("Sum = %d\nMoy = %f", sum, moy);
    return 0;
}
```



CLion IDE and MinGW Compiler prevent from a very common error for beginners: A division between 2 integers is an integer value

Correct this by casting one of the 2 operands:

```
    moy = (double)sum / N;
```

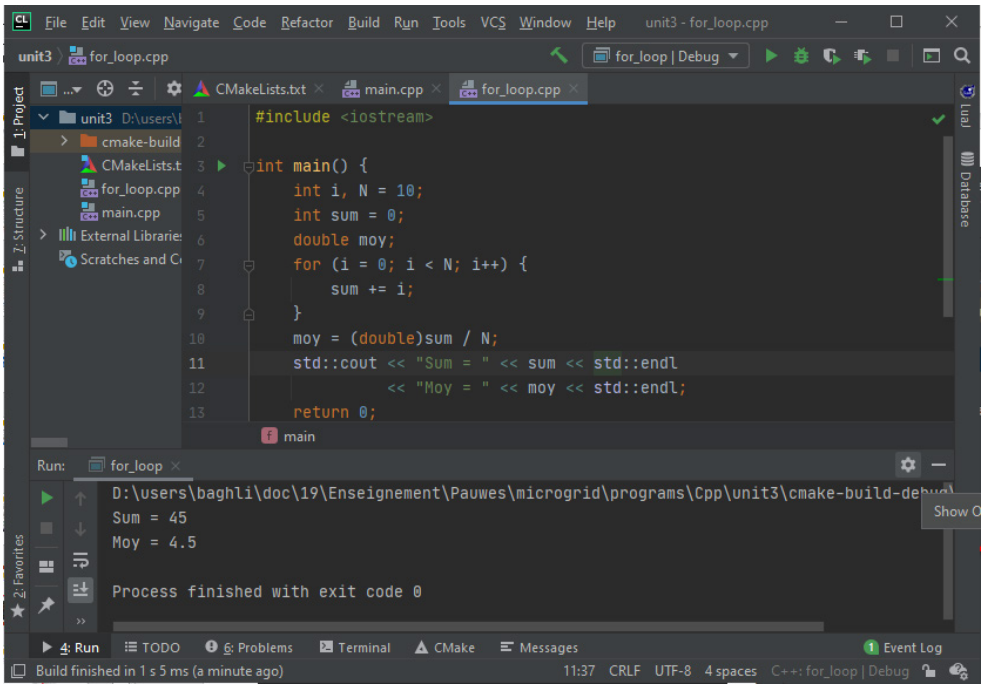
4.4.2. In C++

The structure of the program remains the same except for the header used and the output
In C++, one prefer to use the stream COUT and CIN
http://www.cplusplus.com/reference/iostream/cout/?kw=cout

```
#include <iostream>

int main() {
    int i, N = 10;
    int sum = 0;
```

```
    double moy;
    for (i = 0; i < N; i++) {
        sum += i;
    }
    moy = (double)sum / N;
    std::cout << "Sum = " << sum << std::endl
              << "Moy = " << moy << std::endl;
    return 0;
}
```

Implementing the correct cast of the operand to compute the division gives the correct result 4.5

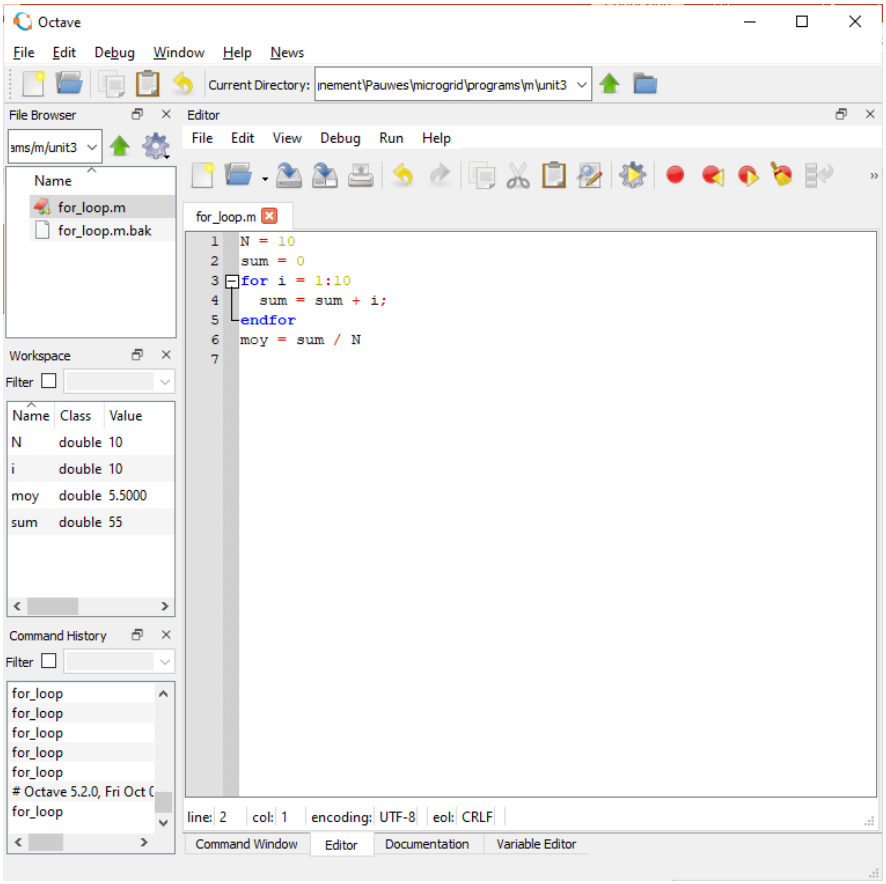


4.4.3. In Octave

The structure of the program is close to C/C++

```
N = 10
sum = 0
for i = 1:10
  sum = sum + i;
endfor
moy = sum / N
```

Notice the ; at the end of a line to prevent from displaying the result
The operation `moy = sum / N` displays the result
In Matlab/Octave, we are used to loop from 1, `for i = 1:10`
because the array indices begins counting from 1
In C/C++, arrays indices start from 0



4.5. if statement

4.5.1. In C++

The if keyword is used to execute a statement or block, if, and only if, a condition is fulfilled
http://www.cplusplus.com/doc/tutorial/control
Observe different forms with bloc of instruction and with else If

```cpp
#include <iostream>

using namespace std;

int main() {
    double x = 99+1;
    if (x == 100)
        cout << "x is 100" << endl;
    if (x == 100) {
        cout << "x is ";
        cout << x << endl;
    }
    if (x == 100)
        cout << "x is 100" << endl;
    else
        cout << "x is not 100" << endl;
    return 0;
}
```

```cpp
if (x > 0)
    cout << "x is positive";
else if (x < 0)
    cout << "x is negative";
else
    cout << "x is 0";
```

A condition is evaluated to a logical true or false and is used mainly in if, while. It needs ( ) around it
http://www.cplusplus.com/doc/oldtutorial/operators/

```cpp
( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

Do not confound && with & which is the Bitwise Operator AND
These operators are used
a lot in programming
microcontrollers

operator	asm equivalent	description
&	AND	Bitwise AND
|	OR	Bitwise Inclusive OR
^	XOR	Bitwise Exclusive OR
~	NOT	Unary complement (bit inversion)
<<	SHL	Shift Left
>>	SHR	Shift Right

4.5.2. In Octave

The if keyword is used to execute a statement or block, if, and only if, a condition is fulfilled
https://octave.org/doc/v5.2.0/The-if-Statement.html#The-if-Statement
Condition does not need ( ) as in C/C++

```
if sum>30
  moy = moy*1.5
endif
```

4.6. Console Read – Write

4.6.1. In C++

`cin` to read from the keyboard
`cout` to write to console
http://www.cplusplus.com/doc/tutorial/control

```
#include <iostream>
using namespace std;

int main() {
    double x;
    cout << "Enter a value" << endl;
    cin >> x;
    cout << "The input value is " << x << endl;
    return 0;
}
```

```
Enter a value
1.3
The input value is 1.3
Process finished with exit code 0
```

4.6.2. In C

`scanf` to read from the keyboard
`printf` to write to console
It uses a formatted syntax
Notice the `&x` , the address of the variable is transmitted to the function `scanf` in order to let it change it
Notice the `%lf` , the format of a double variable is long float
For details on format:
http://www.cplusplus.com/reference/cstdio/printf/

```
#include <stdio.h>

int main() {
    double x;
```

```
    printf("Enter a value\n");
    scanf("%lf", &x);
    printf("The input value is %f", x);
    return 0;
}
```

4.6.3. In Octave

To read from the keyboard
https://octave.org/doc/v5.2.0/The-if-Statement.html#The-if-Statement
To display a value
https://octave.org/doc/v5.2.0/Terminal-Output.html#Terminal-Output
Since Octave normally prints the value of an expression as soon as it has been evaluated, the simplest of all I/O functions is a simple expression
If there is no ; at the end of an evaluation, the value of the result is displayed
Or use
disp (x)
To display the value without the variable name

```
a = input ("Pick a number, any number! ")
disp (a)

>> input_output
Pick a number, any number! 3
a =  3
 3
```

4.7. File Read – Write

4.7.1. In C++

To read or write in C++ we use file streams contained in the library `<fstream>`
http://www.cplusplus.com/doc/tutorial/files/
In order to write a file, we open an output file stream `ofstream`
After opening the file that is in absolute or relative directory in respect to the executable
`myfile << "This is a line.\n";` write the data to the file

```
// writing on a text file
#include <iostream>
#include <fstream>

using namespace std;

int main() {
```

```
    ofstream myfile("example.txt");
    if (myfile.is_open()) {
        myfile << "This is a line.\n";
        myfile << "This is another line.\n";
        myfile.close();
    } else cout << "Unable to open file";
    return 0;
}
```

The produced file by the code above is:

[file example.txt]
This is a line.
This is another line.

How to change the absolute and relative path of the file

```
ofstream myfile("c:\\temp\\example.txt");
ofstream myfile("..\\..\\example.txt");
```

To read from a file from the drive, we open an input file stream
`ifstream`
We get the data line by line, until no more lines are available (condition of the `while` loop)
`getline` will return false when no data
If `myfile.is_open()` return false, then the file is not there

```
// reading a text file
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

int main() {
    string line;
    ifstream myfile("example.txt");
    if (myfile.is_open()) {
        while (getline(myfile, line)) {
            cout << line << '\n';
        }
        myfile.close();
    } else cout << "Unable to open file";

    return 0;
}
```

4.7.2. In Octave

Writing txt files is C like
https://octave.org/doc/v5.2.0/Simple-Output.html#Simple-Output
Even the C printf or fprintf format can be used
https://octave.org/doc/v5.2.0/Formatted-Output.html

```
filename = "example.txt";
fid = fopen (filename, "w");
fputs (fid, "This is a line.\n");
fputs (fid, "This is another line.\n");
fclose (fid);
```

Use `fprintf` instead of `fputs` for Matlab. `fputs` works in Octave

```
filename = "example.txt";
fid = fopen (filename, "w");
fputs (fid, "This is a line.\n");
fputs (fid, "This is another line.\n");
pct = 37;
fprintf (fid,"Processed %d%% of '%s'.\nPlease be patient.\n",
      pct, filename);
fclose (fid);
```

```
[file example.txt]
This is a line.
This is another line.
Processed 37% of 'example.txt'.
Please be patient.
```

Assessment
https://www.wunderground.com/dashboard/pws/IVANDU8/table/2023-08-25/2023-08-25/monthly
take the csv file
read it with Matlab, extract the columns: Average Temperature, Average Humidity, Average Wind speed
Display the curves over the days of the month

4.8. Arrays

4.8.1. In C/C++

http://www.cplusplus.com/doc/tutorial/arrays/
Declare an integer array with known size at compilation time

```
int a[5];
```

Declare an initialized integer array with

```
int a[5] = { 5, 4, 3, 2, 1 };
```

4.8.2. In C++

Declare an integer array with unknown size at compilation time
Operator new and delete must be used
`int *b;` * means b is a pointer on an integer variable
It is the first element of the array

```
// Dynamic arrays
#include <iostream>

using namespace std;

int *b;
int N;

int main() {
    cout << "Enter a value" << endl;
    cin >> N;
    b = new int[N];
    for (int i = 0; i < N; i++) {
        b[i] = i * 2;
        cout << b[i] << endl;
    }
    return 0;
}
```

```
Enter a value
6
0
2
4
6
8
10
```

4.8.3. In C

Syntax is much complex

```
int* b = malloc(N * sizeof(int));
```

It is much simpler to deal with dynamic arrays ic C++.
There are also special classes like vector, list,… with iterators

4.8.4. In Octave

We use 1-dimension Matrices to store arrays of data

https://octave.org/doc/v5.2.0/Matrices.html#Matrices
a(2) is the second elements of the array a
It is much simpler to manipulate than in C/C++
Octave is a scientific tool dedicated to manipulate data

```
a = { 5, 4, 3, 2, 1 };
```

```
>> a
a =
{
  [1,1] =  5
  [1,2] =  4
  [1,3] =  3
  [1,4] =  2
  [1,5] =  1
}
```

```
>> a(2)
ans =
{
  [1,1] =  4
}
```

In Matlab
https://fr.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html
a = [5 4 3 2 1];

a =

   5   4   3   2   1

a(2) is the second element of the array
>> a(2)

ans =

   4

4.9. Plot data

4.9.1. In C/C++

Use a while loop to

- Generate data
- Store them in arrays or directly save them using ofstream

- Plotting is rather difficult. Need external tools. The simplest way is to export the file as CSV (Comma Separated Variables) that Excel, Octave and other tools can read easily and plot
- The format is better readable if \t (tabulation) is used instead of , (comma)

```cpp
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main() {
    double x, z,
           t = 0, dt = 0.01, tf = 1;     // 1 second
    ofstream myfile("dataplot.csv");
    if (myfile.is_open()) {
        myfile << "t \tx \tz" << endl;

        while (t <= tf) {
            x = sin(10*t);
            z = 10 * t;   // helicoidal motion
            myfile << t << " \t" << x << " \t" << z << endl;
            t += dt;
        }
        myfile.close();
    } else cout << "Unable to open file";
}
```

t	x	z
0	0	0
0.01	0.0998334	0.1
0.02	0.198669	0.2
0.03	0.29552	0.3
0.04	0.389418	0.4
0.05	0.479426	0.5
0.06	0.564642	0.6
0.07	0.644218	0.7
0.08	0.717356	0.8
...		
...		
0.96	-0.174327	9.6
0.97	-0.271761	9.7
0.98	-0.366479	9.8
0.99	-0.457536	9.9

Plot it by loading it to a text editor than copy paste to Excel

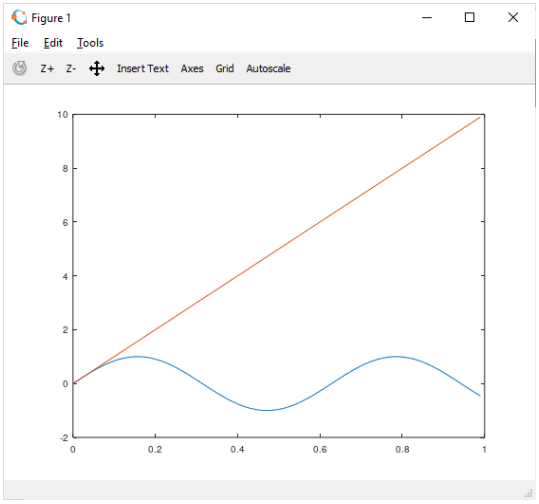
4.9.2. In Octave

Use a while loop to

- Generate data
- Store them in arrays using indice (i)

```
clear
t = 0, dt = 0.01, tf = 1; % 1 second
i=1
while (t <= tf)
        tt(i) = t;
        xt(i) = sin(10*t);
        zt(i) = 10*t;    % helicoidal motion
        t = t + dt;
        i = i + 1;
      end
figure (1)
plot(tt, xt, tt, zt)
```

Plot data on the same figure

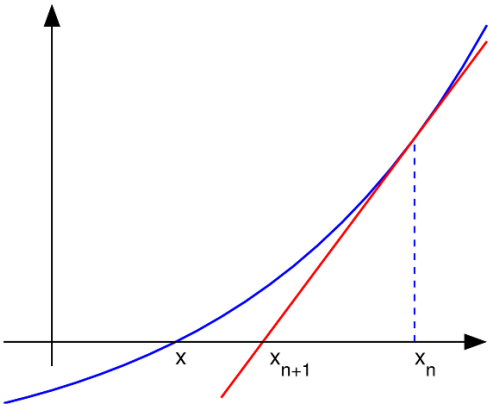


4.10. Summary

The syntax elements of C/C++ and MATLAB/Octave
How to find help
Implement the basic programming structures in C/C++ using CLion and in .m scripts Octave

5. Numerical methods and Application

Numerical methods and Application



Learn how to solve some mathematical problems using numerical methods
Learn how to code some numerical methods in C/C++ language and MATLAB / Octave

References
C++ Tutorials and help http://www.cplusplus.com/doc/tutorial/control/
C++ shell (online compiler) http://cpp.sh
C++ Tutorials https://www.w3schools.com/cpp/default.asp
C Tutorial https://www.javatpoint.com/c-programming-language-tutorial
Octave Documentation https://octave.org/doc/v5.2.0/The-for-Statement.html#The-for-Statement
Learning C++ https://www.codecademy.com/catalog/language/c-plus-plus
W. H. Press et Al., Numerical recipes in C++, Cambridge University Press, 3rd Ed. 2002, https://e-maxx.ru/bookz/files/numerical_recipes.pdf
Scheid, Francis J., Schaum's outline of theory and problems of numerical analysis, Schaum's outline series, McGraw-Hill, 1989

5.1. Fixed-point theorem

To solve $f(x) = 0$ by a simple iterative method
Put the equation on the form of $g(x) - x = 0$

$$x = g(x)$$

Example studied $x - \cos(x) = 0$

$$x = \cos(x) = g(x)$$

5.1.1. Algorithm

- Start form a point $x = x_0 \in Domain$

- At each iteration k compute $x_{k+1} = g(x_k)$
- Check if the error $e_{k+1} = |x_{k+1} - x_k|$ falls bellow a defined precision
- Check if the number of iterations is over a limit
- Otherwise loop

If the function is well conditioned, the successive x_k converge towards solution of the equation

https://en.wikipedia.org/wiki/Fixed-point_theorem#:~:text=In%20mathematics%2C%20a%20fixed%2Dpoint,be%20stated%20in%20general%20terms.

5.1.2. Program

Simple version

```
#include <iostream>
#include <cmath>

using namespace std;

double g_function(double x) {
  return cos(x);
}

int main() {
  double x, xnew, e, x0;
  x0 = 0; // starting point (can be entered by cin >>x 0)
  x = x0;
  e = 1e20;
  while (e > 1e-6) {
    xnew = g_function(x);
    e = fabs(xnew - x);
    x = xnew;
    cout << " x = " << x << " e = " << e << endl;
  }
  return 0;
}
```

Complete version

```
#include <iostream>
#include <cmath>

using namespace std;

double g_function(double x) {
    return cos(x);
}
```

```
int main() {
    double x, xnew, e, elim, x0;
    cout << "Resolution by Fixed-point theorem" << endl;
    int iter = 0, iterlim = 100;
    elim = 1e-6;
    x0 = 0; // starting point (can be entered by cin >>x 0)
    x = x0;
    e = 1e20;
    while ((e > elim) && (iter < iterlim)) {
        xnew = g_function(x);
        e = fabs(xnew - x);
        x = xnew;
        iter++;
        cout << "iter " << iter << " x = " << x
     << " e = " << e << endl;
    }
    return 0;
}
```

In the algorithm implementation, we only need to have the values of x at the actual and previous iteration x_{k+1}, x_k
We call them `xnew, x`
With a limit of iteration of 100 and a precision of 1-6, the while condition
`while ((e > elim) && (iter < iterlim))`
allows to stop the iterations if one of the conditions become false
Solution at iter 35
`x = 0.739086`
with an error of `e = 9.76787e-07`

```
Resolution by Fixed-point theorem
iter 1 x = 1 e = 1
iter 2 x = 0.540302 e = 0.459698
iter 3 x = 0.857553 e = 0.317251
iter 4 x = 0.65429 e = 0.203263
iter 5 x = 0.79348 e = 0.139191
iter 6 x = 0.701369 e = 0.0921116
iter 7 x = 0.76396 e = 0.0625909
iter 8 x = 0.722102 e = 0.0418573
iter 9 x = 0.750418 e = 0.0283153
iter 10 x = 0.731404 e = 0.0190137
iter 11 x = 0.744237 e = 0.0128333
iter 12 x = 0.735605 e = 0.00863261
iter 13 x = 0.741425 e = 0.00582035
iter 14 x = 0.737507 e = 0.0039182
iter 15 x = 0.740147 e = 0.00264045
iter 16 x = 0.738369 e = 0.00177813
```

```
iter 17 x = 0.739567 e = 0.001198
iter 18 x = 0.73876 e = 0.000806882
iter 19 x = 0.739304 e = 0.000543573
iter 20 x = 0.738938 e = 0.000366136
iter 21 x = 0.739184 e = 0.000246643
iter 22 x = 0.739018 e = 0.000166137
iter 23 x = 0.73913 e = 0.000111914
iter 24 x = 0.739055 e = 7.53858e-05
iter 25 x = 0.739106 e = 5.07812e-05
iter 26 x = 0.739071 e = 3.42066e-05
iter 27 x = 0.739094 e = 2.30421e-05
iter 28 x = 0.739079 e = 1.55214e-05
iter 29 x = 0.739089 e = 1.04554e-05
iter 30 x = 0.739082 e = 7.04288e-06
iter 31 x = 0.739087 e = 4.74417e-06
iter 32 x = 0.739084 e = 3.19573e-06
iter 33 x = 0.739086 e = 2.15268e-06
iter 34 x = 0.739085 e = 1.45007e-06
iter 35 x = 0.739086 e = 9.76787e-07
```

5.2. Newton-Raphson

To solve $f(x) = 0$ in 1-dimension
Can be used in N-dimension
We need to compute the derivative of $f(x)$
Or Jacobian in N-dimension, see Unit 5, Load flow application

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

$f(x+h) - f(x) = hf'(x)$
$f(x+h) = 0$ the solution we need to find

$$h = -\frac{f(x)}{f'(x)}$$

h is the step by which the x will change into the next x
We need a starting point not far from the solution

https://en.wikipedia.org/wiki/Newton%27s_method#/media/File:NewtonIteration_Ani.gif

https://en.wikipedia.org/wiki/Newton%27s_method

To solve $f(x) = 0$ by the Newton-Raphson iterative method
We need to compute the derivative of $f(x)$

5.2.1. Algorithm

- Start form a point $x = x_0$ not far from the solution
- At each iteration k compute the derivative $f'(x_k)$
- and the step $h = -\frac{f(x_k)}{f'(x_k)}$ by which the x_k will change into x_{k+1}
- Check if the error $e_{k+1} = |x_{k+1} - x_k|$ falls bellow a defined precision

- Check if the number of iterations is over a limit
- Otherwise loop

The difference in regard to the previous algorithm are given in a black background

5.2.2. Program

```
double f_function(double x) {
    return (x-cos(x));
}

double df_function(double x) {
    return (1+sin(x));
}


double df_approxim_function(double x) {
    return ( (function(x+1e-8)-function(x))/1e-8);
}

int main() {
    double x, xnew, e, elim, x0, df;
    cout << "Resolution by N-R" << endl;
    int iter = 0, iterlim = 100;
    elim = 1e-6;
    x0 = 0;
    x = x0;
    e = 1e20;
    while ((e > elim) && (iter < iterlim)) {
        df = df_function(x);
        if (df !=0) {
            xnew = x -f_function(x)/df;
        }
        else{
            cout << "df = 0, end of iterations" << endl;
            break;
        }

        e = fabs(xnew - x);
        x = xnew;
        iter++;
        cout << "iter " << iter << " x = " << x
      << " e = " << e << endl;
    }
    return 0;
}
```

If the function is well conditioned and the initial point is not too far from the solution, the successive x_k converge very fast (quadratic convergence) towards solution of the equation

The same function needs 5 iterations, with Newton-Raphson, to reach the same precision as the Fixed-point theorem algorithm, for which it took 35 iterations
It can be impossible to compute the derivative in some conditions
In next unit (PWM harmonics cancellation) or when dealing with (Load flow), we will see how to approximate the derivative functions to compute the Jacobian matrix of N-R method

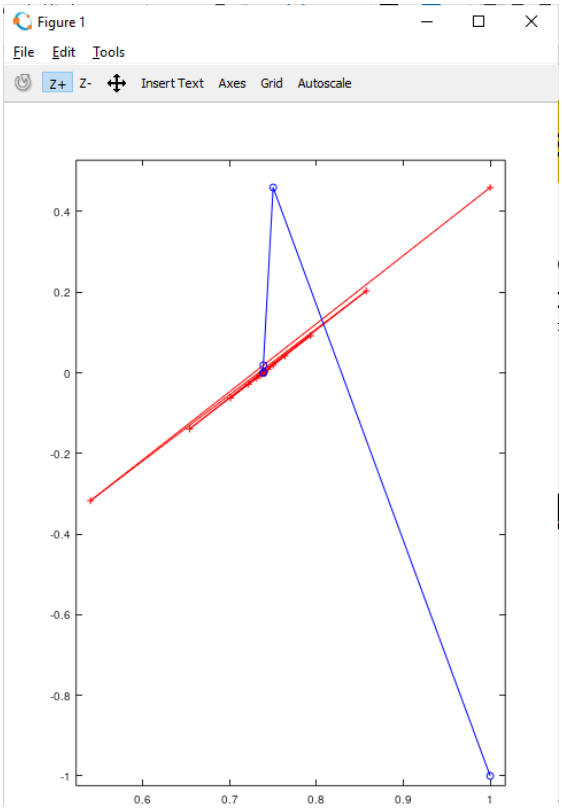
```
Resolution by N-R
iter 1 x = 1 e = 1
iter 2 x = 0.750364 e = 0.249636
iter 3 x = 0.739113 e = 0.011251
iter 4 x = 0.739085 e = 2.77575e-05
iter 5 x = 0.739085 e = 1.70123e-10
```

5.2.3. Assessment: Assignment

Implement the algorithms of Fixed-point theorem and Newton-Raphson to solve the same equation $x - \cos(x) = 0$
in Octave using one common .m script
Record the values x_k and $f(x_k)$ for both algorithm in arrays as explained in previous Unit
Plot, on the same figure, the values of the arrays like

```
plot(pt_xt, pt_fxt,'d+-r')
hold;
plot(nr_xt, nr_fxt,'o-b')
```

to obtain the figure below



Notice the huge difference of number of iterations to reach the same precision for the 2 algorithms
What do you conclude?

5.3. Gauss-Seidel

It is an iterative method used to solve a system of linear equations

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \\ a_{i1} & a_{i2} & a_{ij} & a_{in} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_i \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ b_i \end{bmatrix}$$

We need a starting vector

5.3.1. Algorithm

At each iteration $[k]$, we compute the elements of the new vector $[k+1]$ with the previously computed elements:

- From $j = 1$ to $i - 1$, using the already computed elements of the new vector $x_j^{[k+1]}$
- From $j = i + 1$ to n, using the computed elements of the previous vector $x_j^{[k]}$

$x_i^{[k+1]} = \frac{b_i}{a_{ii}} - \sum_{j=0}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{[k+1]} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{[k]}$

There are conditions on the A matrix to be symmetric positive-definite

$$i^{th} line \sum_{j=1}^{n} a_{ij} x_j = b_i$$

$$a_{ii} x_i + \sum_{j=1,\, j \neq i}^{n} a_{ij} x_j = b_i$$

$$x_i = \frac{b_i}{a_{ii}} - \sum_{j=1,\, j \neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j$$

From one iteration to the other

$$x_i^{[k+1]} = \frac{b_i}{a_{ii}} - \sum_{j=1,\, j \neq i}^{n} \frac{a_{ij}}{a_{ii}} x_j^{[k]}$$

Variant

$x_i^{[k+1]} = \frac{b_i}{a_{ii}} - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{[k+1]} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{[k]}$

https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method
Scheid, Francis J., Schaum's outline of theory and problems of numerical analysis, Schaum's outline series, McGraw-Hill, 1989

Example of dimension n=4 in Octave
Notice the nested loops

$$x_i^{[k+1]} = \frac{b_i}{a_{ii}} - \sum_{j=0}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{[k+1]} - \sum_{j=i+1}^{n} \frac{a_{ij}}{a_{ii}} x_j^{[k]}$$

5.3.2. Program

For Octave version

```
% Gauss-Seidel example
clc,clear all,close all
n = 4;
A = [3 2 1 -1; 5 7 -1 0; -2 3 3 1; 0 2 1 5];
B = [1 2 3 4];
X = [1 1 1 1];
Xnew = [0 0 0 0];
iterlim = 100;
elim = 1e-6;
e = 1e20;
iter = 0
while ((e > elim) && (iter < iterlim))
    e=0;
    for  i = 1:n
        Xnew(i) = B(i);
        for  j = 1:i-1
            Xnew(i) = Xnew(i) -A(i,j)*Xnew(j);
            endfor
        for  j = i+1:n
            Xnew(i) = Xnew(i) -A(i,j)*X(j);
            endfor
        Xnew(i) = Xnew(i)/A(i,i);
        e = e + abs(Xnew(i)-X(i));
        endfor
    X = Xnew;
    iter++;
    printf ("iter %d  e = %g\n", iter, e);
    disp(X);
    printf ("\n");
endwhile
```

For Matlab version

```
% Gauss-Seidel example
clc,clear all,close all
n = 4;
A = [3 2 1 -1; 5 7 -1 0; -2 3 3 1; 0 2 1 5];
B = [1 2 3 4];
X = [1 1 1 1];
Xnew = [0 0 0 0];
iterlim = 100;
elim = 1e-6;
e = 1e20;
iter = 0
```

```
while ((e > elim) && (iter < iterlim))
 e=0;
 for i = 1:n
     sum = B(i);
     for j = 1:i-1
         sum = sum -A(i,j)*Xnew(j);
     end
     for j = i+1:n
         sum = sum -A(i,j)*X(j);
     end
     Xnew(i) = sum/A(i,i);
     e = e + abs(Xnew(i)-X(i));
 end
 X = Xnew;
 iter = iter +1;
 fprintf ("iter %d e = %g\n", iter, e);
 disp(X);
 fprintf ("\n");
end
```

$Xnew_1 = (B_1 - A_{12}X_2 - A_{13}X_3 - A_{14}X_4)/A_{11}$
$Xnew_2 = (B_2 - A_{21}Xnew_1 - A_{23}X_3 - A_{24}X_4)/A_{22}$
$Xnew_3 = (B_3 - A_{31}Xnew_1 - A_{32}Xnew_2 - A_{34}X_4)/A_{33}$
$Xnew_4 = (B_4 - A_{41}Xnew_1 - A_{42}Xnew_2 - A_{43}Xnew_3)/A_{44}$

Beware of the conditioning of the matrix
The system may not converge for ill matrix or not symmetric positive-definite
After 12 iterations, the absolute error falls below the required precision

```
iter 1  e = 3.31111
-0.33333   0.66667  -0.22222   0.57778

iter 2  e = 2.0146
0.15556   0.14286   0.76825   0.58921

iter 3  e = 0.289088
0.17841   0.26803   0.65451   0.56189

iter 4  e = 0.128426
0.12377   0.29081   0.60441   0.56279

iter 5  e = 0.0211419
0.12559   0.28235   0.61378   0.56430

iter 6  e = 0.006303
0.12861   0.28153   0.61610   0.56417

iter 7  e = 0.0015525
0.12833   0.28206   0.61544   0.56409

iter 8  e = 0.000280412
```

```
0.12817   0.28208   0.61534   0.56410

iter 9  e = 0.000103324
0.12820   0.28205   0.61538   0.56410

iter 10  e = 1.22329e-05
0.12821   0.28205   0.61539   0.56410

iter 11  e = 6.40168e-06
0.12821   0.28205   0.61538   0.56410

iter 12  e = 5.50213e-07
0.12821   0.28205   0.61538   0.56410
```

Try with the classical (not the variant) method and see the increase of iterations.

With Octave we can verify the exact solution by inversing the matrix

```
>> d = eig(A)
d =

1.8382 + 1.a8945i
1.8382 - 1.8945i
8.1205 + 0.0000i
6.2030 + 0.0000i

>> inv(A)*B'
ans =

0.12821
0.28205
0.61538
0.56410
```

5.3.3. Assessment: Assignment

Implement the algorithm of Gauss-Seidel to solve the same previous problem, but this time in C++, using CLion

Declare the arrays and initialize them using

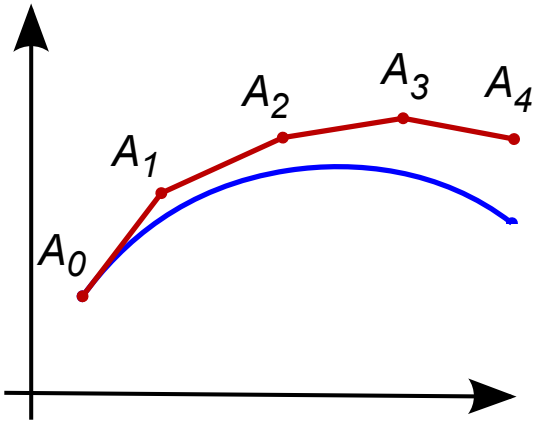
```
const int n = 4;
int i, j;
double A[n][n] = {{3,2,1,-1}, {5,7,-1,0}, {-2,3,3,1}, {0,2,1,5}};
double B[n] = {1,2,3,4};
double X[n], Xnew[n], e, elim;
```

The result should be

```
Resolution by G-S
iter 1 e = 3.31111
X = -0.333333 0.666667 -0.222222 0.577778
iter 2 e = 2.0146
X = 0.155556 0.142857 0.768254 0.589206
iter 3 e = 0.289088
X = 0.178413 0.268027 0.654512 0.561887
iter 4 e = 0.128426
X = 0.123773 0.290807 0.604413 0.562795
iter 5 e = 0.0211419
X = 0.125589 0.282352 0.613776 0.564304
iter 6 e = 0.006303
X = 0.128608 0.281534 0.616103 0.564166
iter 7 e = 0.0015525
X = 0.128332 0.282064 0.615435 0.564087
iter 8 e = 0.000280412
X = 0.128175 0.28208 0.615341 0.5641
iter 9 e = 0.000103324
X = 0.1282 0.282049 0.615384 0.564104
iter 10 e = 1.22329e-05
X = 0.128207 0.28205 0.615387 0.564103
iter 11 e = 6.40168e-06
X = 0.128205 0.282052 0.615384 0.564103
iter 12 e = 5.50213e-07
X = 0.128205 0.282051 0.615384 0.564103
```

Did you pay attention and verify calculus conditions (divide by 0)?

5.4. Numerical integration



The unknown curve is in blue and its polygonal approximation using Euler method is in red

In simulation of systems, we have differential equation of state variables.
Step by step time integration of the differential equations give the behavior of the system vs time
We will use Euler integration method, to simulate the DC motor and its current control loop
We will also implement modified Euler, trapezoidal and Runge Kutta 4 integration methods for solving transient stability studies of a power grid
As of all integration methods, they are more accurate if the step size h is smaller

https://en.wikipedia.org/wiki/Euler_method
https://en.wikipedia.org/wiki/Predictor%E2%80%93corrector_method
Scheid, Francis J., Schaum's outline of theory and problems of numerical analysis, Schaum's outline series, McGraw-Hill, 1989

5.4.1. Euler Algorithm

Polynomial approximation
Knowing an initial point y_0 at t_0: $y(t_0) = y_0$
and the derivative of the function $y'(t) = f\big(t,\, y(t)\big)$
We choose an integration step h enough small
The basic Euler method gives the next point

$$y_1 = y_0 + hf(t_0,\, y_0)$$

When generalized to a step n

$$y_{n+1} = y_n + hf(t_n,\, y_n)$$

We notice that a small error at a step will be cumulative
A modified Euler method use the predictor-corrector

$y_{n+1}^{p} = y_n + hf(t_n,\, y_n)$ predictor

$y_{n+1}^{c} = y_n + \frac{h}{2}\Big(f(t_n, y_n) + f\big(t_{n+1}, y_{n+1}^{p}\big)\Big)$ corrector

Example of the Euler Predictor-Corrector integration algorithm used in Transient Stability program

5.4.2. Program

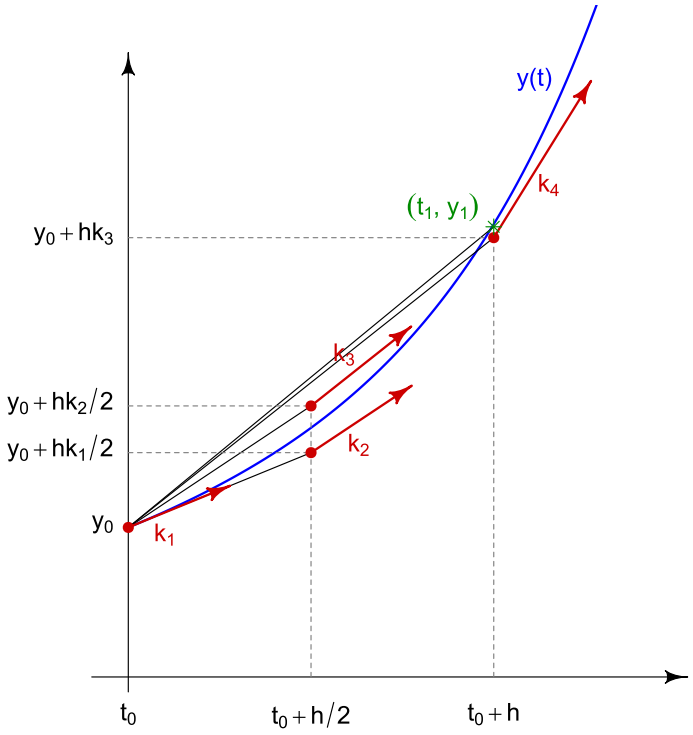
```
void TEuler::EulerPredictor()
{
   for ( int i=0; i<=Npv; i++)
      {
      dXvp[i]=Xv[Npv1+i]-Ws;           //   |-- Damper
      dXvp[Npv1+i]=Ws/2/H[i]*( Pm[i]-Pg[i]-D[i]*dXvp[i]);
      Xvp[i]=Xv[i]+dXvp[i]*dt;
      Xvp[Npv1+i]=Xv[Npv1+i]+dXvp[Npv1+i]*dt;
      }
}
```

```
void TEuler::ModifiedEulerCorrector()
{
complex<double> Ig;
int i;
   for ( i=0; i<=Npv; i++)
      {
          dXvc[i]=Xvp[Npv1+i]-Ws;          //   |--Damper
          dXvc[Npv1+i]=Ws/2/H[i]*( Pm[i]-Pg[i]-D[i]*dXvc[i]);  // the new
Pg
          Xv[i]+=( dXvp[i]+dXvc[i] )*dt/2;
          Xv[Npv1+i]+=( dXvp[Npv1+i]+dXvc[Npv1+i] )*dt/2;
          Eg[i]=polar( abs( Eg[i]), Xv[i] );
      }
   LFGauss();
CalcPg();
}
```

dXvp is the derivative for the predictor
Xvp is the result of the first integration (predictor)
dXvc is the derivative using the predictor result
Xvc (which is the new Xv) is the corrector result of the integration
In the example the Xv vector is split into 2 parts, one to integrate the speed, the second one to integrate the acceleration of the generators

5.4.3. Runge Kutta 4 Algorithm

The Runge Kutta 4 (RK4) method is very popular and efficient



$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$k_1 = f(t_n,\ y_n)$$
$$k_2 = f\left(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_1\right)$$
$$k_3 = f\left(t_n + \frac{h}{2},\ y_n + \frac{h}{2}k_2\right)$$
$$k_4 = f(t_n + h,\ y_n + hk_3)$$

Now that computers and CPU are very fast in regard to 30 years ago, this method becomes the basic of numerical integration
The implementation of the algorithm under octave is called ode45
https://octave.org/doc/v4.2.0/Matlab_002dcompatible-solvers.html

https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods#The_Runge%E2%80%93Kutta_method
https://en.wikipedia.org/wiki/List_of_Runge%E2%80%93Kutta_methods
J. Arrillaga, C. P. Arnold , Computer Analysis of Power Systems, John Wiley & Sons, 1990,
http://eword.yolasite.com/resources/51621752-Computer-Analysis-of-Power-Systems.pdf

Presenting some iterative methods to solve $f(x) = 0$

The iterative solving of AX=B, can also be used if the system is nonlinear

The numerical integration principle and usage to simulate systems modelized by differential equations

Implementation code in C++ and Octave

6. Application of numerical methods

Application of numerical methods

Apply numerical methods using C/C++ / Octave to solve actual problems
Simulation of a DC Motor, Current control of a DC Motor, PWM harmonics cancellation, determine sun position, Maximum Power Point Tracking of a PV panel

6.1. Introduction

In order to learn the programming languages and the numerical methods, we will present some examples with programming solutions in C++ and/or Octave

For a better understanding, the simulations must be done by the student
Assigned tasks of programming in C++ or Octave and changing simulation conditions must be done
Report of the results must be sent in pdf format

6.2. Simulation of a DC Motor

DC Motor model
The model is represented by 2 differential equations, involving the variation of current I and speed Ω

Electrical equation $U = RI + L\frac{dI}{dt} + k\Omega$

Mechanical equation $C_e - C_r = J\frac{d\Omega}{dt}$

with $C_e = kI$ the electromagnetic torque and $C_r = a\Omega + C_{r0}$ the friction and load torque

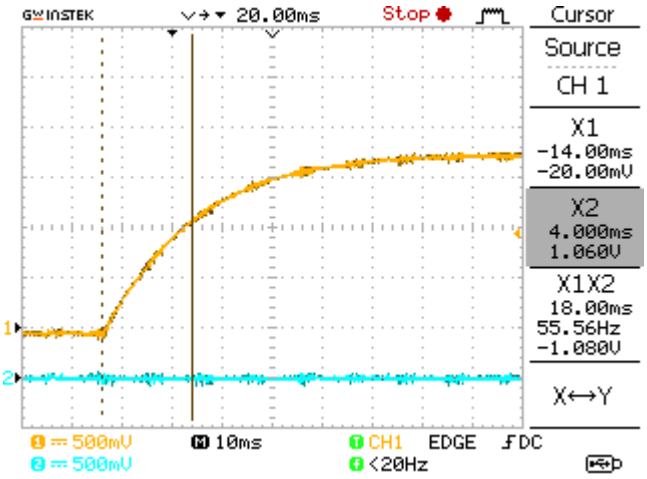
We simulate the DC motor when we apply a sudden nominal voltage of $U = 200V$

Numerical integration of differential equations

$\frac{dI}{dt} = \frac{1}{L}(U - RI - k\Omega)$

$$\frac{d\Omega}{dt} = \frac{1}{J}(C_e - C_r)$$

First part, we simulate only the electrical model, considering there is no motor rotation to see if the program is correct because it is easy to see that the current response is a first order (exponential) like in the oscilloscope capture below.



In Octave, electrical model

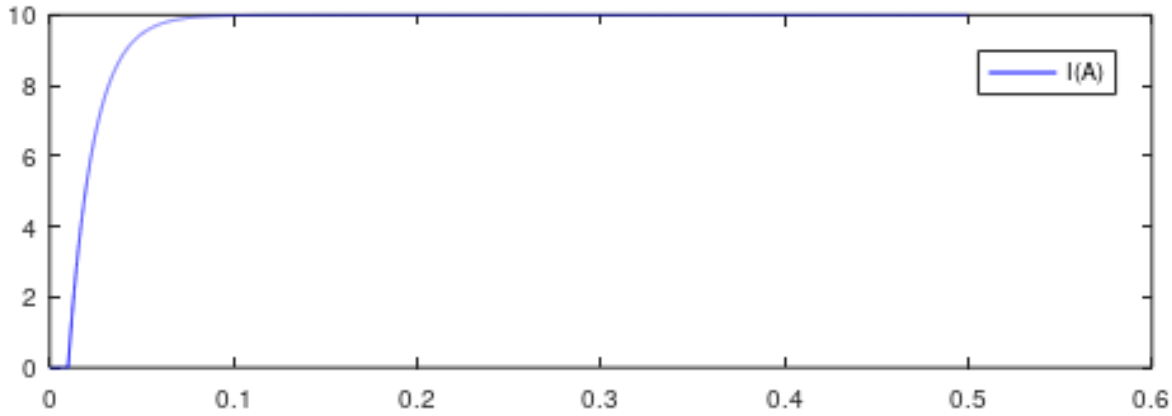
The heart of the simulation is the while loop

While the time didn't reach the final time tf, the differential values dI is computed `dI = (u-R*I-k*w)/L`

and the integration of the current I is done using the simple rectangle method `I = I + dI*dt`

Notice the way we fill the arrays to hold all the values of the simulation variables in order to plot them vs time

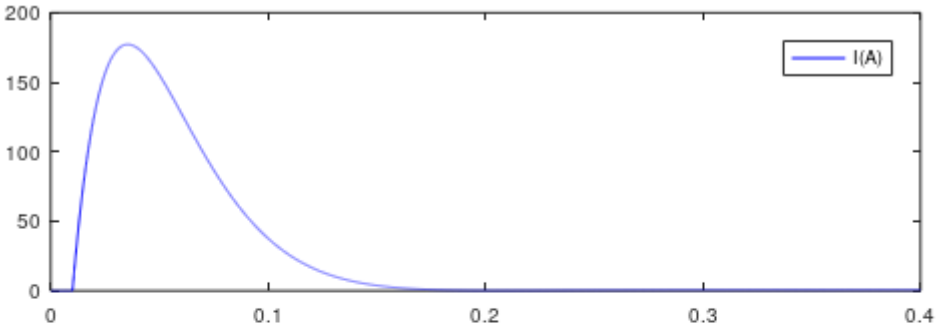
The result of the current variation is as expected

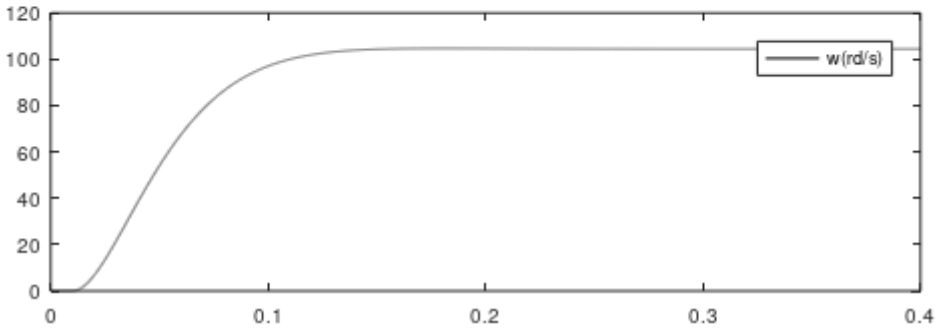


```
% Simulation of DCM at standstill, voltage Step
clc,clear all,close all
%% initialisation
tf=0.5; % final simulation time
dt=0.0001; % time step
tk=1; % array indice
Ntf=floor(tf/dt);
%% systeme variables
R= 0.8; L = 0.011; k = 1.91; J =0.2; a= 1E-2;
u=0; I=0; w=0; t=0;
u0 = 8; Cr0 = 0;
% initialization
tt=zeros(1, Ntf); It=zeros(1, Ntf); ut=zeros(1, Ntf);
wt=zeros(1, Ntf); Cet=zeros(1, Ntf);
%% boucle du temps de simul
while (t<tf)
  if t>=0.01 % 10 ms power the DCM by a voltage step
          u=u0;
        end
  dI = (u-R*I-k*w)/L;
  Ce = k*I; dw = 0; %(Ce-a*w-Cr0)/J;
  % integration rectangle method
  I = I + dI*dt;
  w = w + dw*dt;
  %% array filling
  tt(tk)=t; It(tk)=I; ut(tk)=u;
  wt(tk)=w; Cet(tk)=Ce;
  t=t+dt; tk=tk+1; % array indice
end % while end
%% plot results
figure(1)
subplot(2,1,2),plot(tt, wt,'k');
legend('w(rd/s)');
subplot(2,1,1),plot(tt,It,'b');
legend('I(A)');
```

In Octave, full model

We add the mechanical model.


200
150
100
50
0
0
0.1
0.2
0.3
0.4
I(A)


120
100
80
60
40
20
0
0
0.1
0.2
0.3
0.4
w(rd/s)

```
% Simulation of DCM at standstill, voltage Step
clc,clear all,close all
%% initialisation
tf=0.4; % final simulation time
dt=0.0001; % time step
tk=1; % array indice
Ntf=floor(tf/dt);
%% systeme variables
R= 0.8; L = 0.011; k = 1.91; J =0.2; a= 1E-2;
u=0; I=0; w=0; t=0;
u0 = 200; Cr0 = 0;
% initialization
tt=zeros(1, Ntf); It=zeros(1, Ntf); ut=zeros(1, Ntf);
wt=zeros(1, Ntf); Cet=zeros(1, Ntf);
%% boucle du temps de simul
while (t<tf)
  if t>=0.01 % 10 ms power the DCM by a voltage step
          u=u0;
        end
  dI = (u-R*I-k*w)/L;
  Ce = k*I; dw = (Ce-a*w-Cr0)/J;
  % integration rectangle method
  I = I + dI*dt;
  w = w + dw*dt;
  %% array filling
  tt(tk)=t; It(tk)=I; ut(tk)=u;
  wt(tk)=w; Cet(tk)=Ce;
  t=t+dt; tk=tk+1; % array indice
end % while end
%% plot results
figure(1)
subplot(2,1,2),plot(tt, wt,'k');
legend('w(rd/s)');
subplot(2,1,1),plot(tt,It,'b');
legend('I(A)');
```

- Change the voltage step to 200V
- Change the voltage step to a ramp finishing to 200V at 0.5s. Change t_f to 2s.
- Apply a load Cr0 =10Nm at t=1s

6.3. Assignment, Simulation of a DC Motor

In C++

Assignment

Using the C++ program that save data in a file (see below) and the previous Octave program that simulate the DC motor on a direct start, write the equivalent program in C++

It has to simulate the start up of the DC motor in the same conditions as the Octave program

The time, the current and the speed must be saved at each (integration) step to a CSV file
This file will be read by Excel or Octave
Compare the results with the one given by the Octave program

```cpp
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

int main() {
    double x, z,
           t = 0, dt = 0.01, tf = 1;     // 1 second
    ofstream myfile("dataplot.csv");
    if (myfile.is_open()) {
        myfile << "t \tx \tz" << endl;

        while (t <= tf) {
            x = sin(10*t);
            z = 10 * t;   // helicoidal motion
            myfile << t << " \t" << x << " \t" << z << endl;
            t += dt;
        }
        myfile.close();
    } else cout << "Unable to open file";
}
```

6.4. Current control of a DC Motor

In octave DC Motor model with transfer functions

To control the current of the DC motor, one common assumption is to neglect the dynamic variation of the speed in respect to the fast variation of current I

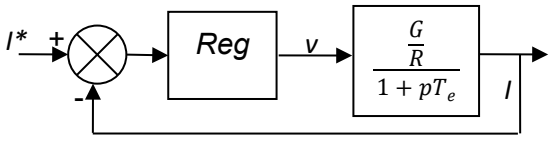
Therefore, the current transfer function of the current in open loop is

$$FTBO = \frac{I(p)}{v(p)} = \frac{\frac{G}{R}}{1 + pT_e}$$

where G is the Amplification factor of the DC chopper

We need to add a PI current controller

We must compute the controller parameters in discrete form, then add the controller to the simulation program



DC Motor model with transfer functions

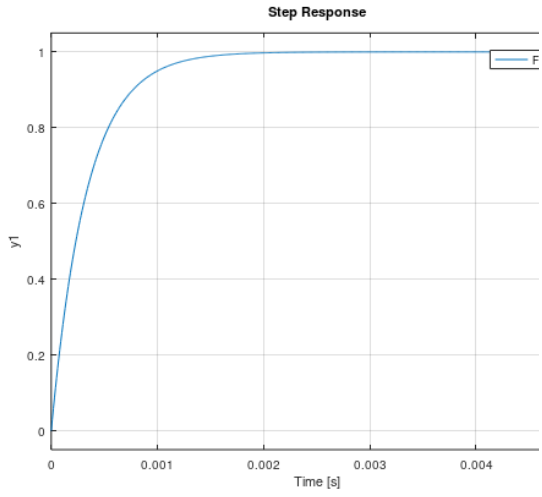
The continuous controller parameters are

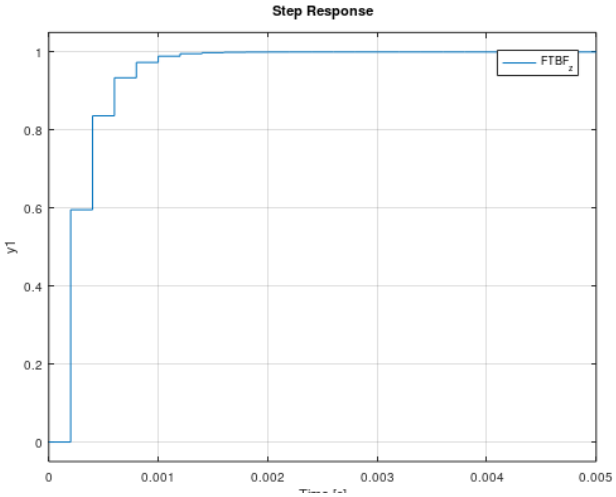
Kp = 1.65, Ki = 120

For a `Ts=2e-4` the discrete ones are

Kpd = Kp = 1.65, Kid = Ki*Te = 0.024

Here is the answer of the machine to an I_{ref} step from 0 to 1 Amps.



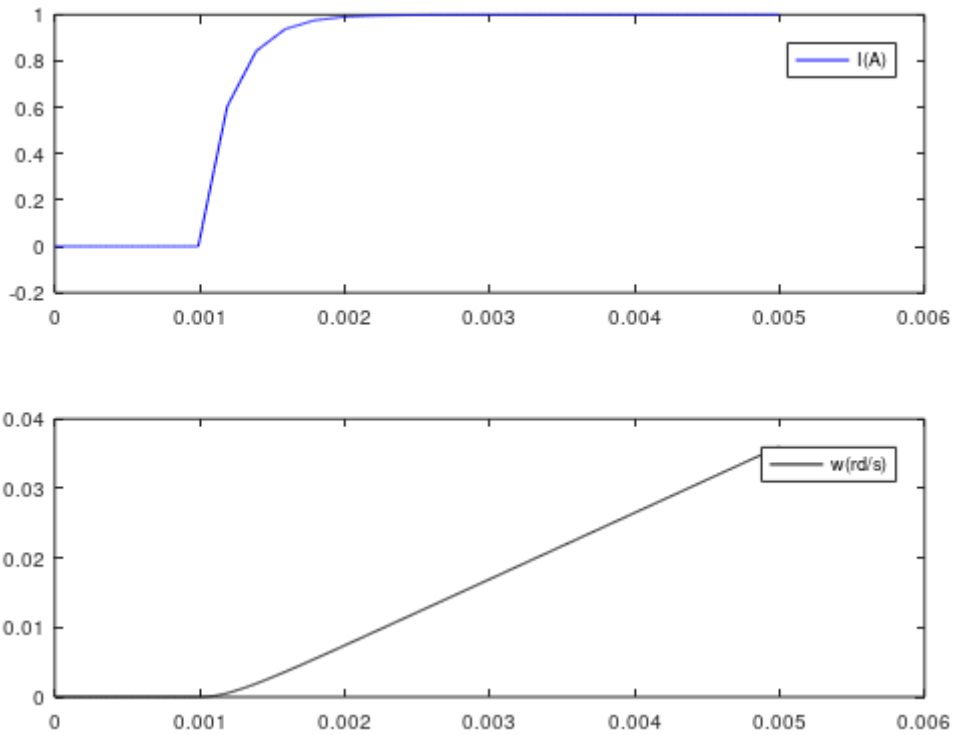



```
%% Current control of DC Motor
clc,clear all,close all
%pkg load control
%% Parameters
R= 0.8; L = 0.011; k = 1.91; J =0.02; a= 1E-2;
Te=L/R;
Tem=J*R/k/k;
G=20;
Ts=0.2e-3;
%%  PI control of I
tr=0.001; % step time at 1 ms
Ti=Te;  % perfect pole compensation
%Ti=0.8*Te;  % bad pole compensation
Kp=3*R*Ti/G/tr;
Ki=Kp/Ti;

s=tf('s');
% TF DCM
MCC=G/R/(1+s*Te)
MCC_z=c2d(MCC, Ts, 'zoh') % discrete format

%  TF PI
PI=Kp+Ki/s
PI_z=c2d(PI, Ts, 'zoh')
FTBO=series(MCC, PI)
FTBO_z=series(MCC_z, PI_z)

% Closed Loop transfer function
FTBF=feedback(FTBO,1);
FTBF_z=feedback(FTBO_z,1);

% step response - continious state
figure(1);
t=[0: 1e-5: 0.005];
step(FTBF,t);
[y x]=step(FTBF,t);
sprintf('Over shoot=%5.2f %%', 100*(max(y)-1))

% step response - discrete state
figure(2);
```

```
t=[0: Ts: 0.005];
step(FTBF_z,t);
[y x]=step(FTBF_z,t);
sprintf('Over shoot=%5.2f %%', 100*(max(y)-1))
```

DC Motor simulation with current control

Here is the answer of the machine to a I_{ref} step from 0 to 1 Amps.

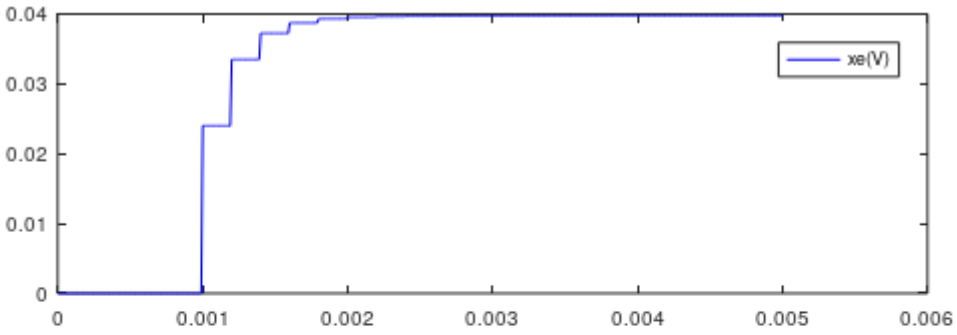


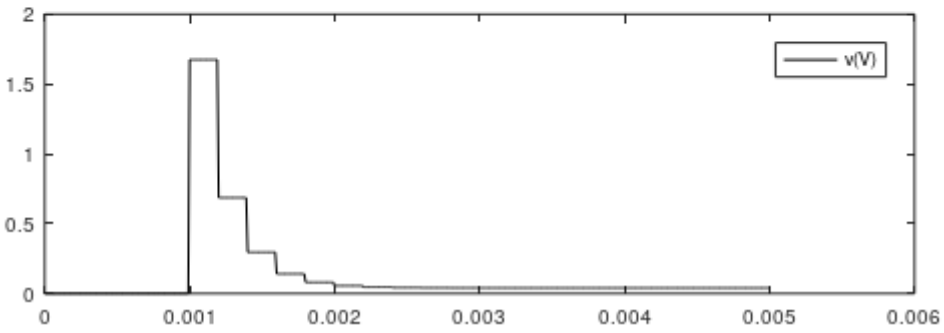

```
%% Current control of a DC Motor
clc,clear all,close all
%% Parameters
R= 0.8; L = 0.011; k = 1.91; J =0.2; a= 1E-2; % frottement visqueux
Te=L/R;
Tem=J*R/k/k;
G=20;
Ts=0.2e-3;  % 200 us
%%  Current (I) PI Controller
tr=0.001; % temps de tep à 1 ms
Ti=Te;
Kp=3*R*Ti/G/tr;
Ki=Kp/Ti;
Ki_d=Ki*Ts;
%% initialization
tf=0.005;      % final simulation time
dt=0.01e-3;    % time step
tk=1;          % array indice
Ntf=floor(tf/dt);
%% system variables
u=0; I=0; w=0; Iref=0;
xe=0; Flagtm=0;
Vlim=10;  % maximum 200 V=10*G
```

```
Iref0 = 1; % step value
Cr0 = 0;
% initialization of arrays and
tt=zeros(1, Ntf);    t=0;
It=zeros(1, Ntf);    I=0.0; % Current
ut=zeros(1, Ntf);
wt=zeros(1, Ntf);
vt=zeros(1, Ntf);
xet=zeros(1, Ntf);
Cet=zeros(1, Ntf);
```

DC Motor simulation with current control

Integral part (xe) and voltage at the output of the PI controller






```
%%DC Motor simulation with current control
%Integral part (xe) and voltage at the output of the PI controller
while (t<tf) %% while loop
  if t>=0.001   % at 1 ms, ther is z stre
    Iref=Iref0;
  end
%% MODULULO to have an execution at a different sampling period
  tm100=t-(Ts*floor(t/Ts));
  if(tm100>dt) Flagtm=0; end;
  if((tm100<=dt) && (Flagtm==0))
    Flagtm=1;
    % Controller
    e=Iref-I;
    xe=xe+Ki_d*e;
    v=Kp*e+xe;
    end;
  u=G*v;
```

```
  dI = (u-R*I)/L;
  %dI = (u-R*I-k*w)/L;
  Ce = k*I;
  dw = (Ce-a*w-Cr0)/J;
  % integration
  I = I + dI*dt;
  w = w + dw*dt;
  tt(tk)=t; %% array filling
  It(tk)=I;
  ut(tk)=u;
  wt(tk)=w;
  Cet(tk)=Ce;
  vt(tk)=v;
  xet(tk)=xe;
  t=t+dt;
  tk=tk+1; % array indice
end % while end
%% plot results
figure(1)
subplot(2,1,2),plot(tt, wt,'k');
legend('w(rd/s)');
subplot(2,1,1),plot(tt,It,'b');
legend('I(A)');

figure(2)
subplot(2,1,2),plot(tt, vt,'k');
legend('v(V)');
subplot(2,1,1),plot(tt,xet,'b');
legend('xe(V)');
```

6.5. Assignment, Current control of a DC Motor

DC Motor simulation with current control, Full model

Change the program to take into account the full model including the mechanical differential equation

- What happens if we keep the current I_{ref} step at 8A for a long time (tf=2s)?
- What happens if we limit the output of the current controller to 10 with

```
if (v>Vlim) v = Vlim; end;
if (v<-Vlim) v = -Vlim; end;
```

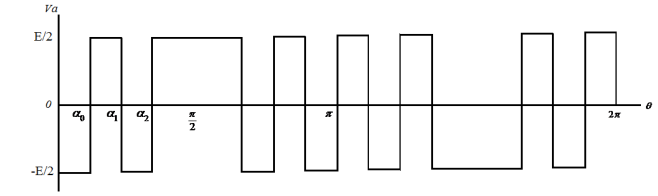
6.6. PWM harmonics cancellation

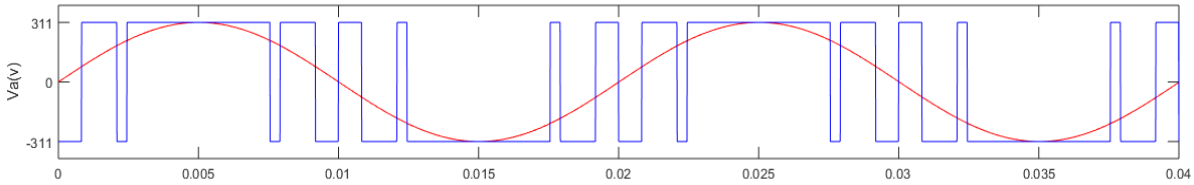
Subject

Make a program that find the angles (alphas) of a PWM pattern with elimination of harmonics.

Nh: number of harmonics to be imposed (one fundamental and Nh-1 harmonics to be eliminated)

The program has to be flexible, that is to say that it must work for any chosen Nh





Functions

`harm(i)` computes the value of the harmonic number given an index I

The harmonics are $h = 6i \pm 1$

`Bn( Nh, E, h, Xhere)` computes the amplitude (max) of the harmonic h. B_h

`fonctn( Nh, E, Vfmax, i, Xhere)` computes the i^{th} constraint

For i=0, it is the constraint First harmonic amplitude minus the reference Fundamental amplitude value

$$f_1 = B_1 - V_{fmax}$$

For i>0, constraint is the harmonic amplitude itself

$$f_i = B_i$$

These constrains must be forced to zero by the algorithm by finding the corresponding angle configuration (alphas)

`jacob( Nh, E, Vfmax, Xn)` computes the Jacobian (`Jac`) matrix given a vector Xn. We use an approach derivative, by computing a slightly different vector with a step of 1e-8

harm.m

```
function harmo=harm(i)
%calculation of the number of harmonic n ° i
if (i==0) harmo=1;
    elseif (rem(i,2)==1) harmo=i*3+2;
    else harmo=(i/2)*6+1;
    end
%disp( sprintf('i=%d harm=%d', i,harmo));
```

Bn.m

```
function somme=bn( Nh, E, h, Xhere)
%calculation of harmonic n ° h
```

```
somme=-0.5;
for i=0 : Nh-1
    if ( rem(i,2)==1 ) somme=somme-cos( h*Xhere(i+1));
        else           somme=somme+cos( h*Xhere(i+1));
        end;
    end;
somme=somme*4*E/pi/h;
%disp( sprintf('h=%d Bn=%f', h,somme));
```

fonctn.m

```
function f=fonctn( Nh, E, Vfmax, i, Xhere)
%calculation of harmonic n ° h - value to be imposed
if (i==0) f=bn(Nh, E,1, Xhere)-Vfmax;
    else  f=bn(Nh, E, harm(i), Xhere);
    end;
%disp( sprintf('i=%d fonctn=%f', i,f));
```

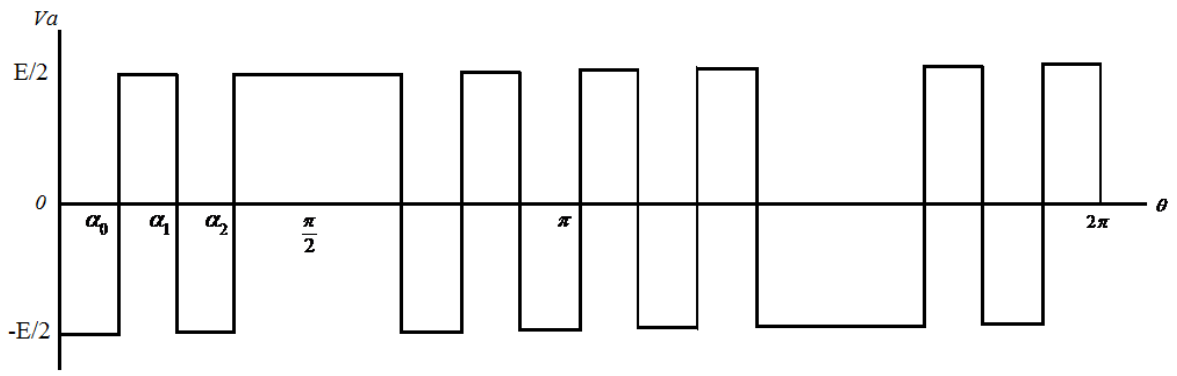
jacob.m

```
function Jac=jacob( Nh, E, Vfmax, Xn)
% calculation of Jacobian
deltaX=1e-8;
for j=0 : Nh-1
    Xp=Xn;   % copy
    Xp(j+1)=Xp(j+1)+deltaX;
    for i=0 : Nh-1
        Jac(i+1,j+1)=( fonctn( Nh, E, Vfmax, i, Xp) - fonctn( Nh, E,
Vfmax,i, Xn) )/deltaX;
        end;
    end;
%disp( Jac);
```

mliharm.m

```
%initialization
clc,clear all,close all
E=622; Es2=E/2;
Vfmax=E/4
Nh=3; % Nh harm to impose
X(Nh,1)=0;
% initialize Xn1
for i=0 :Nh-1
   Xn1(i+1,1)=(i+1)*pi/3.0/(Nh+1);
   end
iter=0;
FlagOut=0;
err=1e-10;
errMax=100;
while ( (errMax>err) & (FlagOut<50))
  Xn=Xn1;
   for i=0 :Nh-1
```

```
      fun(i+1)=fonctn( Nh, E,Vfmax, i, Xn);
   end
   errMax=max(abs(fun));
   Jac=jacob( Nh, E, Vfmax, Xn);
   Xn1=Xn-inv(Jac)*fun';
% verifOut
for i=0 :Nh-1
   if (Xn1(i+1)>pi/2) Xn1(i+1)=pi/2;
      FlagOut=FlagOut+1;
      end
   if (Xn1(i+1)<0)          Xn1(i+1)=0;
      FlagOut=FlagOut+1;
      end
   end
   iter=iter+1;
   disp( '------------------------');
   disp( sprintf('iter=%d           funct=', iter));
   disp( sprintf('%f      ', fun));
   disp( 'Xn1=');
   disp( sprintf('%f   ', Xn1));
   disp( sprintf('Erreur Max=%f', errMax));
   end;

if (FlagOut>40)  disp( '**** erreur calcul de Aref, Newton-Raphson diverge
****');
   end;
disp( '*************');
disp( '** resultat :');
disp( sprintf('iter=%d', iter));
disp( 'Les alphas =');
disp( Xn1);
disp( 'funct=');
disp( sprintf('%f   ', fun));
disp( sprintf('Erreur Max=%f', errMax));
```



Solution

After 6 iteration, the error drastically diminish corresponding to the Newton-Raphson algorithm that has a quadratic convergence
The result for a Vfmax = 155.5 V is the configuration Pulse Width Modulation (PWM) angles (0.40130, 0.60356, 0.92840 rd)

```
Vfmax =  155.50
------------------------
iter=1           funct=
87.635889     -13.029613     92.128480
Xn1=
0.447134  0.656382  0.967946
Erreur Max=92.128480
------------------------
iter=2           funct=
-15.738229     0.032983     -56.715685
Xn1=
0.405106  0.604161  0.927195
Erreur Max=56.715685
------------------------
iter=3           funct=
-0.148284     -3.618873     -1.166163
Xn1=
0.401269  0.603524  0.928398
Erreur Max=3.618873
------------------------
iter=4           funct=
-0.005571     0.012046     0.034344
Xn1=
0.401296  0.603561  0.928403
Erreur Max=0.034344
------------------------
iter=5           funct=
0.000000     -0.000002     0.000000
Xn1=
0.401296  0.603561  0.928403
Erreur Max=0.000002

------------------------
iter=6           funct=
-0.000000     0.000000     0.000000
Xn1=
0.401296  0.603561  0.928403
Erreur Max=0.000000
**********
** resultat :
iter=6
Alphas =
   0.40130
   0.60356
```

0.92840

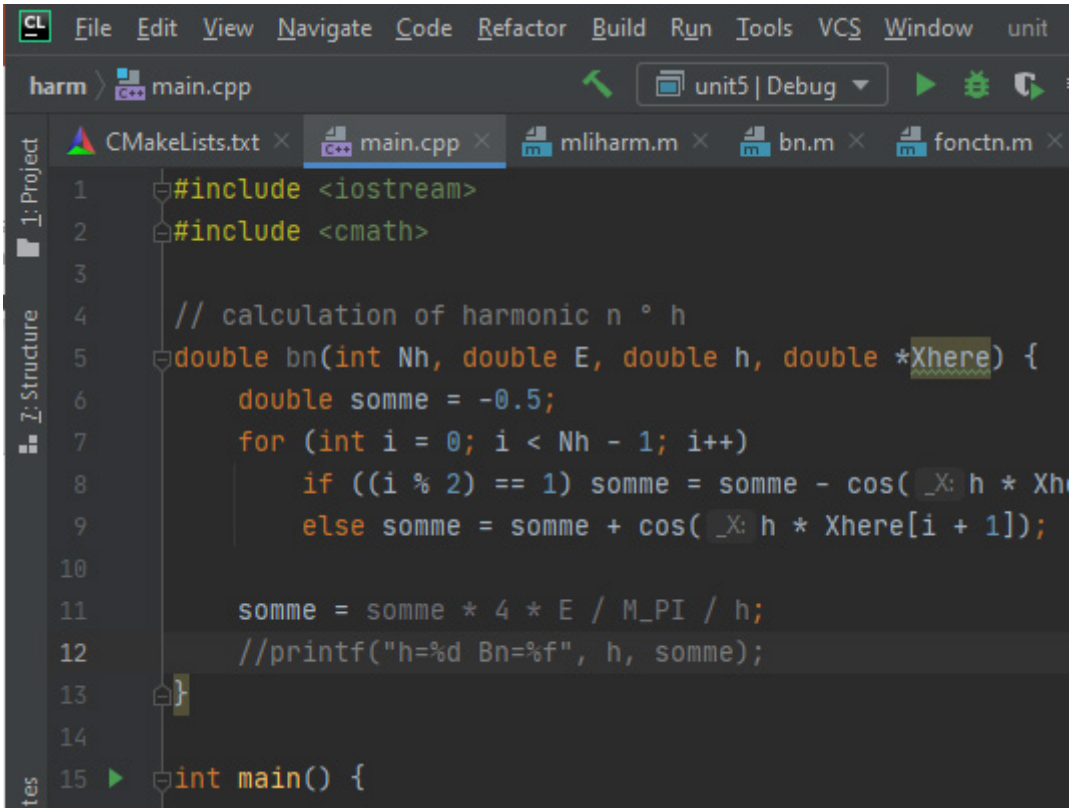
For 155.5V 0.401296 0.603561 0.928403

For 100V 0.446325 0.575147 0.972370

For 105V 0.442341 0.577730 0.968500

6.7. Assignment, PWM harmonics cancellation

Assignment task
Translate the Octave .m scripts to C++
All the files/functions must be in one main.cpp file
From the main routine, you compute the angles for a Vfmax = 155.5 V
Verify that the results of the PWM angles are the same (0.40130, 0.60356, 0.92840 rd)
Is the number of the iteration different from the Octave version



6.8. Sun position and PV panel orientation

Optimal PV panel orientation
It concerns the orientation of the panels and the calculation of the position of the sun in relation to the point of observation / installation of the PV

The sun travels along a circle in respect to the ground on 24 hours.
http://baghli.com/cs/sth3d.html

The inclination of this circle with respect to the horizon depends on the latitude of the place.
The center of this circle shifts vertically below and above the observation point depending on the day of the year

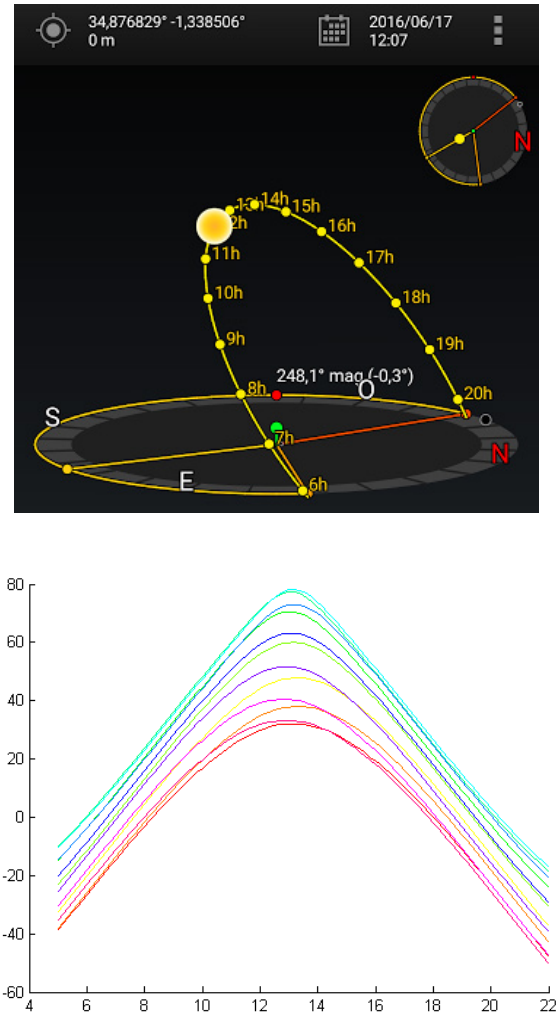


Figure 4.1. Height (in °) of the sun depending on the time of day
A negative angle means the sun is below the horizon

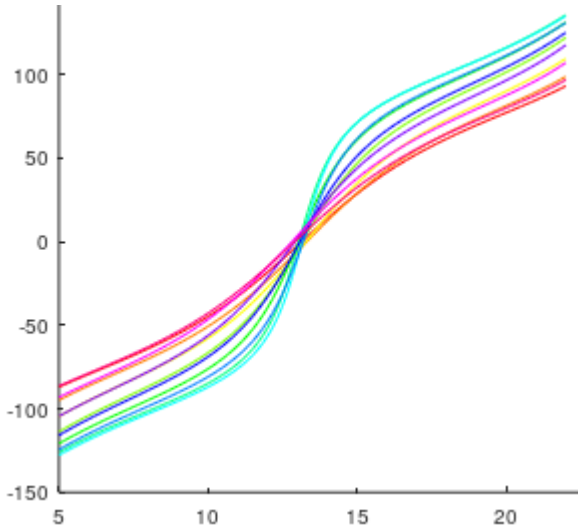


Figure 4.2. Azimut (in °) of the sun depending on the time of day.
0° indicates directly south

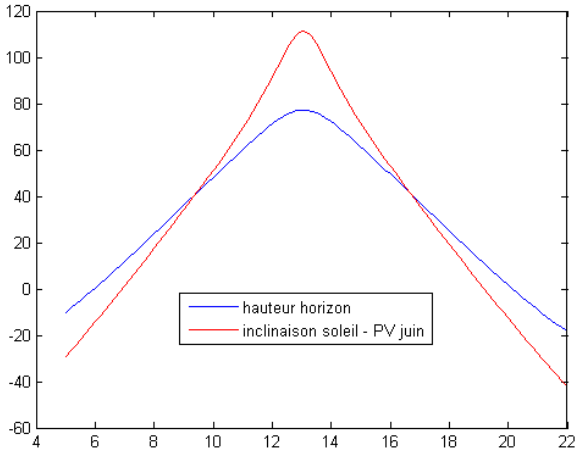


Figure 4.3. Comparison between the sun's height and the incidence of rays on the PV, according to the time of day, for the month of June. The inclination of the PV is 34°

Calculus code for Matlab, C++ and Arduino
http://baghli.blogspot.com/2016/09/orientation-dun-panneau-photovoltaique.html

6.8.1. Optimal PV panel orientation

Code for Octave to compute sun position along the months (colored graphs) and the time of the day (x-axis)

```
% Sun position over the year at Tlemcen
%[h, az] = calchaz(34.88,-1.31, 1, 0, 14, 8,6,2016 )
close all; clear all;
deg2rd = 0.017453292519943295769236907684886;
rd2deg = 57.295779513082320876798154814105;
inclinaisonPV = 34;
heure = [5:10./60:22];
for mm = 1:12
    for hh = 1:length(heure)
        [h, az] = calchaz(34.88,-1.31, 1, 0, heure(hh), 1,mm,2016 );
        Hauteur(mm,hh)=h;
        Azimut(mm,hh)=az;
        inclinaison(mm,hh)= h + rd2deg*incline(az*deg2rd,
inclinaisonPV*deg2rd);
    end
end
cc=hsv(12);
% Hauteur
figure (1);
hold on;
for mm = 1:12
    plot(heure, Hauteur(mm,:),'color',cc(mm,:));

end
% Azimut
figure (2);
hold on;
for mm = 1:12
    plot(heure, Azimut(mm,:),'color',cc(mm,:));

end

% inclinaison absolue
figure (3);
hold on;
for mm = 1:12
    plot(heure, inclinaison(mm,:),'color',cc(mm,:));

end

figure (4);
plot(heure, Hauteur(6,:),'b', heure, inclinaison(6,:),'r' );
legend('hauteur horizon', 'inclinaison soleil - PV juin');
```

Optimal PV panel orientation
Download the .m files from
https://drive.google.com/drive/u/0/folders/13GGf_4azJ6WYjrJBTLIMrc2w180M4EP9
Execute the course.m program using Octave

It uses the 2 other files that contains the functions which computes the sun position given the parameters

- Latitude: `34.88`
- Longitude: `-1.31`
- Time Zone: `1`
- DST (Daylight Saving Time): `0`
- Time: `14 h` (15.5 means 15h30 and 15.75 means 15h45)
- Day: `8,6,2016` (day,month,year)

It returns the height and azimuth in ° (deg)

```
[h, az] = calchaz(34.88,-1.31, 1, 0, 14, 8,6,2016 )
```

Observe the result in your town and determine the highest point of the sun in the day (at noon)
Use `calchaz` to compute for your town
Check it with experiments

6.8.2. Example for Tlemcen:

To compute the Zenith hour of a particular day, just give any hour for example 12
calc_z(34.88,-1.31, 1, 0, 12, 21,12,2022)
gives
13.0554 h which is 13:03:19
The sun is at it maximum height on this time

Sun position at 10 AM for March, 21st 2022:
[h,az] = calchaz(34.88,-1.31, 1, 0, 10, 21,3,2022)
gives
h = 33.4065 °
az = -63.1074 ° so in the East when you are facing South

Town	Latitude	Longitude	Irradiance Global tilted irradiation at optimum angle	June 21 Zenith hour	Sun height	Dec 21 Zenit
Biskra	34.84	5.73	2229.0	12.6493	78.5961	12.5879
Niamey	13.494	2.129	2200.7	12.8886	80.0576	12.8260
Juba	4.86	31.57	2121.5	11.9256	71.4236	11.8627
Tlemcen	34.88	-1.31	2111.6	13.1179	78.5563	13.0554
Guelma	36.38	7.76	1892.4	12.5129	77.0544	12.4503
Monrovia	6.428	9.429	1845.8	12.4019	72.9917	12.3392
Ouagadougou	12.366	-1.518	2168.92	13.1359	78.9297	13.0734
Dar es Salaam	-6.82349	39.26951	1968.2	12.4122	59.7401	12.3492
Abidjan	5.345317	-4.024429	1662.9	12.2989	71.9090	12.2364
N'Djamena	12.137752	15.054325	2241.5	12.0268	78.7014	11.9641
Khartoum	15.5	32.35	2363.8	11.8736	81.8653	11.8106
Wa	10.060074	-2.509891	2075.5	12.1994	76.6240	12.1394

Addis Abeba	8.980	38.7577	2104.2	12.4464	74.1876	12.3833

Town	Latitude	Longitude	Irradiance Global	June 21 Zenith hour	Sun height	Dec 21 Zenit	Sun height	March 21 Zenith hou	March 21 10 AMSun heigh	March 21 10 AMSu
Biskra	34.84	5.73	2229.0	12.6493	78.5961	12.5879	31.7237	12.7367	38.5158	-57.0817
Niamey	13.494	2.129	2200.7	12.8886	80.0576	12.8260	53.0703	12.9779	43.8372	-77.1055
Juba	4.86	31.57	2121.5	11.9256	71.4236	11.8627	61.6333	12.0156	59.4525	-82.2657
Tlemcen	34.88	-1.31	2111.6	13.1179	78.5563	13.0554	31.6843	13.2072	33.4065	-63.1074
Guelma	36.38	7.76	1892.4	12.5129	77.0544	12.4503	30.1824	12.6024	38.9240	-54.0642
Monrovia	6.428	9.429	1845.8	12.4019	72.9917	12.3392	60.1364	12.4914	60.0533	-82.1314
Ouagadougou	12.366	-1.518	2168.92	13.1359	78.9297	13.0734	54.1983	13.2252	77.9790	-79.5985
Dar es Salaam	-6.82349	39.26951	1968.2	12.4122	59.7401	12.3492	73.3882	12.5024	51.8801	-99.1995
Abidjan	5.345317	-4.024429	1662.9	12.2989	71.9090	12.2364	61.2189	12.3881	53.8749	-83.1644
N'Djamena	12.137752	15.054325	2241.5	12.0268	78.7014	11.9641	54.4267	12.1165	56.3433	-71.7231
Khartoum	15.5	32.35	2363.8	11.8736	81.8653	11.8106	61.2189	11.9636	57.1719	-65.1239
Wa	10.060074	-2.509891	2075.5	12.1994	76.6240	12.1394	56.5036	12.2850	54.5520	-76.2997
Addis Abeba	8.980	38.7577	2104.2	12.4464	74.1876	12.3833	57.1010	12.5366	51.1192	-79.1238

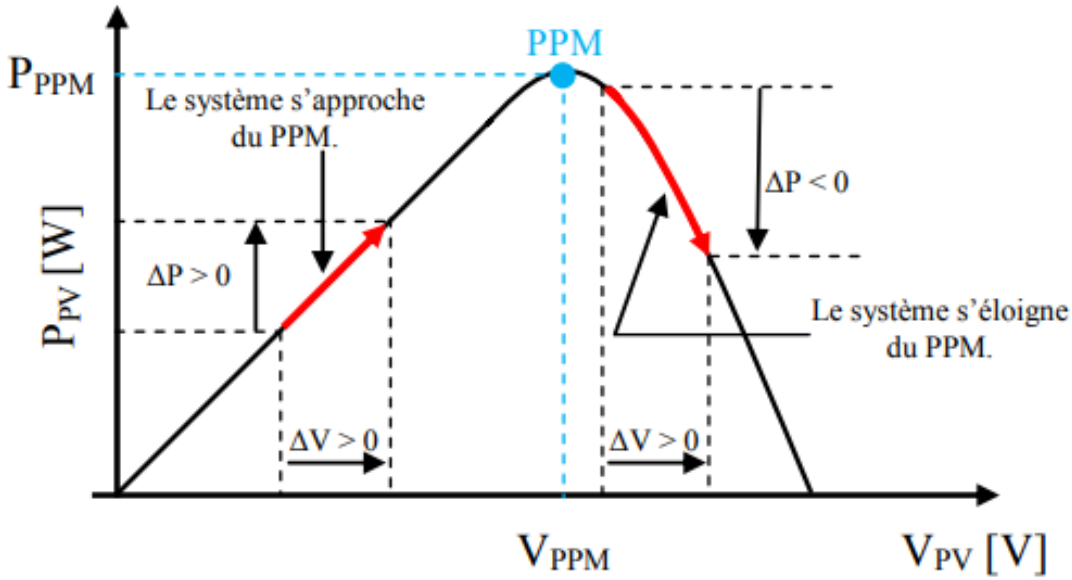
Have a look to the animation to understand how the shadow move
https://fr.wikipedia.org/wiki/Gnomon#/media/Fichier:Gnomon--21juin.gif

6.9. Maximum Power Point Tracking of a PV panel

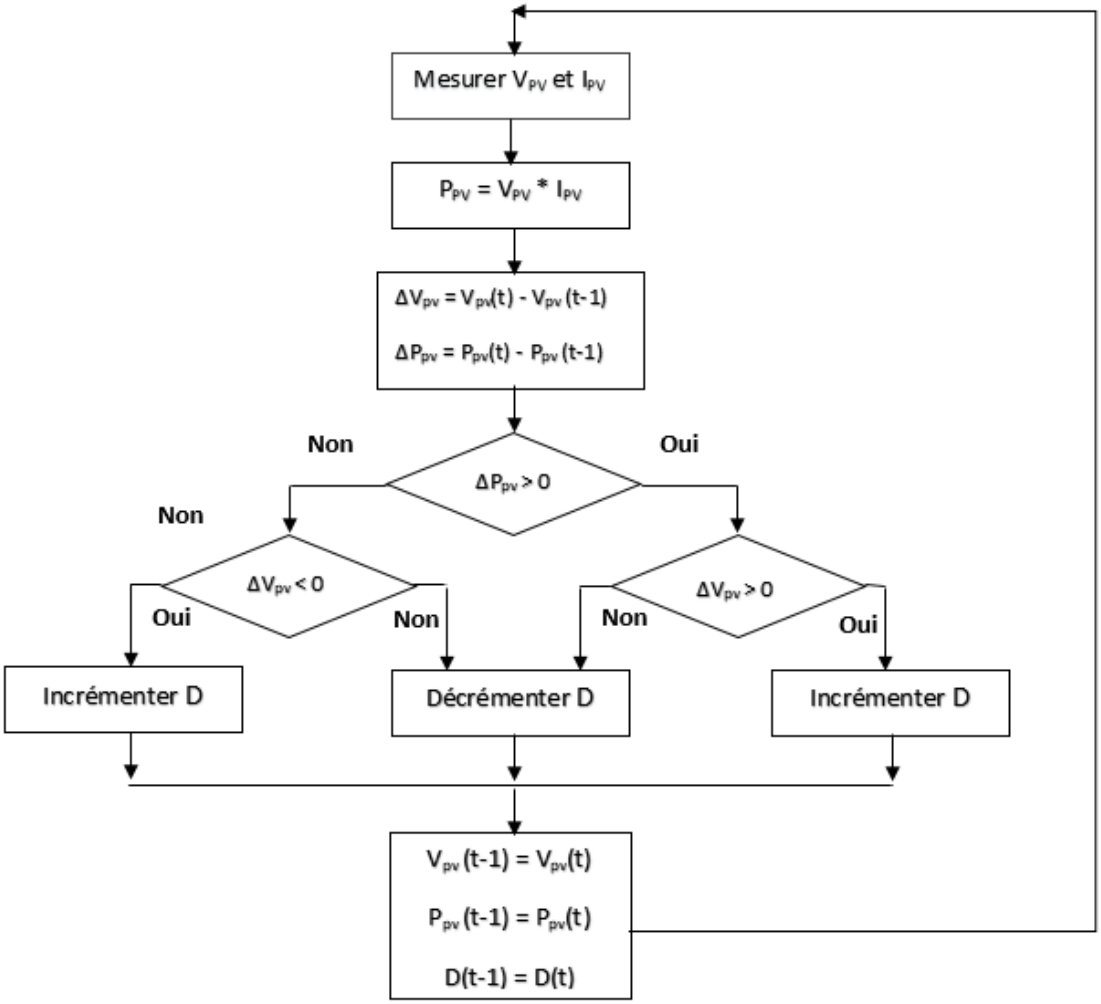
6.9.1. Model of a PV panel, MPPT algorithm

$$I = I_{ph} - I_o\left[\exp\left(\frac{V + R_s I}{V_t \alpha}\right)\right] - \left(\frac{V + R_s I}{R_p}\right)$$

$I_{ph} = (I_{pv,n} + K_I(T_n - T))\frac{G}{G_n}$ and $I_o = \frac{I_{sc} + K_I \Delta t}{\exp((V_{oc} + K_V \Delta t)/V_t \alpha) - 1}$ and $V_t = \frac{N_s K T}{q}$

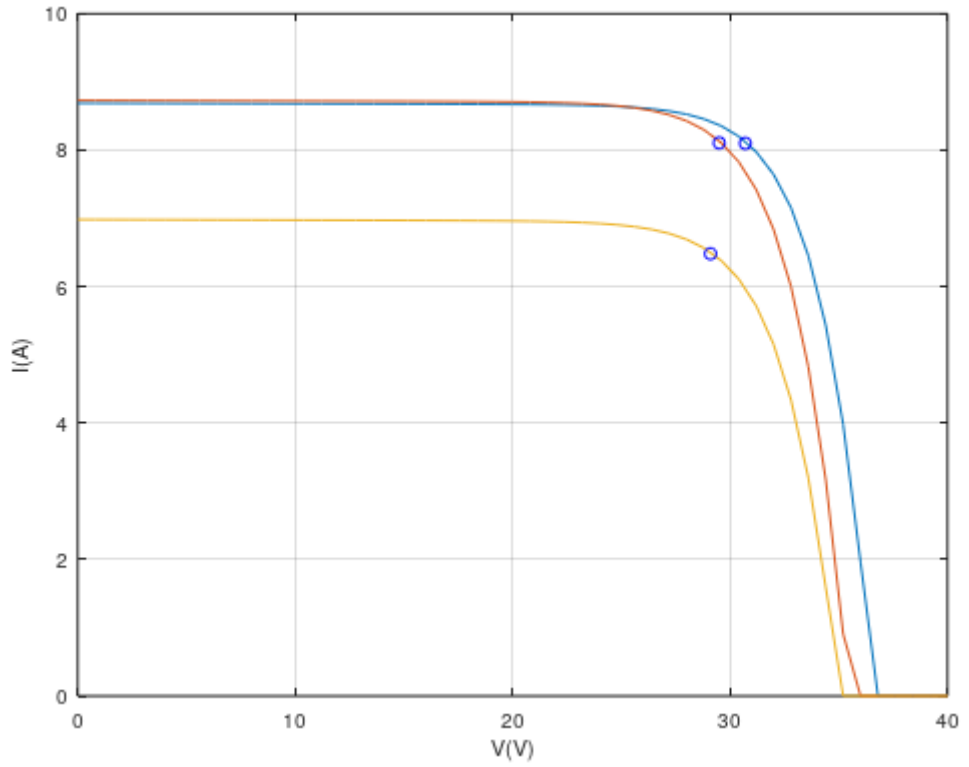


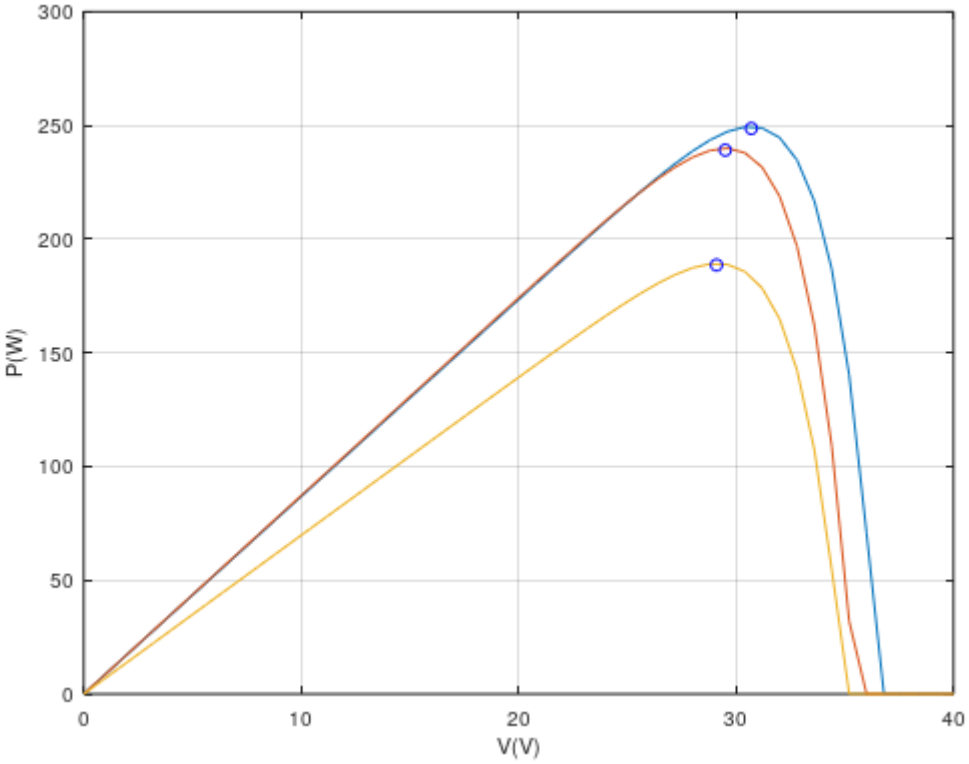
Disturb and observe algorithm



Model of a PV panel, MPPT algorithm
The circle indicates the maximum power point
The I-V curves are plotted while varying the voltage
We need to solve by Newton-Raphson the equation in order to compute the corresponding current for a voltage value

```
>> plotpv(1000,25)
ans =  30.700
>> plotpv(1000,35)
ans =  29.500
>> plotpv(800,35)
ans =  29.100
```





Model of a PV panel, I-V curves and MPP
The circle indicates the maximum power point

The I-V curves are plotted while varying the voltage
We need to solve by Newton-Raphson the equation in order to compute the corresponding current for a voltage value.

6.9.2. Program

The program is composed of 3 .m functions
`plotpv(Suns,Tc)`
`mppt(Suns,Tc)`
`PVNR(V,G,T)`

plotpv(Suns,Tc)

```
function [Vs, Is, Ps] = plotpv(Suns,Tc)
[Vs Is Ps] = mppt(Suns,Tc);

Vt=zeros(1,50);
It=zeros(1,50);
Pt=zeros(1,50);
for k = 0:1:50
    V = k*40/50;
    I = PVNR(V,Suns,Tc);
    P=V*I;
    Vt(k+1)= V;
    It(k+1)= I;
    Pt(k+1)= P;
end

figure(1)
plot(Vt,It);
axis([0 40 0 10]);
xlabel('V(V)');
ylabel('I(A)');
hold on;
plot(Vs, Is,'ob');
grid on;

figure(2)
plot(Vt,Pt);
axis([0 40 0 300]);
xlabel('V(V)');
ylabel('P(W)');
grid on;
hold on;
plot(Vs, Ps,'ob');
end
```

mppt(Suns,Tc)
This function is called from the previous function plotPV

To find the maximum power point
We start from a point of 20V on the I-V curves
We apply the perturb and observe algorithm
mppt(Suns,Tc)

```
function [V, I, P] = mppt(Suns,Tc)
V=20;
I = PVNR(V,Suns,Tc);
P=I*V;
D = 0.1;
Vnew = V+D;
dP = 1;
iter=0;
while (abs (dP)> 1e-2 && (iter<200))
    iter=iter+1;
    I = PVNR(Vnew,Suns,Tc);
    Pnew=I*Vnew;
    dP = Pnew-P;
    dV = Vnew - V;
    V = Vnew;
    P = Pnew;
    sprintf('iter=%d V=%g I=%g, P=%g\n', iter, V, I, P);
    if (dP>0)
        if (dV>0) Vnew = V + D;
        else  Vnew = V - D;
        end
    else
        if (dV<0) Vnew = V + D;
        else  Vnew = V - D;
        end
    end

end
% I;
V=Vnew;
P=I*V;
end
```

PVNR(V,G,T)
Model of a PV panel, I-V curve
This function is called by the previous function
It is the one that find the operating point on the I-V curves for a given voltage
It solves by Newton-Raphson the equation in order to compute the corresponding current for a voltage value
We need iterations until we fall bellow 0.1% of error

PVNR(V,G,T)

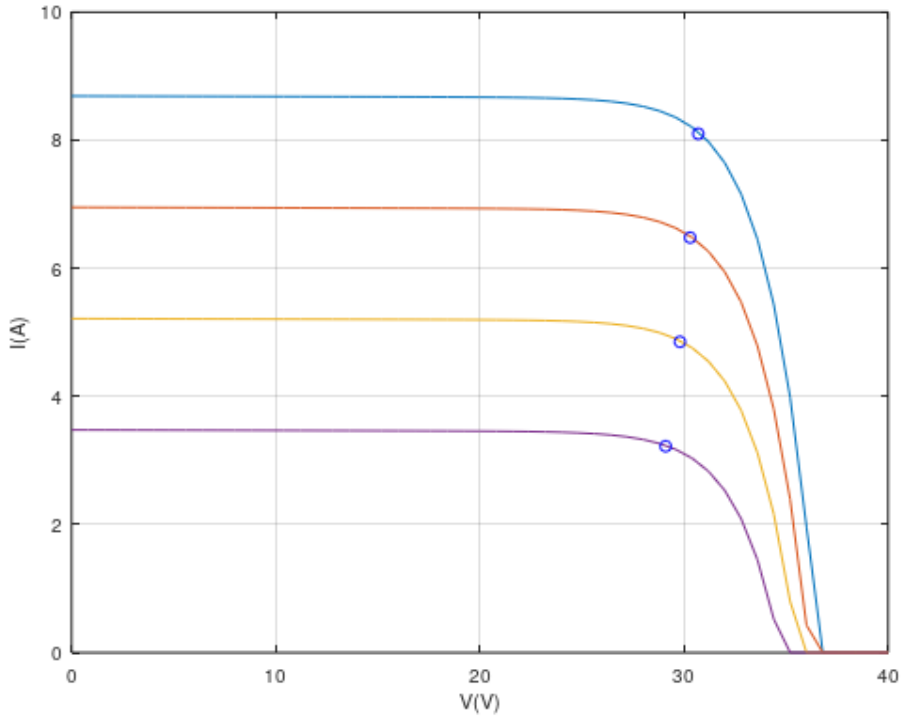
```
function I = PVNR(V,G,T)
T=T+273.15;
V=V/60;

Isc=8.69; Voc=0.61;
Imp=8.19; Vmp=30.55; Pmax_e=250; Kv=-0.0038; Ki=0.004; Ns=60;
k=1.3806503e-23; q=1.60217646e-19;
a=1.3; Rs=0.04/60; Rp=(1.2711e+03)/60;
Gn=1000; Tn=298.15; dT=T-Tn;
Vt=k*T/q; Vtn=k*Tn/q;

Io=(Isc+Ki*dT)/(exp(q*((Voc)+Kv*dT)/(a*k*Tn))-1);
Iph=(Isc+Ki*dT)*G/Gn;
I=4;
    g=Iph-Io*(exp(q*(V+I*Rs)/(k*T*a))-1)-((V+I*Rs)/Rp)-I;

    while(abs(g)>0.001)
      g=Iph-Io*(exp(q*(V+I*Rs)/(k*T*a))-1)-((V+I*Rs)/Rp)-I;
      glin=(-Io*q*Rs/(k*T*a))*exp(q*(V+I*Rs)/(k*T*a))-(Rs/Rp)-1;
  if (glin !=0)
    I=I-g/glin;
    else
    sprintf('error glin zero')
    break
    end
  end
  if I<0
    I=0;
    end
end
```

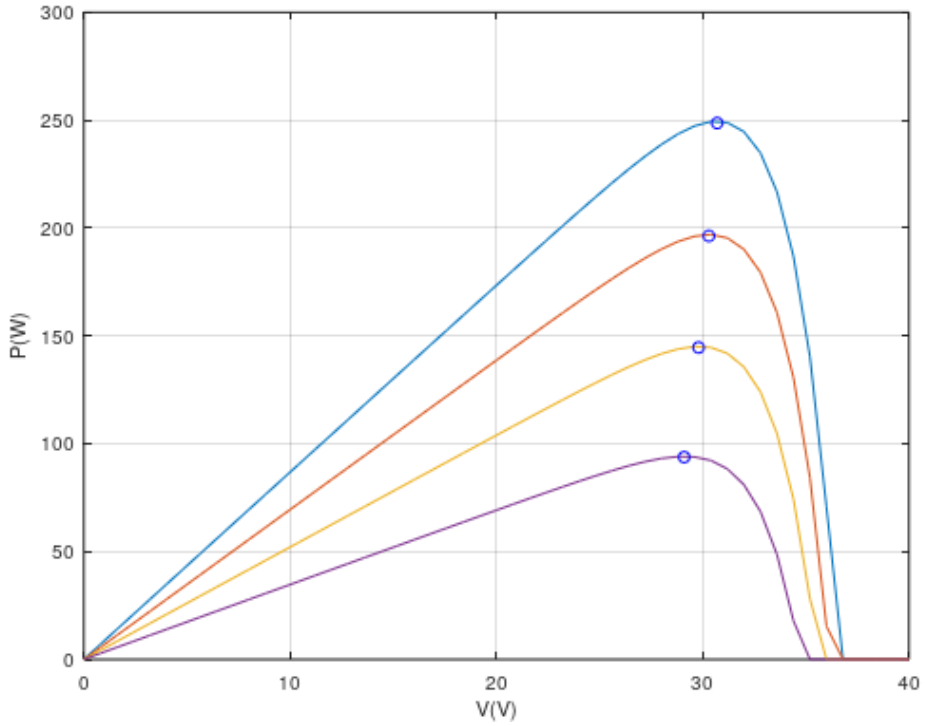
6.9.3. Activity: Assessment

Create the 3 script files with their corresponding names `plotpv.m, mppt.m, PVNR.m`
Copy the code and run `plotpv` program using Octave

plotpv(1000,25) will display the curve for a solar radiation of 1000 W/m^2 and a temperature of 25°C

Plot on the same graph the curves corresponding to 25°C and 1000, 800, 600 and 400 W/m^2
Determine the exact value of the maximum power point, what do you notice?





6.10. Assignement: BMS battery emulator

We need to build a battery emulator with its BMS.

1s is 10 min (accelerated time)

The model should include charging, discharging process, State Of Charge (SOC)

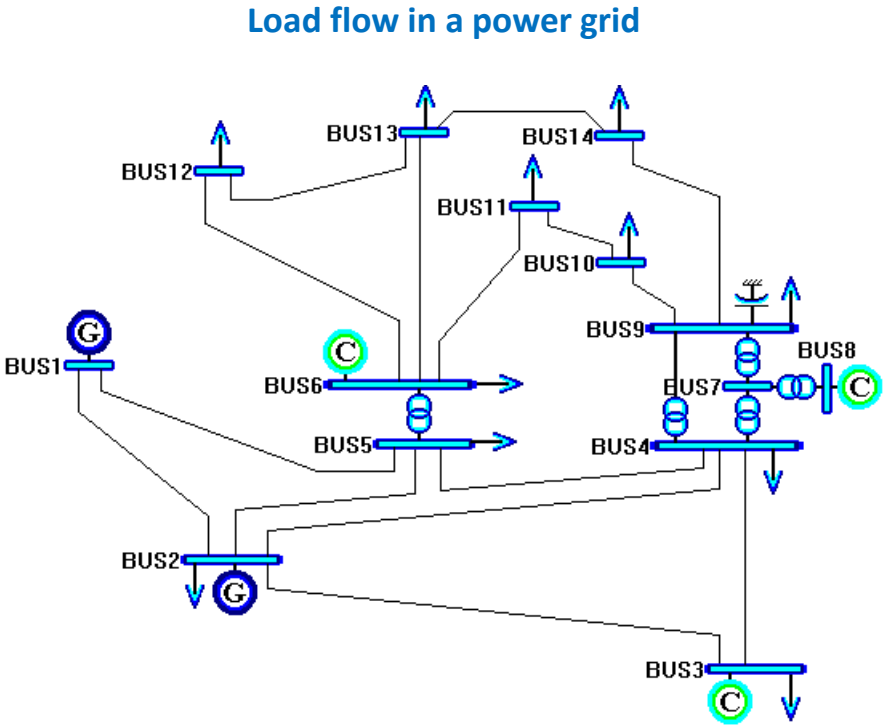
Simulation using accelerated time in a loop

Advanced features (optional)

6.11. Summary

Chose the appropriate programing language (C++ or MATLAB/Octave) to solve a given problem
Complex calculus and transfer function are better handled with Octave
Iterative and intensive calculus are solved with compiled C++ programs faster than with Octave

7. Load flow in a power grid

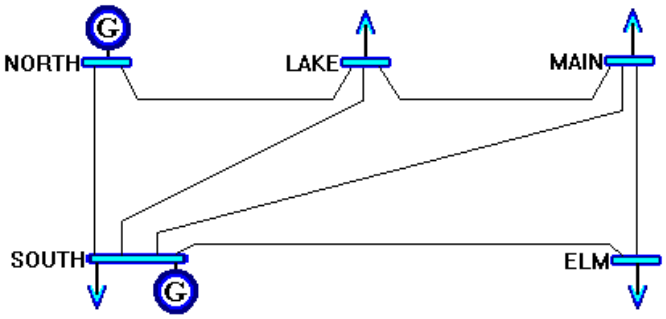


Learn how a power grid is modelized and the load flow is computed
See the application of Gauss-Seidel and Newton-Raphson methods to solve the LF
Compile and simulate by yourself in C++ with the given program

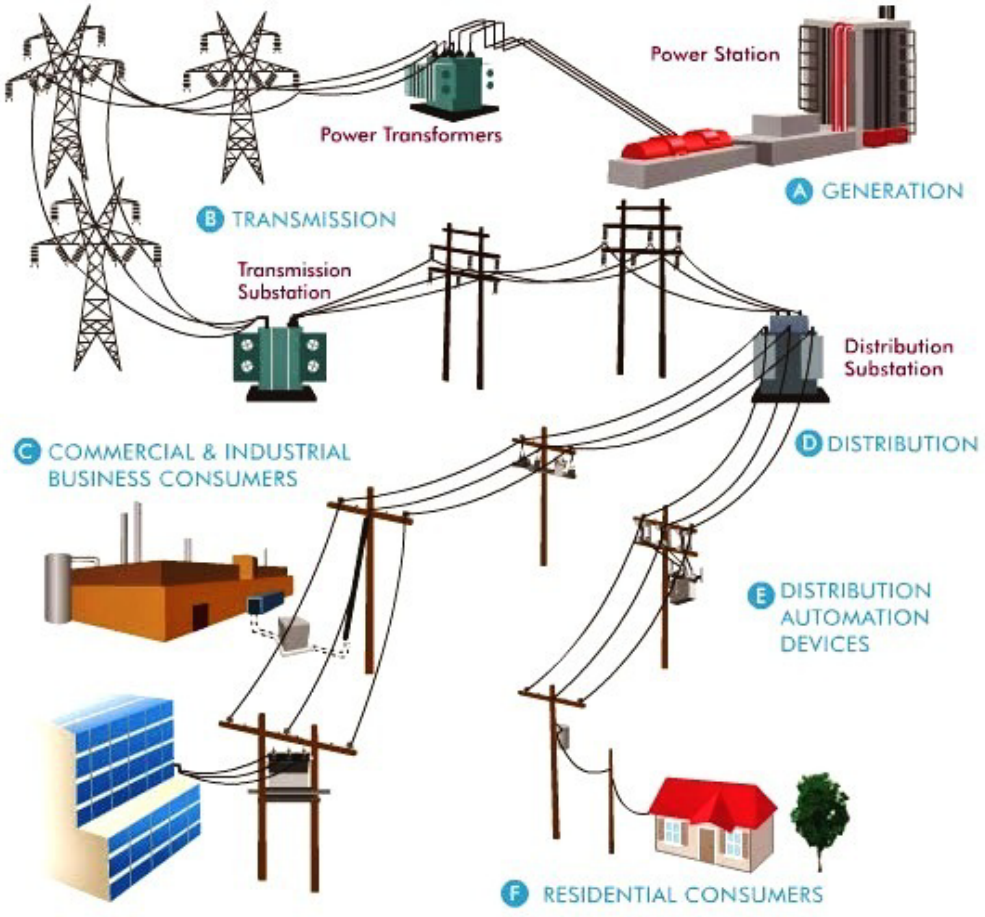
References
C++ Tutorials and help http://www.cplusplus.com/doc/tutorial/control/
J. Arrillaga, C. P. Arnold , Computer Analysis of Power Systems, John Wiley & Sons, 1990, http://eword.yolasite.com/resources/51621752-Computer-Analysis-of-Power-Systems.pdf

7.1. Introduction

An interesting application of C++ programming is to model the power grid and simulate it
A power grid can be studied at different levels
Production, transmission, distribution, consumption
We consider here only the high voltage transmission grid



Each node (bus) represents a high voltage station
We will show how modified versions of Gauss Seidel and Newton-Raphson algorithms are implemented in C++ to compute the load flow between the buses of a power grid
C++ allows a seamless use of complex variables



http://www.actemium.net.in/wp-content/uploads/2016/11/Battery-1.jpg

7.1. Power definition

In AC systems we generally consider 3-phase systems.

The power is represented by Active (P in W), Reactive (Qin VAr) and Apparent Power (S in VA)

$$S = \sqrt{P^2 + Q^2}$$

$P = \sqrt{3}UIcos\varphi = PF$ the power factor (pure sinusoidal signals), that we need to keep close to 1

Where $U = \sqrt{3}V$ is the line to line voltage

And

V is the line voltage (in respect to the neutral point

$$Q = \sqrt{3}UIsin\varphi$$

If there are power electronics involved, there is an additional D Distorsion power in W

$$S = \sqrt{P^2 + Q^2 + D^2}$$

In single phase

$$P = VIcos\varphi$$

$cos\varphi = PF$ if sinusoidal

Using complex representation

$$S = VI^*$$

$$P = Re(VI^*)$$

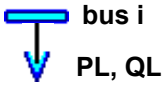
$$Q = Im(VI^*)$$

7.2. Power transfer

Between 2 nodes or buses in AC systems, power flows if there is difference in term of phase and magnitude

7.3. Power grid elements

We consider a phaser representation of the 3-phases system, in steady state operation

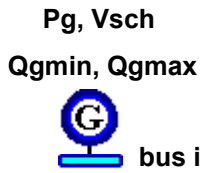


At each bus (node), the power injection and connection are considered

Load

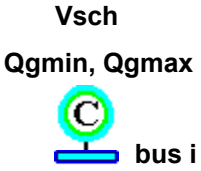
The load is modeled by an impedance which consumes a constant active power PL and reactive power QL

Generator



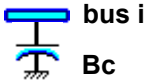
A generator is represented by a source of constant voltage Vmag = Vsch which injects at its bus an active power Pg and reactive power Qg.
Pg is constant throughout the calculation, Qg on the other hand varies in order to maintain Vmag = Vsch. If Qg reaches one of the two limits (Qgmin or Qgmax), it is fixed at this limit and Vmag is released

Compensator



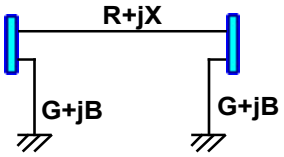
A synchronous compensator is a reactive power generator. It is represented by a generator with
Pg = 0

Static Compensator



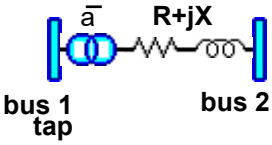
The simple static compensator is modeled by a shunt capacitor whose admittance is jBc = jCW

Line



The lines are represented by their equivalent π diagram. The R + jX is the series impedance. There is a shunt admittance at each arm G + jB.
A line has a maximum transit power. If the transit power is greater than this limit, a warning is generated

Transformers

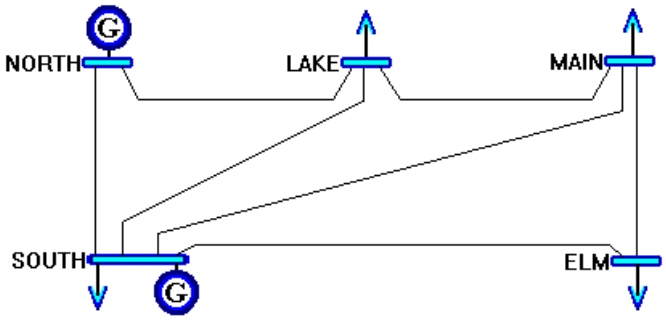


Transformers are represented by their admittance matrix
Its transformation ratio a = Amag * Adelta which may or may not be complex.
The leakage impedance Z = R + jX.
It has also a maximum transit power.
In the case where the argument of "a" is not zero, an equivalent π scheme does not exist. We represent the transformer by its admittance matrix which is in this case not symmetrical (Y21 * Y12). Therefore, we need to store the entire matrix Y and not just its upper triangular part.

$$\begin{bmatrix} I1 \\ I2 \end{bmatrix} = \begin{bmatrix} \frac{1}{Z \cdot |a^2|} & -\frac{1}{Z \cdot a*} \\ -\frac{1}{Z \cdot a} & \frac{1}{Z} \end{bmatrix} \begin{bmatrix} V1 \\ V2 \end{bmatrix}$$

Power grid

- 1 slack bus (reference bus or swing bus): most powerful bus in term of power generation
- PV buses where generators (power plants) are connected. We specify the generated power and the voltage magnitude
- PQ buses with only loads and lines connected. We specify the active (P) and reactive (Q) loads
- We also enter the power transportation lines (π model) parameters (R+jX, G+jB)



5-bus power grid
http://baghli.com/powerd.html

Data of the Abiad 5-bus power grid

8	----------	Bus Data	----------				
8	BUS	Pg	Qg	Pl	Ql	Vb	
1	NORTH	0	0	0	0	1.06	
1	SOUTH	0.4	0	0.2	0.1	1.0475	
1	LAKE	0	0	0.45	0.15	1	
1	MAIN	0	0	0.4	0.05	1	
1	ELM	0	0	0.6	0.1	1	
8	----------	Active Bus Limits	----------				
8	BUS	Qgmin	Qgmax				
2	SOUTH	-0.3	0.4				
8	----------	Line Data	----------				
8	BUS	BUS	Rs	Xs	Gs	Bs	Smax
3	NORTH	LAKE	0.08	0.24	0	0.025	0
3	LAKE	MAIN	0.01	0.03	0	0.01	0
3	MAIN	ELM	0.08	0.24	0	0.025	0
3	NORTH	SOUTH	0.02	0.06	0	0.03	0
3	SOUTH	LAKE	0.06	0.18	0	0.02	0
3	SOUTH	MAIN	0.06	0.18	0	0.02	0
3	SOUTH	ELM	0.04	0.12	0	0.015	0

7.4. Gauss-Seidel algorithm

The GS method is easy to implant and is suitable for small-sized networks
For bigger networks, the convergence is attainable after a large number of iterations
Flowchart
The load flow algorithm is iterated until the voltage difference, at each bus, between two successive iterations, falls under a fixed tolerance

The current injected through the i^{th} bus to the network equals

$$I_i = \sum_{k=1}^{N} I_{ik} = \sum_{k=1}^{N} Y_{ik}V_k = Y_{ii}V_i + \sum_{\substack{k=1 \\ k\neq i}}^{N} Y_{ik}V_k$$

Y_{ii} represents the admittance at the node i and Y_{ik} represents the admittance between this node and another one
The Y matrix is constituted from all these elements
It is built thanks to the power system elements, prior to power flow calculation.

The power at a node is expressed as

$$S_i = P_i + jQ_i = V_i I_i^*$$

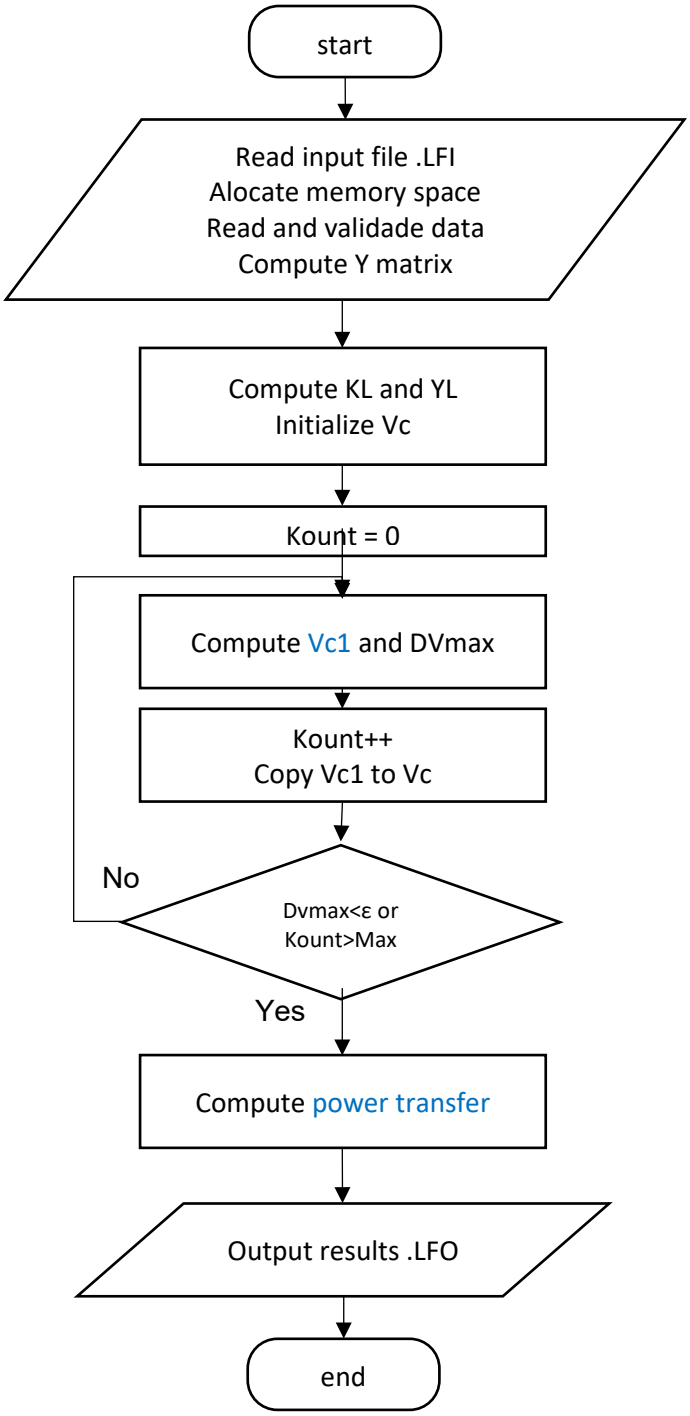
$$I_i = \frac{P_i - jQ_i}{V_i^*} = \sum_{k=1}^{N} Y_{ik} V_k = Y_{ii} V_i + \sum_{\substack{k=1 \\ k \neq i}}^{N} Y_{ik} V_k$$

$$V_i = \underbrace{\frac{P_i - jQ_i}{Y_{ii}}}_{KL_i} \frac{1}{V_i^*} - \sum_{\substack{k=1 \\ k \neq i}}^{N} \underbrace{\frac{Y_{ik}}{Y_{ii}}}_{YL_{ik}} V_k$$

To compute ${V_i}^{m+1}$, the voltage, at the bus numbered *i*, at iteration *m*+1, we use voltages computed at previous iterations ${V_i}^{m}$

$${V_i}^{m+1} = KL_i \frac{1}{{V_i^*}^m} - \sum_{\substack{k=1 \\ k \neq i}}^{N} YL_{ik} {V_k}^m$$

In the algorithm, we use Vc1 for the new vector V^{m+1} and Vc for the vector V^m of the previous iteration



A slightly modified version allows the use of the recently computed voltages at iteration $m+1$; those below the i^{th} node to evaluate the voltage of bus i

$$V_i^{m+1} = KL_i \frac{1}{V_i^{*m}} - \sum_{k=1}^{i-1} YL_{ik} V_k^{m+1} - \sum_{k=i+1}^{N} YL_{ik} V_k^{m}$$

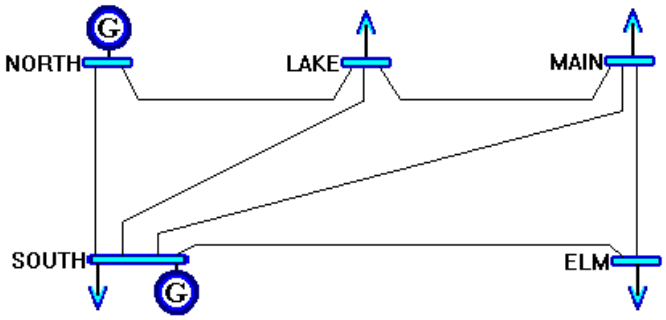
These equations are suitable for PQ buses

For PV buses, the algorithm is adapted, because the voltage magnitude is kept constant to its nominal value and the active power is also fixed, the phase is free to change.
The active power produced is fixed to the one of the generator connected to that bus.
Only the phase of the voltage vector at a PV bus changes
If the reactive power produced by the generator connected to a PV bus reaches its limit, then the PV bus becomes a PQ bus with a fixed power and with a voltage vector free to change in magnitude and phase

- Read power grid data
- Compute Y matrix
- Execute load flow computation algorithm (Gauss-Seidel, Newton-Raphson or FDLF)
- End of simulation criterion: Voltage stabilization (GS) or power mismatch (NR, FDLF)
- Compute data at each bus
- Compute injected power (generated, demand) at each bus
- Compute line power transfer

```
---- YBUS matrix  ( no null elements )------
Busi          Busj                    real          imag          magnitude     arg(deg)
NORTH         NORTH         Y[0 ,0 ]=6.25           -18.695       19.712        -71.514
NORTH         SOUTH         Y[0 ,1 ]=-5             15            15.811        108.43
NORTH         LAKE          Y[0 ,2 ]=-1.25          3.75          3.9528        108.43
SOUTH         SOUTH         Y[1 ,1 ]=10.833         -32.415       34.177        -71.52
SOUTH         LAKE          Y[1 ,2 ]=-1.6667        5             5.2705        108.43
SOUTH         MAIN          Y[1 ,3 ]=-1.6667        5             5.2705        108.43
SOUTH         ELM           Y[1 ,4 ]=-2.5           7.5           7.9057        108.43
LAKE          LAKE          Y[2 ,2 ]=12.917         -38.695       40.794        -71.541
LAKE          MAIN          Y[2 ,3 ]=-10            30            31.623        108.43
MAIN          MAIN          Y[3 ,3 ]=12.917         -38.695       40.794        -71.541
MAIN          ELM           Y[3 ,4 ]=-1.25          3.75          3.9528        108.43
ELM           ELM           Y[4 ,4 ]=3.75           -11.21        11.821        -71.504
```

```
Bus name : SOUTH                    Bus number : 1
                      real          imag          magnitude     arg(deg)
  Voltage           : 1.0462        -0.051287     1.0475        -2.8065
  Generated power   : 0.4           0.3004        0.50024       36.907
  Power demand      : -0.2          -0.1          0.22361       -153.43
  Power transfer    : Bus name       Bus n'       real          imag
        to          : NORTH          0           -0.87443      0.094925
        to          : LAKE           2            0.24691      0.057391
        to          : MAIN           3            0.27932      0.051541
        to          : ELM            4            0.54812      0.089807
              Total Mismatch                      real          imag
                                                  8.445e-05     -2.776e-17
```



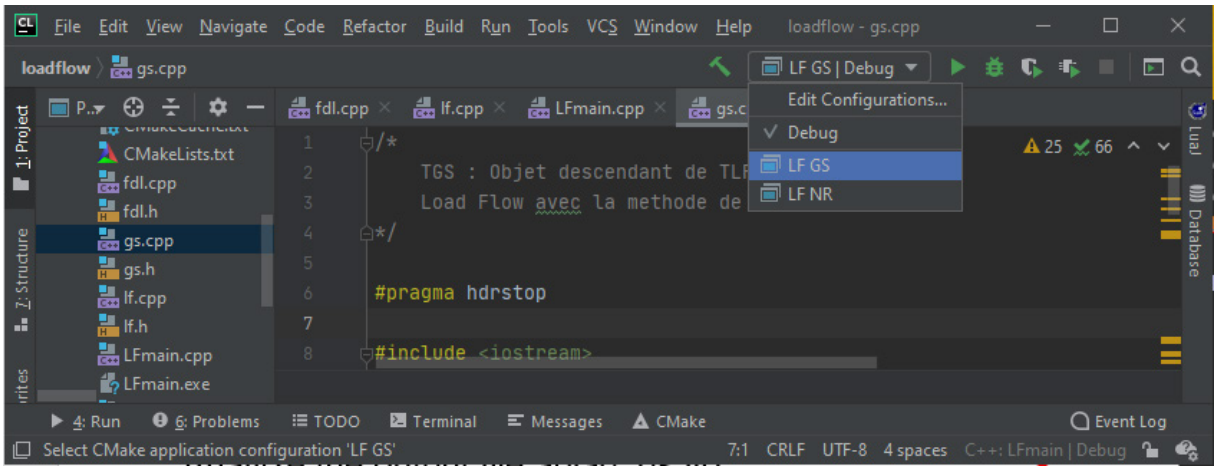
5-bus power grid
http://baghli.com/powerd.html

7.5. Load Flow – GS Application

In C++
Assignment
Using the C++ program provided in the Zip file
Delete cmake_install.cmake
Delete CMakeCache.txt
Reload CMake Project on CMakeLists.txt

- Compile and run with the profile LF GS
- Compute the load flow
- Analyze the input file abiad.lfi
- Analyze the output file abiad_gs.lfo
- Compare the results with the ones of the previous slide

```
//---------------------------------------------------------------
// Load Flow
// Lotfi BAGHLI 06/2020
int __fastcall CallLoadFlow( int LFint, char * InN, char * OutN)
{
TLF * LFProg;
  switch( LFint)
      {
      case id_GS :   LFProg=new TGS( InN, OutN); break;
      case id_NR :   LFProg=new TNR( InN, OutN); break;
      case id_NRB :  LFProg=new TNRB( InN, OutN); break;
      case id_FDL :  LFProg=new TFDL( InN, OutN); break;
      default :  return AllocError;
      }
```


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
loadflow - gs.cpp
loadflow
gs.cpp
LF GS | Debug
Edit Configurations...
Debug
LF GS
LF NR
fdl.cpp
lf.cpp
LFmain.cpp
CMakeLists.txt
fdl.cpp
fdl.h
gs.cpp
gs.h
lf.cpp
lf.h
LFmain.cpp
LFmain.exe
/*
TGS : Objet descendant de TLF
Load Flow avec la methode de
*/
#pragma hdrstop
#include <iostream>
25
66
1: Project
7: Structure
Database
4: Run
6: Problems
TODO
Terminal
Messages
CMake
Event Log
Select CMake application configuration 'LF GS'
7:1
CRLF
UTF-8
4 spaces
C++: LFmain | Debug

7.6. Newton-Raphson algorithm

The Newton-Raphson (NR) algorithm converges quadratically
It takes less iterations than the Gauss-Seidel one
It is a more complicated algorithm to implement, it takes a larger amount of memory and it is computation intensive
For this reason, the Fast Decoupled Load Flow (FDLF) reduces the matrix size by one half, while conserving the quadratic convergence property of N-R, but this is beyond the target of the course
We present the NR method to compute the load flow.
The constraint is to reduce to zero the power mismatch ΔP and ΔQ at all buses
The resolution of the system needs to compute the Jacobian *J*

$$J \cdot \begin{bmatrix} \Delta\delta \\ \Delta V \end{bmatrix} = -\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} \quad \text{where} \quad J = \begin{bmatrix} \frac{\partial \Delta P}{\partial \delta} & \frac{\partial \Delta P}{\partial V} \\ \frac{\partial \Delta Q}{\partial \delta} & \frac{\partial \Delta Q}{\partial V} \end{bmatrix}$$

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} 0 \\ \Delta P_1 \\ . \\ \Delta P_N \\ 0 \\ 0 \\ . \\ \Delta Q_{Npv+1} \\ . \\ \Delta Q_N \end{bmatrix}$$

The Jacobian *J* is computed at each iteration
Each part is computed apart

Example for J1

indices de noeuds	0	1	·	·	·	·	N	$N+1$	·	$N+Npv$	$N+Npv+1$	·	$2N$
0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	$\frac{\partial \Delta P_1}{\partial \delta_1}$	·	·	·	·	$\frac{\partial \Delta P_1}{\partial \delta_N}$	0	0	·	$\frac{\partial \Delta P_1}{\partial V_{Npv+1}}$	·	$\frac{\partial \Delta P_1}{\partial V_N}$
·	0	·	·	·	·	·	·	0	0	·	·	·	·
·	0	·	·	·	·	·	·	0	0	·	·	·	·
·	0	·	·	·	·	·	·	0	0	·	·	·	·
·	0	·	·	·	·	·	·	0	0	·	·	·	·
N	0	$\frac{\partial \Delta P_N}{\partial \delta_1}$	·	·	·	·	$\frac{\partial \Delta P_N}{\partial \delta_N}$	0	0	·	$\frac{\partial \Delta P_N}{\partial V_{Npv+1}}$	·	$\frac{\partial \Delta P_N}{\partial V_N}$
$N+1$	0	0	0	0	0	0	0	1	0	0	0	0	0
·	0	0	0	0	0	0	0	0	1	0	0	0	0
$N+Npv$	0	·	·	·	·	·	·	0	0	·	·	·	·
$N+Npv+1$	0	$\frac{\partial \Delta Q_{Npv+1}}{\partial \delta_1}$	·	·	·	·	$\frac{\partial \Delta Q_{Npv+1}}{\partial \delta_N}$	0	0	·	$\frac{\partial \Delta Q_{Npv+1}}{\partial V_{Npv+1}}$	·	$\frac{\partial \Delta Q_{Npv+1}}{\partial V_N}$
·	0	·	·	·	·	·	·	0	0	·	·	·	·
$2N$	0	$\frac{\partial \Delta Q_N}{\partial \delta_1}$	·	·	·	·	$\frac{\partial \Delta Q_N}{\partial \delta_N}$	0	0	·	$\frac{\partial \Delta Q_N}{\partial V_{Npv+1}}$	·	$\frac{\partial \Delta Q_N}{\partial V_N}$

```
for ( i=1; i<=N; i++)   // avoid slack bus
  {
// J1
  Jc[i][i]=0;
  for ( j=0; j<=N; j++)
      if ( i!=j ) {
        Jc[i][i]+=abs(Y[i][j])*Vmag[j]*Vmag[i]*
             sin( -argn(Y[i][j])-delta[j]+delta[i] );
        if ( j!=0 ) Jc[i][j]=-abs(Y[i][j])*Vmag[j]*Vmag[i]*
                    sin( -argn(Y[i][j])-delta[j]+delta[i] );
        }

...
  }
```

The Jacobian is computed with the function `JacobNR()`
The resolution of the equation system permits to find the increment in angles and voltage magnitude

Gaussian elimination is used

```
void TNR::Run()
{
int i;
for ( i=0; i<=N; i++)
      if (i<=Npv || Qgmin[i]!=0 || Qgmax[i]!=0 )
             { Vmag[i]=Vb[i];
               delta[i]=0;
             }
       else  { Vmag[i]=Vb[0]; // slack bus magn
               delta[i]=0;
             }
   Kount=0;
   do{
      if (LFOpt & id_V_auxOut)  OutV_delta();
      CalcMismatch(); //  call ControlNR()
      CalcWorstNR();
      if (LFOpt & id_ArrilagaOut) OutWorstNR();
      if ( Worst<=Tol || Kount>=NIterMax ) break;
      Kount++;
      JacobNR();
      // solve
      Gauss();
      Split();  //  transform x in d_delta and d_V then update
      if (LFOpt & id_JacobienOut)  OutX();
      } while (1);
}
```

```
void TNR::Gauss()
{
int k, i, j;
double M;
// Triangularisation de la matrice
  for (k=0; k<=2*N-1; k++)
     {
     if (Jc[k][k]==0)  Pivot( k);
     for (i=k+1; i<=2*N; i++)
    {
  M=Jc[i][k]/Jc[k][k];
    fpq[i]=fpq[i]-M*fpq[k];
    for (j=k; j<=2*N; j++)  Jc[i][j]-=M*Jc[k][j];
    }
     }
  //  Resoulution
  x[2*N]=fpq[2*N]/Jc[2*N][2*N];
  for (i=2*N-1; i>=0; i--)
     {
     x[i]=fpq[i];
     for (j=i+1; j<=2*N; j++)  x[i]-=Jc[i][j]*x[j];
```

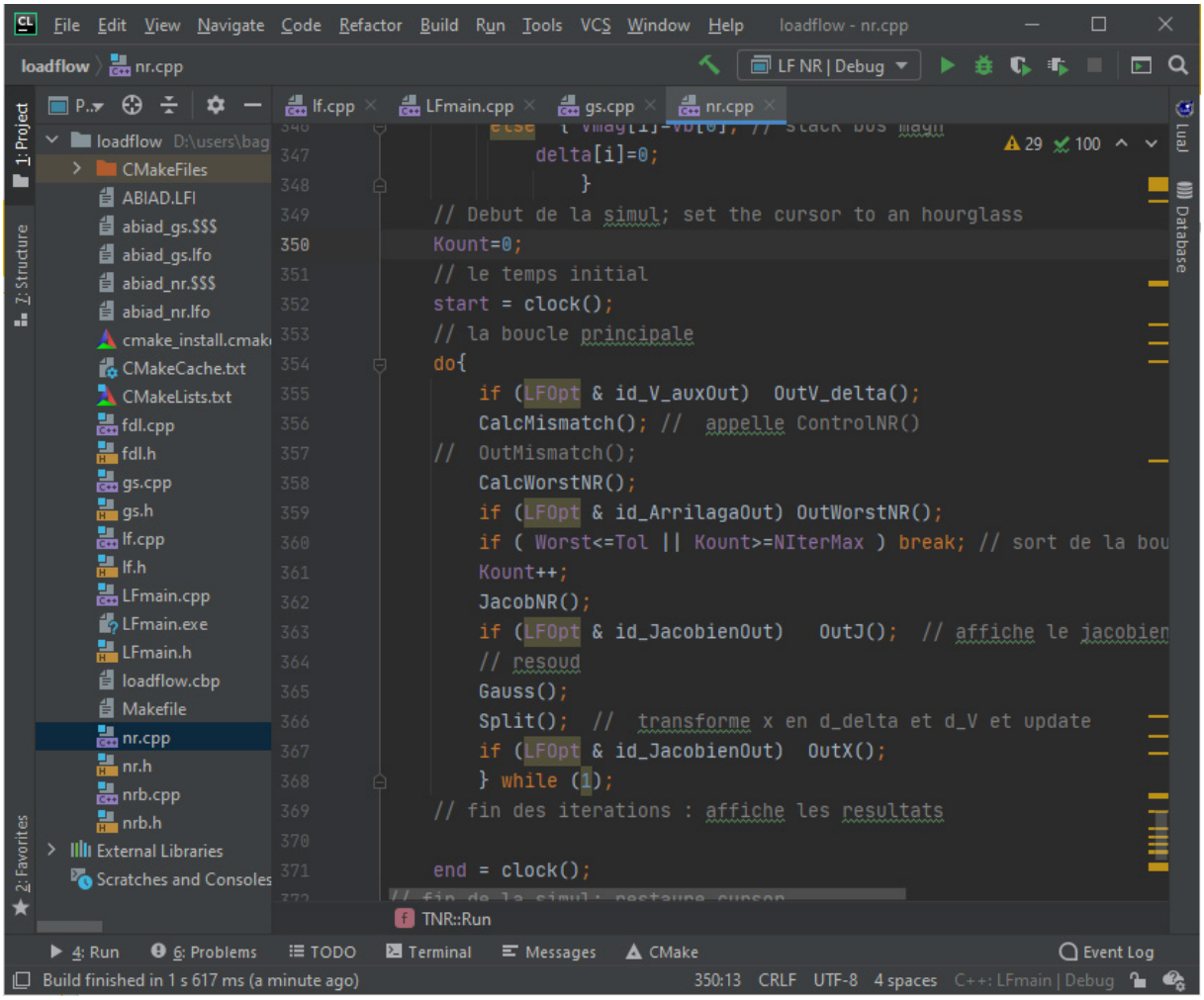
```
      x[i]/=Jc[i][i];
      }
}
```

```
void TNR::Pivot( int k)
{
  int j, L=k+1;
  double M;
  while ( Jc[L][k]==0 && L<2*N ) L++;
  if (L>=2*N && Jc[L][k]==0 ) {
      cout << "Impossible triangularisation : all pivots are null"
        << " : Error in Load Flow Program";
      exit(1);
      }
  M=fpq[k];
  fpq[k]=fpq[L];
  fpq[L]=M;
  for ( j=k; j<=2*N; j++)  {
          M=Jc[k][j];
          Jc[k][j]=Jc[L][j];
          Jc[L][j]=M;
          }
}
```

7.7. Load Flow – NR Application

In C++
Assignment
Using the C++ program provided in the Zip file that was already decompressed of old one?

- Compile and run with the profile LF NR
- Compute the load flow
- Analyze the input file abiad.lfi
- Analyze the output file abiad_nr.lfo
- Compare the results with the ones of Gauss-Seidel algorithm



Sometimes, we use the load flow method called Gauss-Seidel in order to start the first iterations and get a littile bit close to the solution, then we switch to Newton-Raphson to find the solution with less error.
If we start with N-R directly and if the problem is ill conditioned, it may diverge.

7.8. DC Load Flow method

8.14 THE DC POWER FLOW METHOD

In certain power system studies (e.g., *reliability studies*), a very large number of power flow runs may be needed. Therefore, a very fast (and not necessarily accurate, due to the linear approximation involved) method can be used for such studies. The method of calculating the real flows by solving first for the bus angles is known as the *dc power flow method*, in contrast with the exact nonlinear solution, which is known as the *ac solution*.

Assume that bus i is connected to bus j over an impedance of Z_{ij}. Thus, the active power flow can be expressed as

$$P_{ij} = \frac{V_i V_j}{Z_{ij}} \sin(\delta_i - \delta_j) \tag{8.166}$$

where

$$\mathbf{V}_i = |\mathbf{V}_i| \angle \delta_i \quad \mathbf{V}_j = |\mathbf{V}_j| \angle \delta_j$$

The following simplifying approximations are made:

$$X_{ij} \cong \mathbf{Z}_{ij} \text{ since } X_{ij} \gg R_{ij}$$

$$|\mathbf{V}_i| \cong 1.0 \text{ pu}$$

$$|\mathbf{V}_j| \cong 1.0 \text{ pu}$$

$$\sin(\delta_i - \delta_j) \cong \delta_i - \delta_j$$

Hence, the active power flow Equation 8.166 can be expressed as

$$P_{ij} \cong \frac{\delta_i - \delta i}{X_{ij}} \cong B_{ij}(\delta_i - \delta_j) \tag{8.167}$$

Thus, in matrix form,

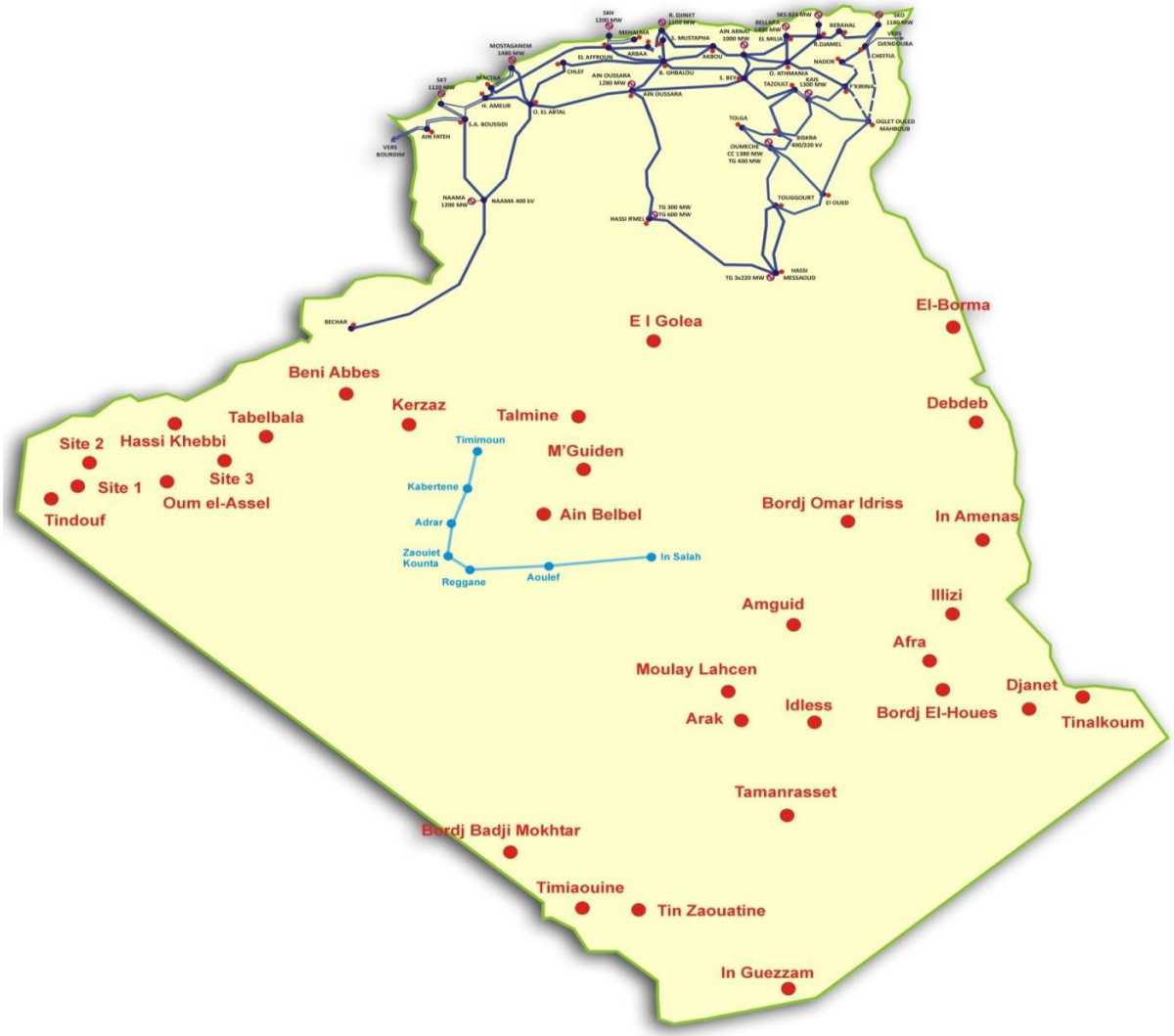
$$[P] = [B][\delta] \tag{8.168}$$

from which

$$[\delta] = [B]^{-1}[P] \tag{8.169}$$

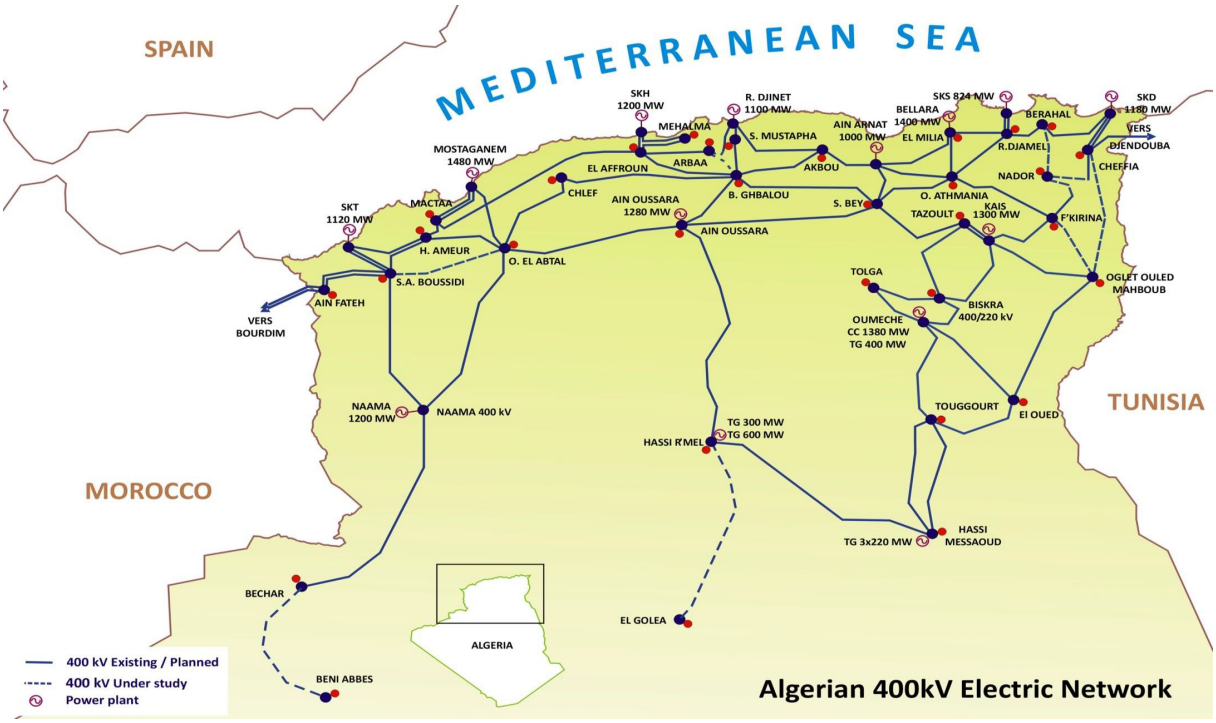
7.9. Optimal Load Flow

c.f. constraints, costs TSO (losses, maintenance), Production costs, distribution not considered.
Run a modified load flow with cost function and constrains. Can be coupled to metaheuristic algorithms.



Réseau algérien d'électricité (2016)

Taux d'électrification: 99%
Réseaux Isolés : 786 MW
Réseau Interconnecté : 17 477 MW
Réseau région Adrar: 719 MW
Capacité installée de production
Réseaux Isolés : 1.0 TWh
Réseau Interconnecté : 64.1TWh
Réseau région Adrar: 1.1 TWh



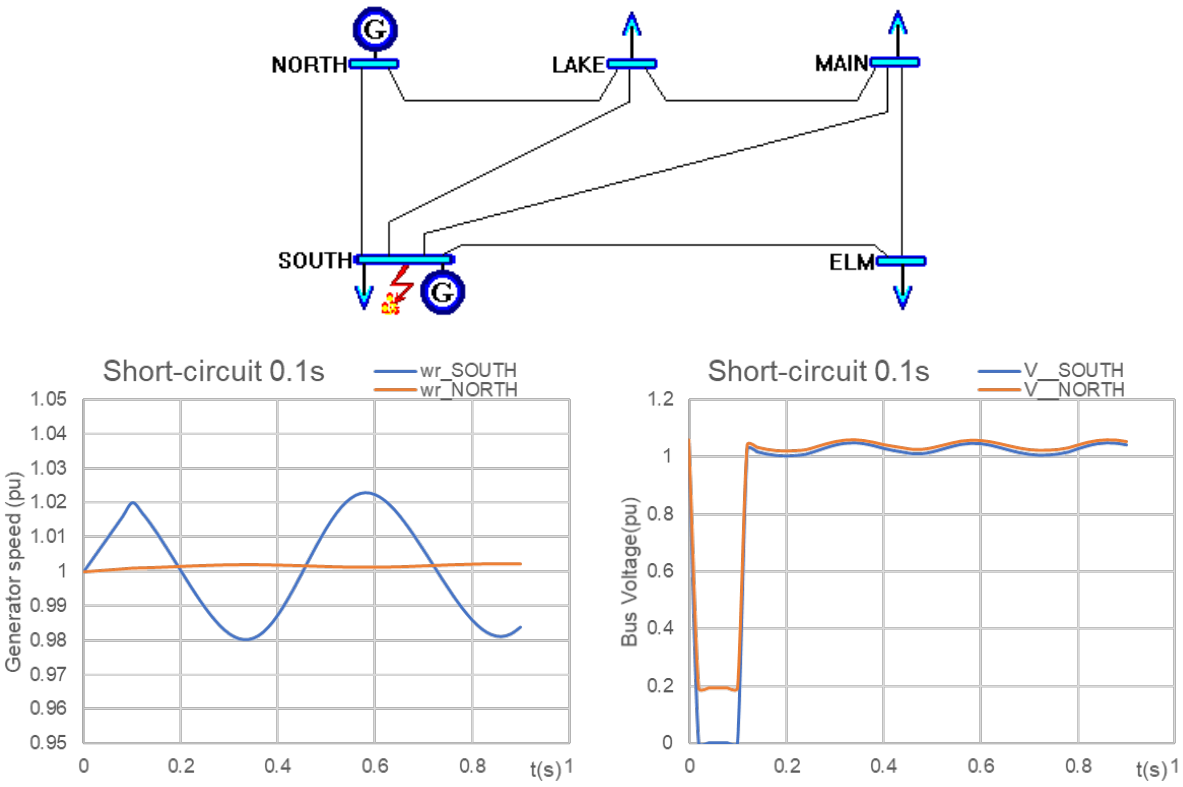
400 kV Energy transportation system

Problem statement

7.10. Summary

Learn how we modelized a power grid to form the admittance matrix Y
Learn to represent the power grid network with single-line connections between buses
Present the algorithms of Gauss-Seidel and Newton-Raphson adapted to Load Flow computation
Simulation of Load Flow are requested to be done using CLion and the program code provided

8. Transient stability of a power grid

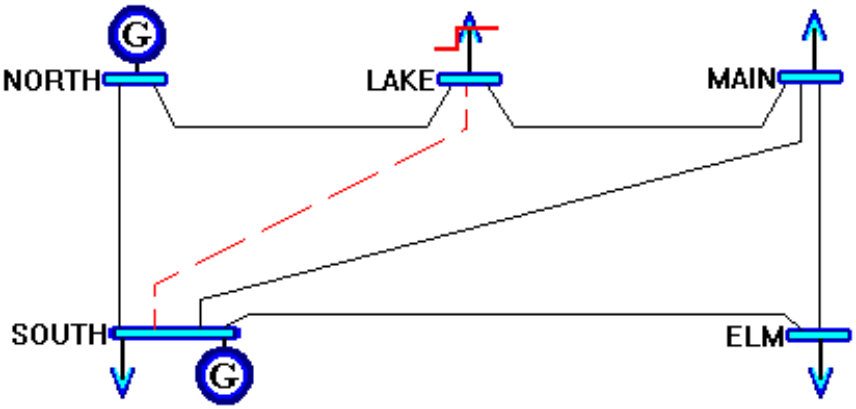


Learn how to use C++ simulation programs to evaluate the 1st swing stability criterion on a power grid
See the application of numerical integration of the generator differential equations
Compile and simulate by yourself in C++ with the given program

References
C++ Tutorials and help http://www.cplusplus.com/doc/tutorial/control/
J. Arrillaga, C. P. Arnold , Computer Analysis of Power Systems, John Wiley & Sons, 1990, http://eword.yolasite.com/resources/51621752-Computer-Analysis-of-Power-Systems.pdf
Stagg, G.W. and El-Abiad, A.h., “Computer Methods in Power System Analysis,” Mc Graw Hill, 1968
L. Baghli, G. Didier, S. Bendali, J. Leveque, An Open Source Real-Time Power System Simulator with HIL, ICEN 2010, Sidi Bel Abbes, 28-29 October, 2010.
G. Andersson, Modelling and Analysis of Electric Power Systems, ETHZ, 2008

8.1. Introduction

With the massive introduction of micro and big renewable energy power plants, the stability of the power grid is more and more fragile
The sudden loss or gain of energy production due to bursts of wind or to clouds that affect a massive territory can have a big impact on the stability of a power grid
We will show how to modify the previous LF program to implement the numerical integration of differential equations coupled to Gauss-Seidel load flow algorithm
The program is implemented in C++ and are given to study the way we can use them to solve



Fault example
Sudden line loss and load increase

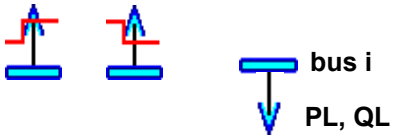
8.2. Power grid elements

Short circuit

We can apply a short circuit with or without impedance to any bus and clear it after a reaction time

Load

The load is modelized by an impedance which consumes a constant active power PL and reactive power QL. In TS study, the load can suddenly increase or decrease, representing a PV power plant that lose or gain production, a town that is disconnected from the grid,…



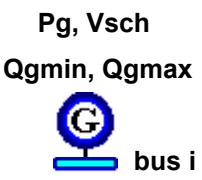
Line

The lines can be opened (disconnected) or reconnected due to the breakers at each end of the long high voltage line.
We can observe the behaviour of the grid according to that fault

Generator

A generator model is more complex in TS than in LF and introduces the mechanical differential equation. We can simulate a loss of a power plant

$J\frac{d\omega_m}{dt} = J\frac{d^2\theta_m}{dt^2} = J\frac{d^2\delta_m}{dt^2} = C_m - C_e$ with:



$\theta_m = \omega_{sm}t + \delta_m \theta = \omega_s t + \delta$

Generator

$\theta = p\theta_m$ and $\delta = p\delta_m$

$J\omega_{sm}\frac{d^2\delta_m}{dt^2} = P_m - P_e$

with $\omega_m \approx \omega_{sm}$

$\frac{1}{S_b}\left(\frac{1}{2}J\omega_{sm}^2\right)\frac{2}{\omega_{sm}}\frac{d^2\delta_m}{dt^2} = \frac{P_m - P_e}{S_b}$ (W)

We define $H = \frac{1}{S_b}\left(\frac{1}{2}J\omega_{sm}^2\right)$ the ratio of the kinetic energy stored at synchronous speed over the base power in MVA.

The mechanical differential equation

$\frac{2H}{\omega_{sm}}\frac{d^2\delta_m}{dt^2} = P_m - P_e$ (p.u.)

As ω_{sm} and δ_m are mechanical quantities, we obtain:

$\frac{2H}{\omega_s}\frac{d^2\delta}{dt^2} = P_m - P_e$ (p.u.)

If we include the mechanical damping, the equation becomes

$\frac{2H}{\omega_s}\frac{d^2\delta}{dt^2} = P_m - P_e - D\frac{d\delta}{dt}$ (p.u.)

1 second order differential equation can be put a 2 first order differential equations

The system to solve for each generator is

$$\begin{cases} \frac{d\delta}{dt} = \omega - \omega_s \\ \frac{d^2\delta}{dt^2} = \frac{\omega_s}{2H}\left(P_m - P_e - D\frac{d\delta}{dt}\right) \end{cases}$$

The electrical quantities related to the generator are

$E_g = V_t + \frac{I_g}{y_g}$

with $y_g = \frac{1}{R_a + jX'_d}$ and $I_g = \left(\frac{S_g}{V_t}\right)^*$

$P_e = Re(E_g I_g^*)$

This power equals the mechanical power P_m of the turbine in a steady state operation

L. Baghli, G. Didier, S. Bendali, J. Leveque, An Open Source Real-Time Power System Simulator with HIL, ICEN 2010, Sidi Bel Abbes, 28-29 October, 2010.

6.3 Transient stability study

First swing stability

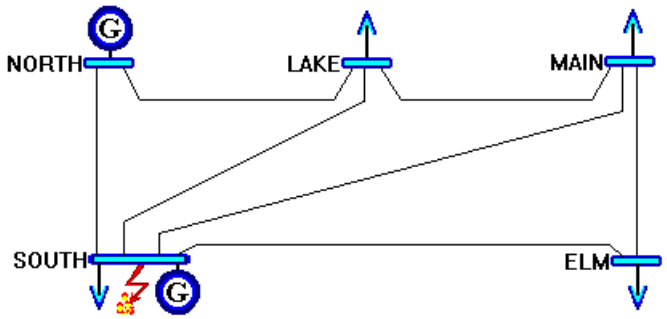
- The study is based on a simple generator model without the control systems (PSS)
- We only consider the first second after the disturbance. If the machines remain stable after this time, the power system is considered stable for this disturbance

Multi-swing stability

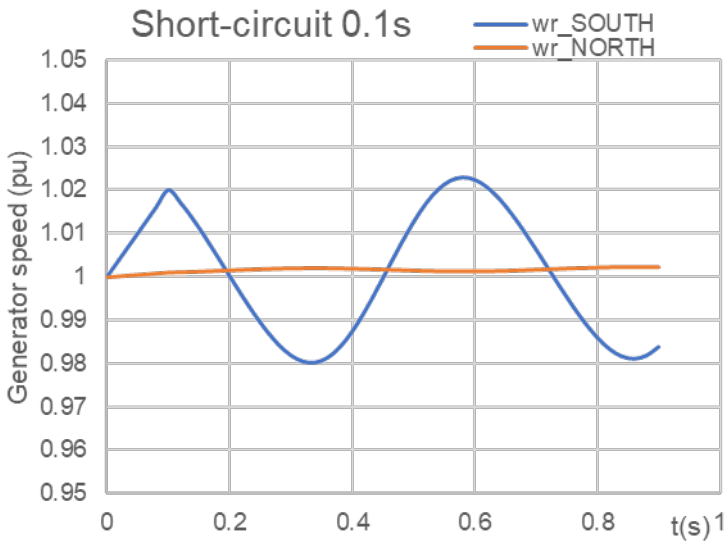
- The study extends for several seconds after the disturbance
- It requires more rigorous models and consideration of control systems to reproduce as closely as possible the behavior of the machines in the system

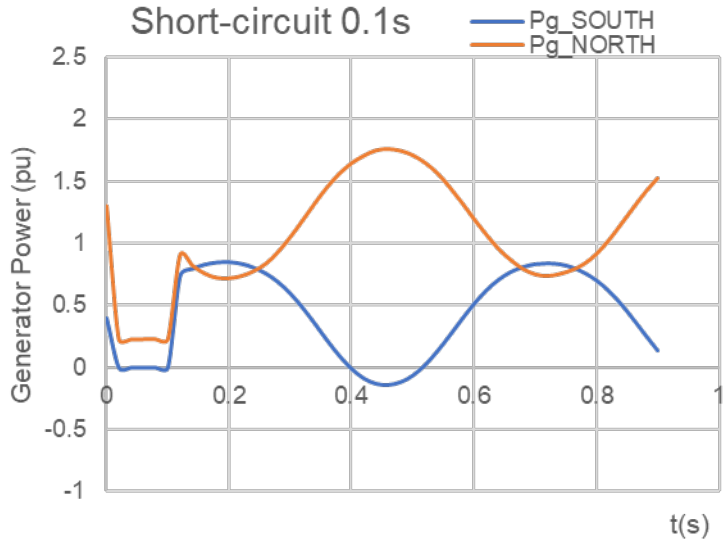
First swing stability

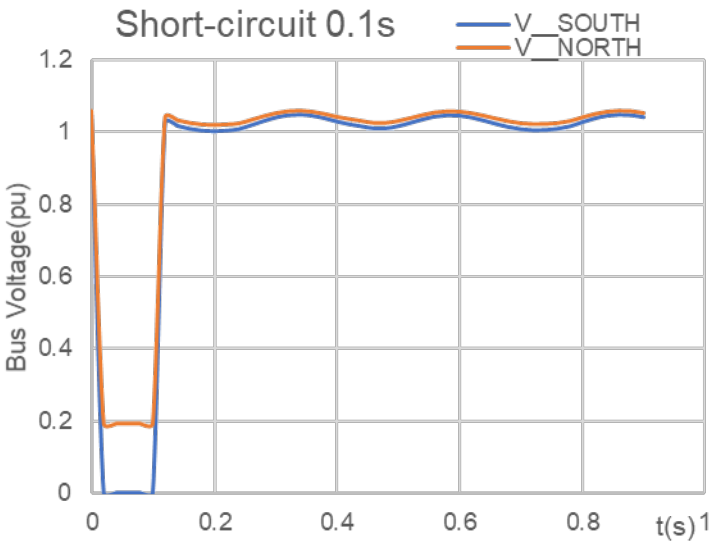
- Short-circuit fault at bus South, duration 0.1s
- Fault is cleared by protection systems after 0.1s
- The power grid is considered stable for this disturbance



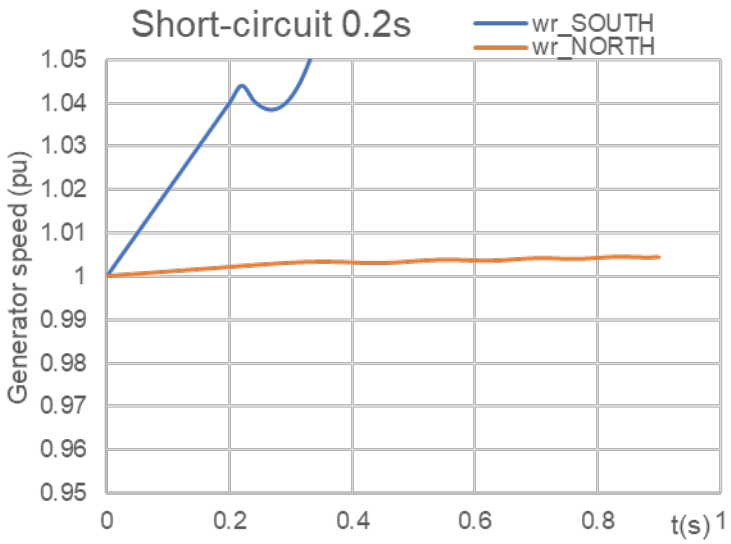
5-bus power grid with a short-circuit fault at bus South

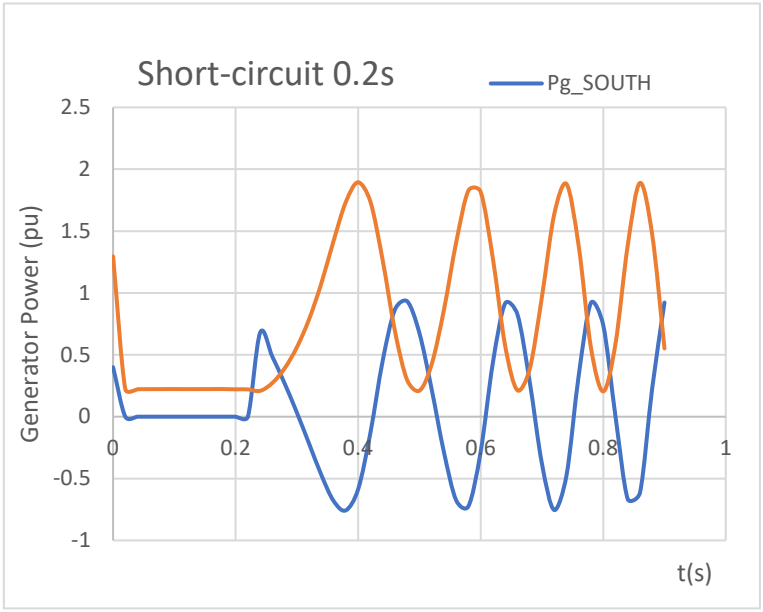

Short-circuit 0.1s
wr_SOUTH
wr_NORTH
Generator speed (pu)
1.05
1.04
1.03
1.02
1.01
1
0.99
0.98
0.97
0.96
0.95
0
0.2
0.4
0.6
0.8
1
t(s)

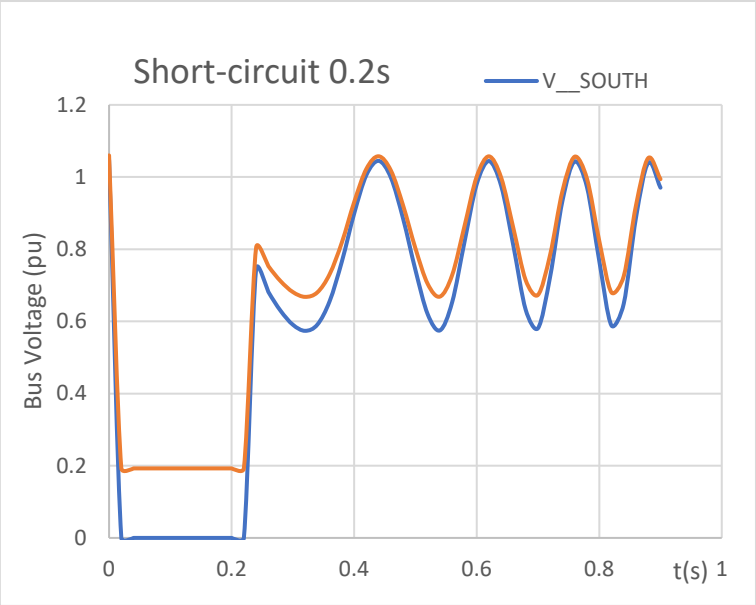

Short-circuit 0.1s
Pg_SOUTH
Pg_NORTH
Generator Power (pu)
2.5
2
1.5
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1
t(s)


Short-circuit 0.1s
V_SOUTH
V_NORTH
Bus Voltage(pu)
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
t(s)

Short-circuit fault at bus South, duration 0.2s, System unstable


Short-circuit 0.2s
wr_SOUTH
wr_NORTH
Generator speed (pu)
1.05
1.04
1.03
1.02
1.01
1
0.99
0.98
0.97
0.96
0.95
0
0.2
0.4
0.6
0.8
1
t(s)


Short-circuit 0.2s
Pg_SOUTH
Generator Power (pu)
2.5
2
1.5
1
0.5
0
-0.5
-1
0
0.2
0.4
0.6
0.8
1
t(s)



8.3. Simulation algorithm

The computation methods used for transient stability studies are step by step time integration of the differential equation of the state system of the generators.
We implemented modified Euler, trapezoidal and Runge Kutta 4 integration methods
At each sampling step, we also perform a G-S load flow computation in order to get the new power distribution and bus voltage
The state vector $[X_v]$ of the multi-machine system is:

$$X_v = [\delta_0 \quad \delta_1 \quad \cdot \quad \delta_{N_{PV}} \quad \omega_0 \quad \omega_1 \quad \cdot \quad \omega_{N_{PV}}]^T$$

Each couple of variable $(X_v[i],\ X_v[N_{PV}+i])$ represents the state variables previously introduced in generator modelling
At the beginning of the transient stability study (at t=0), the state vector is initialized thanks to a load flow computation with:

$$X_v[i] = arg\big(E_g[i]\big)$$

$X_v[N_{PV}+i] = \omega_s$ (i taking values from 0 to the number of PV buses N_{PV})
We include the slack bus, which has a large generator as well
We also initialize the mechanical power that counter-balances the electrical power at the latter in steady state operation, before the fault occurrence

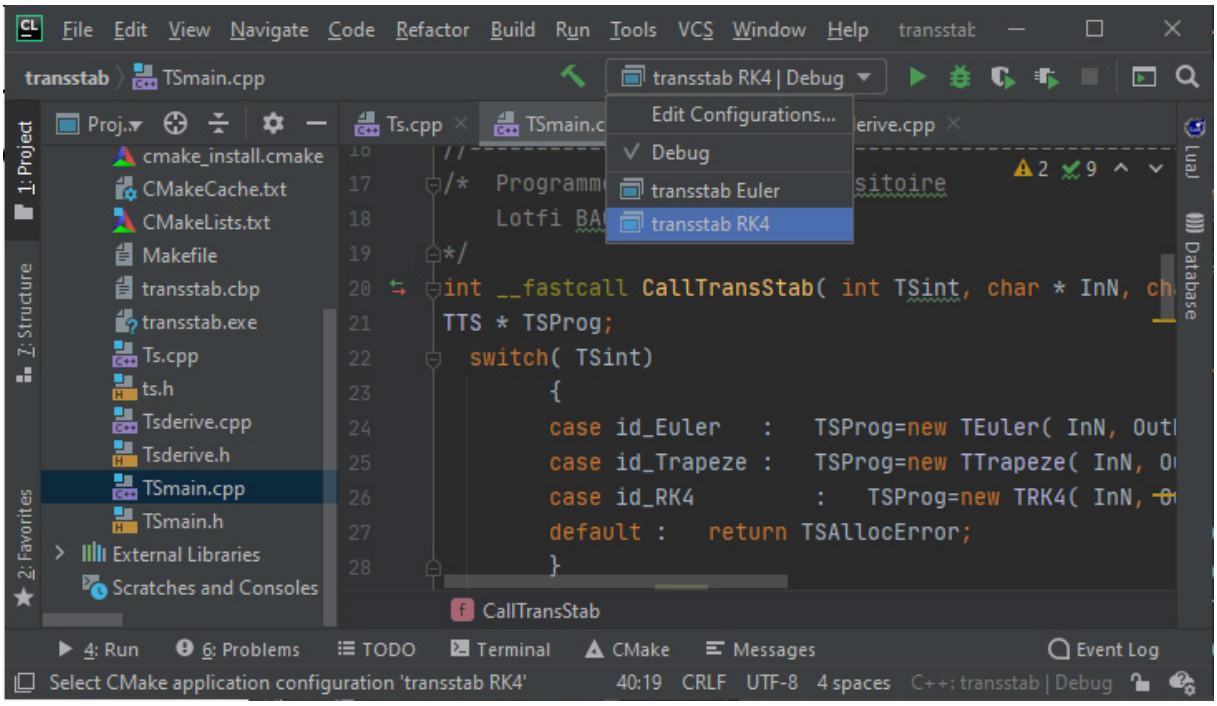
8.4. Transient stability – Application

In C++
Assignment
Using the C++ program u6_transstab.zip provided in the zip file
Delete cmake_install.cmake
Delete CMakeCache.txt
Reload CMake Project on CMakeLists.txt

- Compile and run with the profile transstab RK4
- Simulate the Transient Stability Analysis

- Analyze the input file abiad.tsi
- Analyze the output file abiad_rk4.tso
- Display the plot output file abiad_rk4.tsg using the software provided Graph.exe
  Command line: graph.exe abiad_rk4.tsg
- Change the short circuit clear time from 0.1s to 0.2s
- Observe and compare with the previous slides
- Apply a sudden increase in the load of bus ELM of 0.5 pu

```
/*  Transient stability
   Lotfi BAGHLI 06/2020
*/
int __fastcall CallTransStab( int TSint, char * InN, char * OutN, char * GraphN) {
TTS * TSProg;
  switch( TSint)
      {
      case id_Euler   :  TSProg=new TEuler( InN, OutN, GraphN ); break;
      case id_Trapeze :  TSProg=new TTrapeze( InN, OutN, GraphN ); break;
      case id_RK4       :  TSProg=new TRK4( InN, OutN, GraphN ); break;
      default :  return TSAllocError;
      }
```



Main Run function

After reading the input data file .tsi and the initialization and computation of the Y admittance matrix

We compute the mechanical power P_m and e.m.f E_g of the generators

All faults are cleared

A do-while loop is started and will finish when the time is over
At each step, we verify if a new event arises: A fault start or a fault clearance
If a new event occurs, a flag `WillModifyY`, is set in order to modify the Y admittance matrix

```
void TTS::Run()
{
t=0;
int WillModifyY;
if (TSOpt & id_YMatrixOut) OutY();
    CalcYload(); // Calcule les admitances de charge equivalentes
    if (TSOpt & id_YMatrixOut) OutYg();
    if (TSOpt & id_YMatrixOut) OutYL();
    ModifY(); // Modifie Y en incluant les yL et yg
    OutModif(); // affiche message puis Y
    if (TSOpt & id_YMatrixOut) OutY();
    SaveY(); // appelee apres ModifY pour avoir yL, yg dans les elements Ysave
    CalcInit(); // Compute Eg, Pm and Set initial State Variables Xv
    OutGraphInit(); // entete du GraphFile
    FBFlag=0; // Clear Fault Flags
    LOFlag=0;
    SLFlag=0;
    DLFlag=0;
    LGFlag=0;
    OutXvGraph(); // sort donnees pour graph
    if (TSOpt & id_MachineOut)  OutEgPgVmag();
    start = clock();
    do{    // while ( t<=t2 );
        WillModifyY=0;
        // FaultedBus
        if ( t>=FBt0 && FBFlag==0 && ((FaultType & id_FaultedBus)==id_FaultedBus))
        { FBFlag=1;
            WillModifyY=1; }
        if ( t>=FBt1 && FBFlag==1 ) { FBFlag=2;
            WillModifyY=1; }
        // OpenedLine
        if ( t>=LOt0 && LOFlag==0 && ((FaultType & id_OpenedLine)==id_OpenedLine))
        { LOFlag=1;
            WillModifyY=1; }
        if ( t>=LOt1 && LOFlag==1 ) { LOFlag=2;
            WillModifyY=1; }
        // SwitchedLoad
        if ( t>=SLt0 && SLFlag==0 && ((FaultType & id_SwitchedLoad)==id_SwitchedLoad))
        { SLFlag=1;
            WillModifyY=1; }
        if ( t>=SLt1 && SLFlag==1 ) { SLFlag=2;
            WillModifyY=1; }
```

Main Run function (next)

After all events are checked, if the flag `WillModifyY` is set, the Y admittance matrix is modified and LF and power calculus is done
An integration of the numerical differential equation is done corresponding to the numerical method chosen
We output a line on the graph destination file .tsg
The loop is terminated and the simulation ends when the time is over

```
        // DroppedLoad
        if ( t>=DLt0 && DLFlag==0 && ((FaultType & id_DroppedLoad)==id_DroppedLoad))
        { DLFlag=1;
            WillModifyY=1; }
        if ( t>=DLt1 && DLFlag==1 ) { DLFlag=2;
            WillModifyY=1; }
        // LossGen
        if ( t>=LGt0 && LGFlag==0 && ((FaultType & id_LossGen)==id_LossGen))
        { LGFlag=1;
            WillModifyY=1; }
        if ( t>=LGt1 && LGFlag==1 ) { LGFlag=2;
            WillModifyY=1; }

        if ( WillModifyY==1)   {
            ModifY_Default();
            OutModif();
            if (TSOpt & id_YMatrixOut) OutY();
            // Recalcule
            LFGauss();
            CalcPg();
        }

        // Routine d'integration numerique
        IntegreSysteme();

        OutXvGraph(); // sort donnees pour graph
        // et pour le .TSO
        if (TSOpt & id_MachineOut)  OutEgPgVmag();
        if (TSOpt & id_V_auxOut)  OutV_delta();

        // reboucle
    } while ( t<=t2 );
    end = clock();
    OutFile <<"Total time : "<< (end - start)/CLK_TCK << " s\n";
    OutFile.close();
    GraphFile.close();
}
```

The result file is a table or sheet that we can use to display curves vs time

```
time    del_SOUTH       wr_SOUTH        V__SOUTH        Pg_SOUTH        del_NORTH       wr_N
0       18.3823         1               1.04747         0.4             16.3399         1
0.02    19.2463         1.004           3.29446e-20     0               16.3863         1.00
0.04    21.8383         1.008           3.29172e-20     0               16.5255         1.00
...
0.16    59.6928         1.00919         1.00646         0.831615        19.0527         1.00
0.18    62.7288         1.00473         1.00123         0.849949        19.6587         1.00
0.2     63.7987         1.00017         0.999336        0.851023        20.315          1.00
0.22    62.8977         0.995672        1.00144         0.838312        21.0227         1.00
0.24    60.0813         0.991393        1.00741         0.80882         21.7804         1.00
...
0.34    24.9023         0.980195        1.04772         0.34991         26.1173         1.00
0.36    16.4549         0.981387        1.0442          0.214477        27.0254         1.00
0.38    8.81482         0.983914        1.0367          0.0881866       27.9198         1.00
0.4     2.53911         0.987625        1.0275          -0.0173875      28.7908         1.00
```

Integration of the numerical differential

By C++ Classes virtual methods, the correct integration method is called from the descendent (`TEuler`, `TTrapeze` or `TRK4`) of the class `TTS`

```
void TEuler::IntegreSysteme()
{
  EulerPredictor();
  // new Eg                    avec les angle Xvp -----|
  for ( int i=0; i<=Npv; i++)  Eg[i]=polar( abs( Eg[i]), Xvp[i] );
  LFGauss();
  CalcPg(); // new Pg for the corrector

  t+=dt; // incremente time
  // les trapezes mais une fois seulement donc Modified Euler
  ModifiedEulerCorrector();
  // la remise a jour Eg, LFGAuss et CalcPG dans ModifiedEulerCorrector()
}
```

Integration of the numerical differential

For instance, the Modified Euler integrates in 2 steps, a predictor steps followed by a corrector step

Refer to the Numerical Integration unit, for further details

```
void TEuler::EulerPredictor()
{
   for ( int i=0; i<=Npv; i++)
      {
      dXvp[i]=Xv[Npv1+i]-Ws;          //   |-- Damper
      dXvp[Npv1+i]=Ws/2/H[i]*( Pm[i]-Pg[i]-D[i]*dXvp[i]);
      Xvp[i]=Xv[i]+dXvp[i]*dt;
      Xvp[Npv1+i]=Xv[Npv1+i]+dXvp[Npv1+i]*dt;
      }
}
```

```
void TEuler::ModifiedEulerCorrector()
{
complex<double> Ig;
int i;
   for ( i=0; i<=Npv; i++)
      {
          dXvc[i]=Xvp[Npv1+i]-Ws;          //   |--Damper
          dXvc[Npv1+i]=Ws/2/H[i]*( Pm[i]-Pg[i]-D[i]*dXvc[i]);  // the new Pg
          Xv[i]+=( dXvp[i]+dXvc[i] )*dt/2;
          Xv[Npv1+i]+=( dXvp[Npv1+i]+dXvc[Npv1+i] )*dt/2;
          Eg[i]=polar( abs( Eg[i]), Xv[i] );
      }
   LFGauss();
CalcPg();
}
```

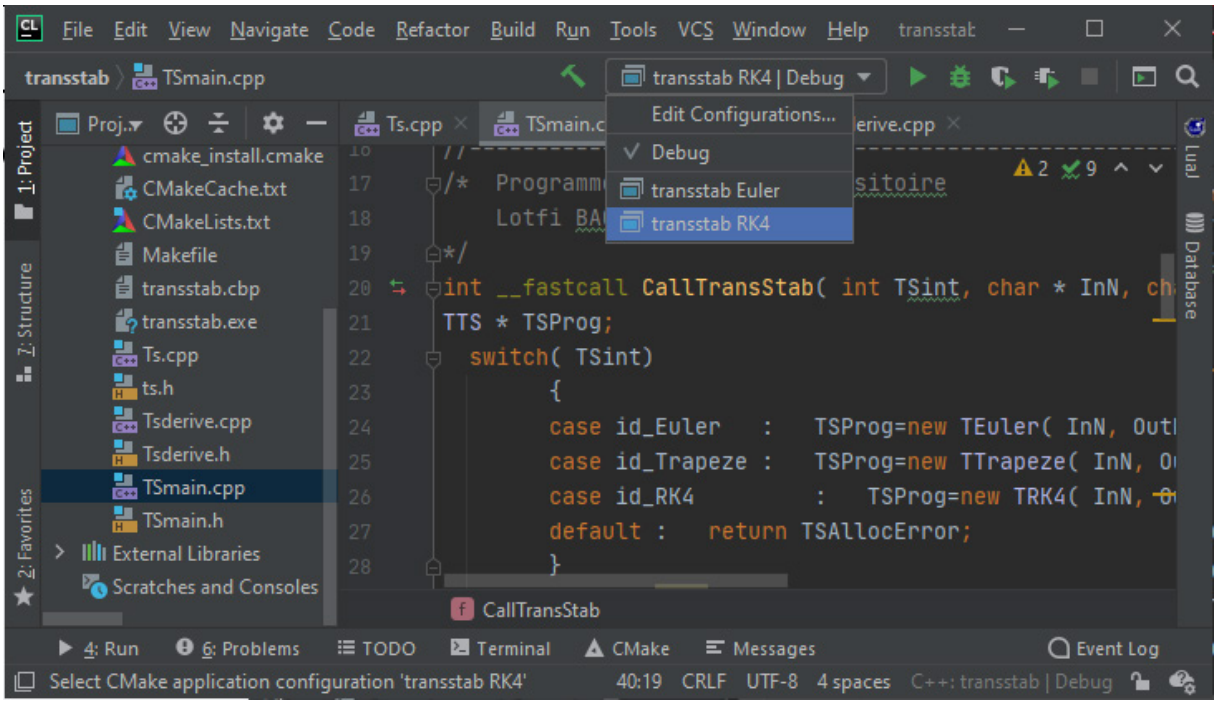
In C++

Assignment

Using the C++ program given

- Compile and run with the other profile transstab Euler using the original input file and specified fault (S/C of 0.1s on bus South)
- Compare the results abiad_e.tso and abiad_e.tsg with the previous ones abiad_rk4.tso and abiad_rk4.tsg and
- Display the plot output files abiad_e.tsg and abiad_rk4.tsg using the software provided Graph.exe

  Command line: graph.exe abiad_rk4.tsg

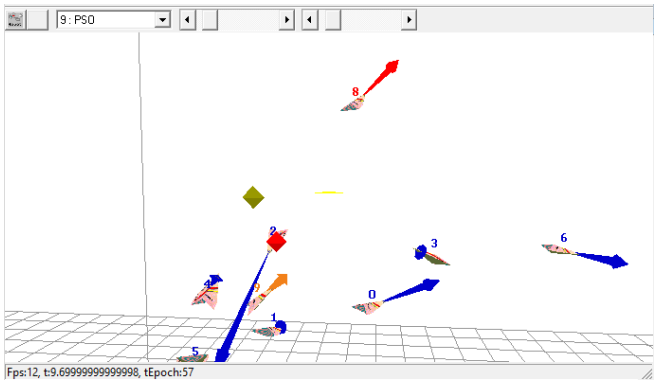


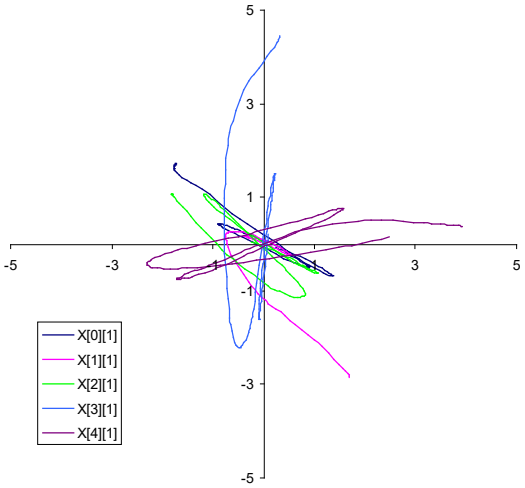
8.5. Summary

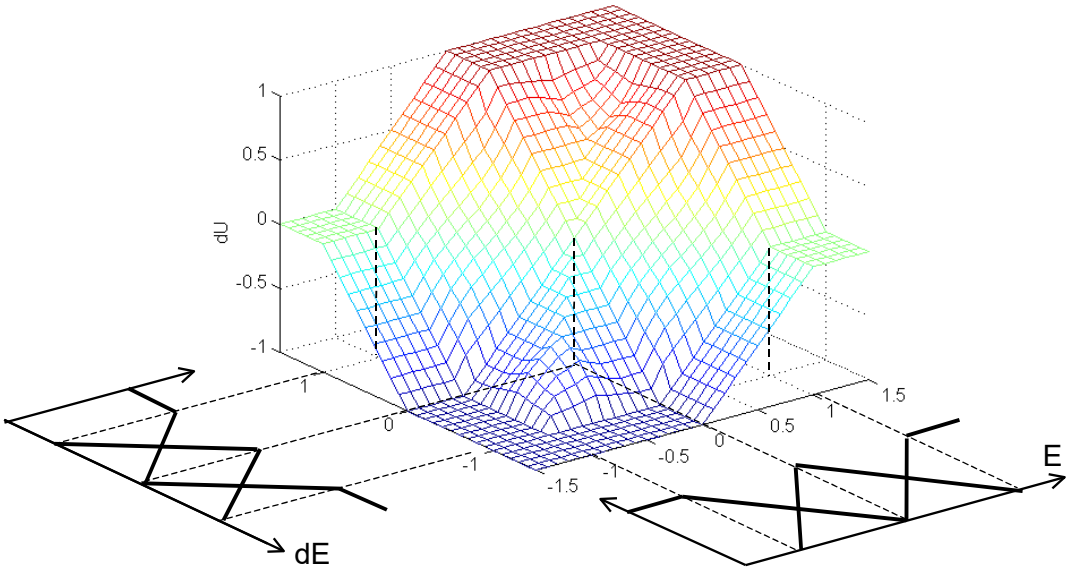
For the first swing stability study, we only consider the first second after the disturbance
If the machines remain stable after this time, the power system is considered stable for this disturbance
The numerical integration method allows to simulate the behavior of the system through the time.
Command line programs produces text files of results that are used to produces graphs

9. Artificial Intelligence (AI) and metaheuristics

Artificial Intelligence (AI) and metaheuristics







Learn some methods of AI used in optimization and security

- Fuzzy logic
- Neural Networks
- Genetic Algorithms
- Particle Swarm Optimization (PSO)

Application

- Fourier analysis using Particle Swarm Optimization (PSO) in C / MATLAB script

References

Metaheuristics, https://en.wikipedia.org/wiki/Metaheuristic

Fuzzy logic and Neural Networks,
L. BAGHLI, H. RAZIK, "Nouvelles technologies de commande", in "Commande des systèmes électriques : perspectives technologiques", Editor L. LORON, 414p, Oct. 2003, Hermes.
ISBN : 2-7462-0735-4.
https://www.lavoisier.fr/livre/electricite-electronique/commande-des-systemes-electriques-perspectives-technologiques/loron/descriptif-9782746207356

Kennedy, J.; Eberhart, R.; "Particle swarm optimization" in *1995 IEEE International Conference on Neural Networks Proceedings (Cat. No.95CH35828), 1995, V4, pp 1942-1948*

Kennedy, J.; Spears, W. M.; "Matching algorithms to problems : an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator" in *Proceedings of the 1998 International Conference on Evolutionary Computation, pp 78-83*

L. BAGHLI, A. REZZOUG, "Particle Swarm and Genetic Algorithms applied to the identification of Induction Machine Parameters",
EPE'03, 2-4 September 2003, 768.pdf pp.1-10, Toulouse, France.
ISBN : 90-75815-07-7, http://baghli.com/dl/baghli_EPE2003.pdf

https://www.researchgate.net/publication/222686244_Harmonic_elimination_in_diode-clamped_multilevel_inverter_using_evolutionary_algorithms

https://www.researchgate.net/publication/229036332_Particle_Swarm_and_Genetic_Algorithms_applied_to_the_identification_of_Induction_Machine_Parameters

https://youtu.be/uDPy-6DThXw

https://en.wikipedia.org/wiki/Cyclic_redundancy_check

https://www.pwc.com/gx/en/industries/assets/blockchain-technology-in-energy.pdf

Alladi, T.; Chamola, V.; Rodrigues, J.J.P.C.; Kozlov, S.A. Blockchain in Smart Grids: A Review on Different Use Cases. Sensors 2019, 19, 4862, doi:10.3390/s19224862

Christopher Olah, Understanding-LSTMs, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

V. Flovik, How (not) to use Machine Learning for time series forecasting: Avoiding the pitfalls, June 2018, https://towardsdatascience.com/how-not-to-use-machine-learning-for-time-series-forecasting-avoiding-the-pitfalls-19f9d7adf424

9.1. Introduction

Artificial Intelligence

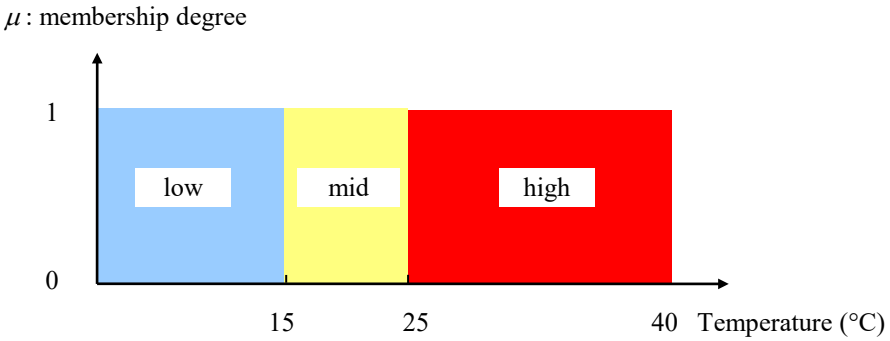
- Everywhere, from the dishwasher to the lane following assist
- Power of embedded microcontrollers increases a lot
- Assisted by huge connected clusters like TensorFlow
- Can be implement online or offline (NN learning process)

Applications

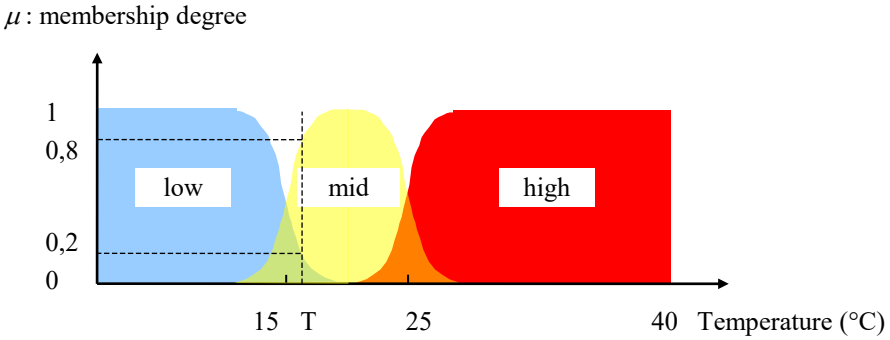
- Optimization
- Find the best solution of a problem given many different constraints
- Mapping of functions or behaviors
- Clustering data
- Big data
- Mimic human behavior or thinking
- Coding transactions within a secured decentralized structure
- …

9.2. Fuzzy Logic

Example of sets in Boolean logic



Example of sets in Fuzzy logic

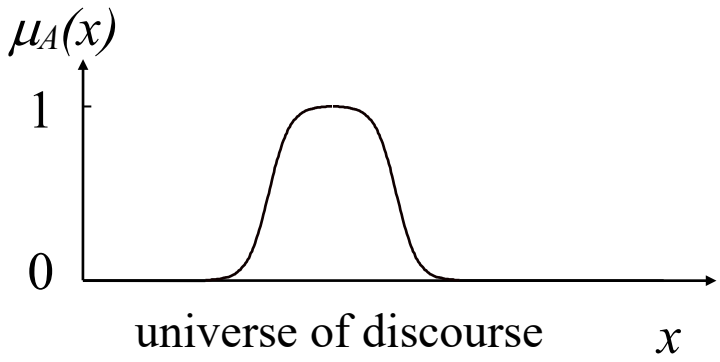


Fuzzy variable

For a fuzzy variable x, we define a fuzzy set A on the universe of discourse X by a membership function

$$\mu_A : X \to [0,1]$$
$$x \mapsto \mu_A(x)$$

The universe of discourse is the set of the real values
that can take the fuzzy variable x and $\mu_A(x)$ is the membership
of the element x to the fuzzy set A



3 steps
Fuzzification: from real world to the fuzzy representation of the input
Inference: Apply the inference rules to compute the output or take the decision, in its fuzzy representation
Defuzzification: from the fuzzy representation of the output to the real world

9.2.1. *Fuzzification*

Compute the membership of the inputs of all its fuzzy sets.

9.2.2. *Inference*

Classical logic
modus ponens
x is *A*
and
if *x* is *A* then *y* is *B*
to conclude that *y* is *B*

Fuzzy logic
generalised modus ponens
x is *A'*
and
if *x* is *A* then *y* is *B*
to conclude that *y* is *B'*

Inference matrix

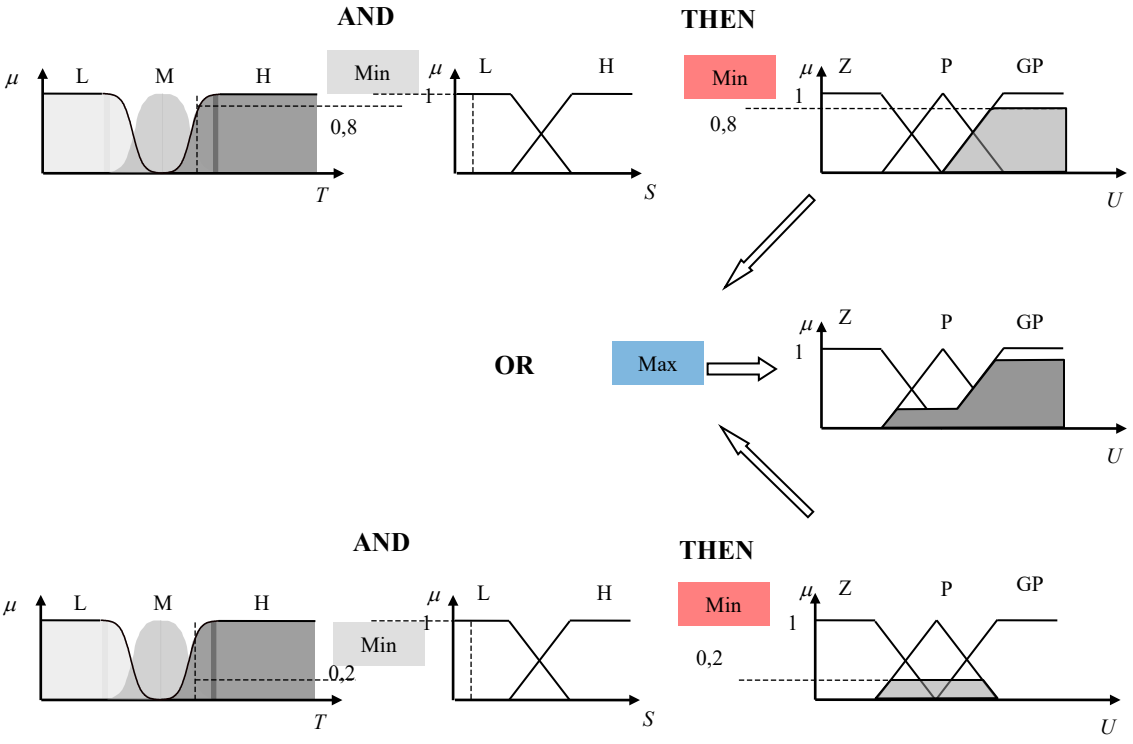
Symbolic form
IF (*T* is L **AND** *S* is L) **THEN** *U*=Z
OR
IF (*T* is M **AND** *S* is L) **THEN** *U*=P
OR
IF (*T* is H **AND** *S* is L) **THEN** *U*=GP

OR

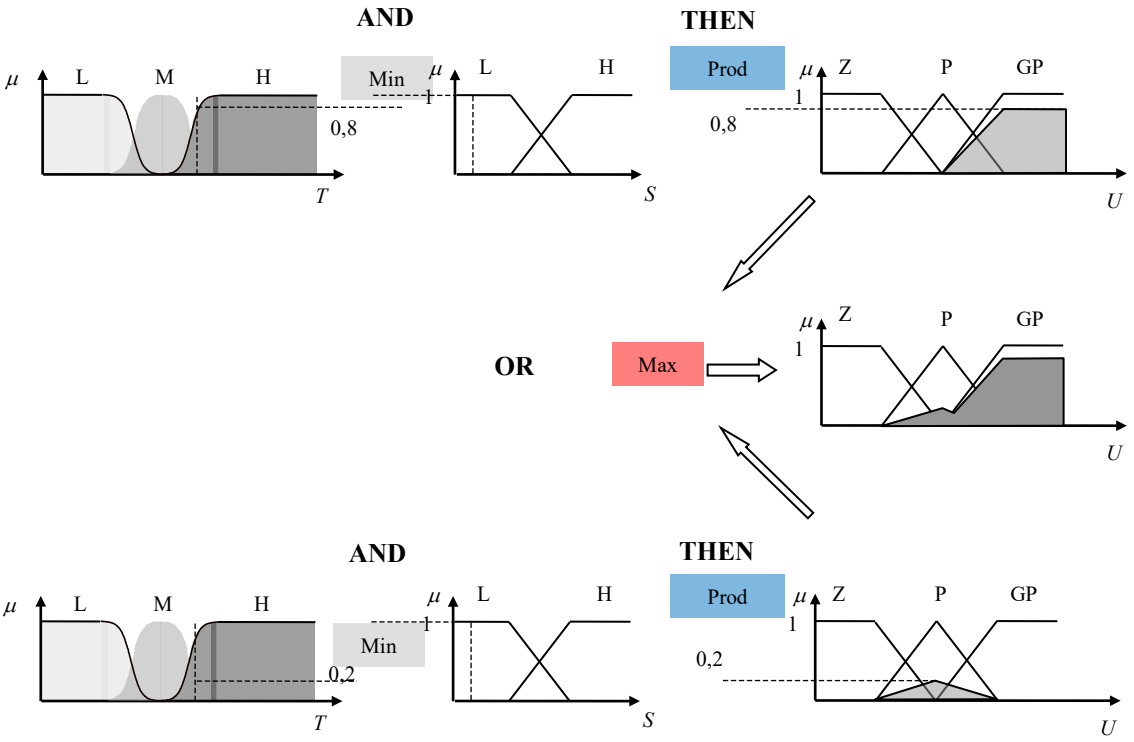
...

U		T		
		L	M	H
S	L	Z	P	GP
	H	Z	Z	P

Inference method Max-Min



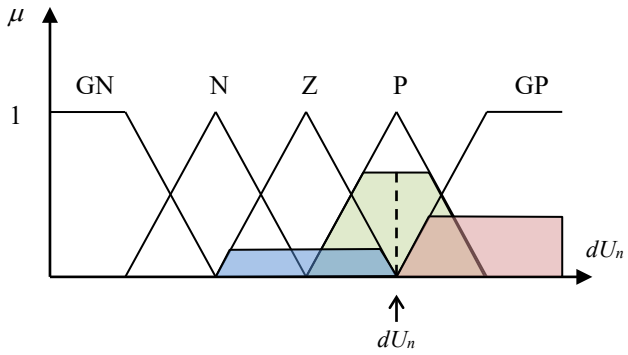
Inference method Max-Prod



9.2.3. *Defuzzification*

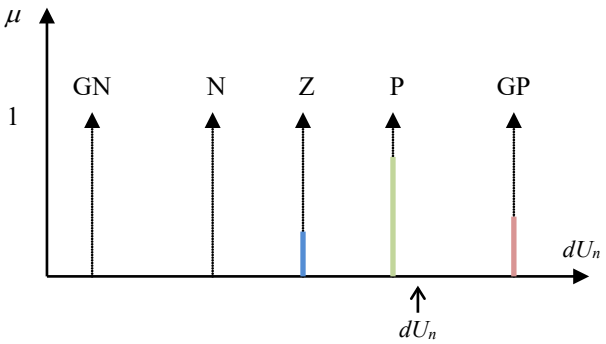
Center of gravity (COG) method

$$dU_n = \frac{\int x\mu_R(x)dx}{\int \mu_R(x)dx}$$



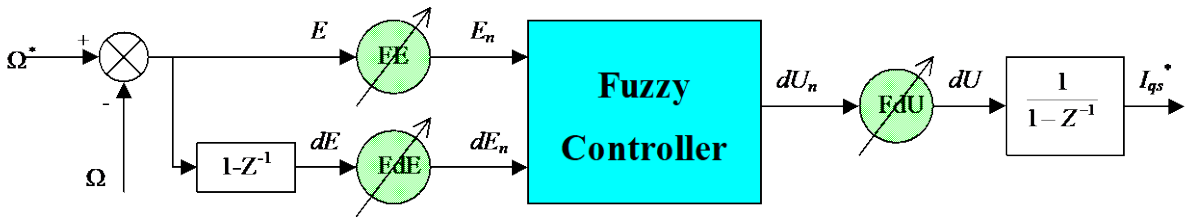
Mean of maximum (MOM) method
Height Method (HM)

$$dU_n = \frac{\sum x\mu_{Ri}(x)}{\sum \mu_{Ri}(x)}$$

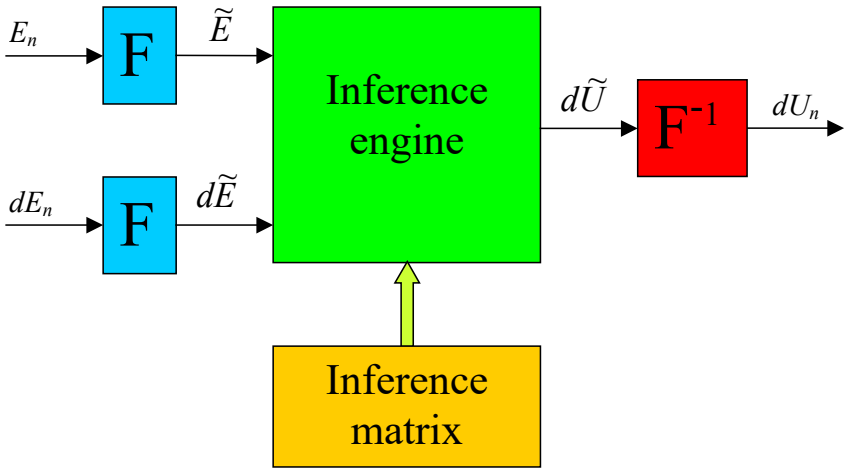


9.3. Fuzzy controller

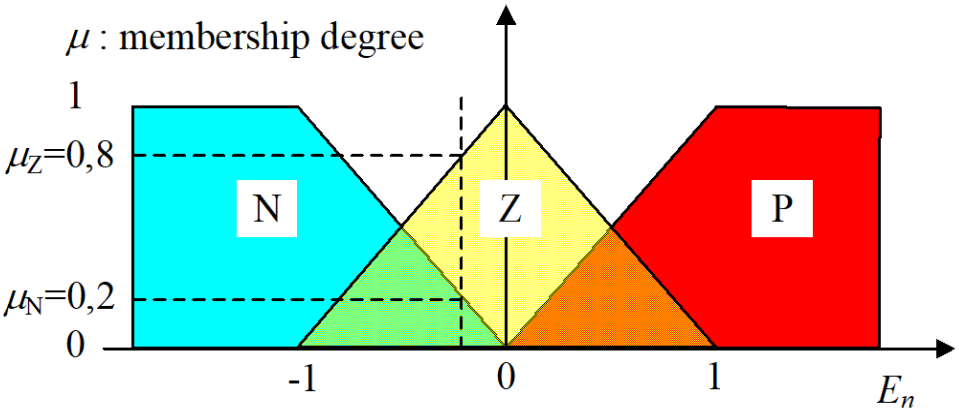
Fuzzification
Inference
Defuzzification



Synoptic scheme of a Fuzzy speed controller

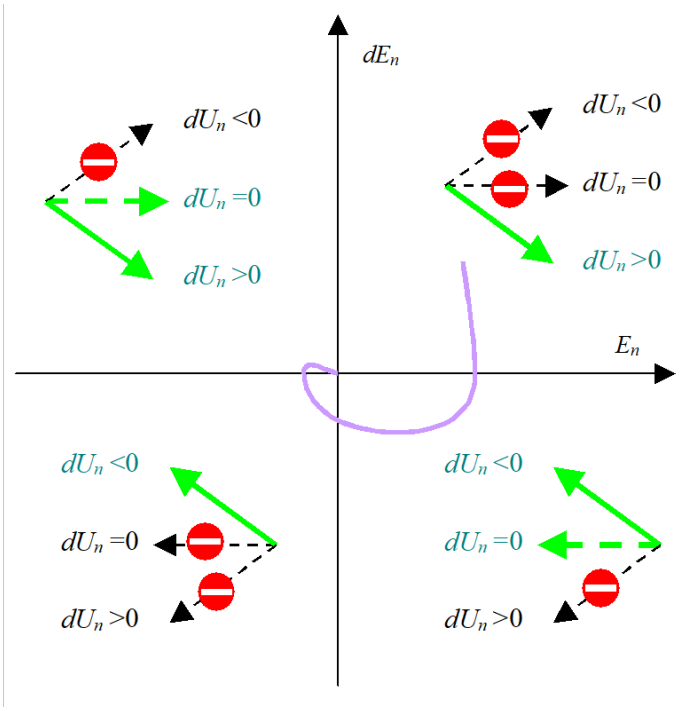


Fuzzy controller
Fuzzification

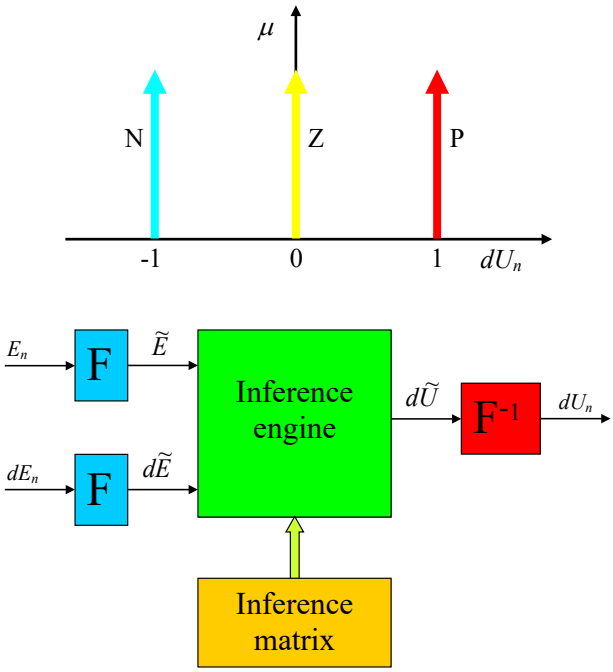


Inference

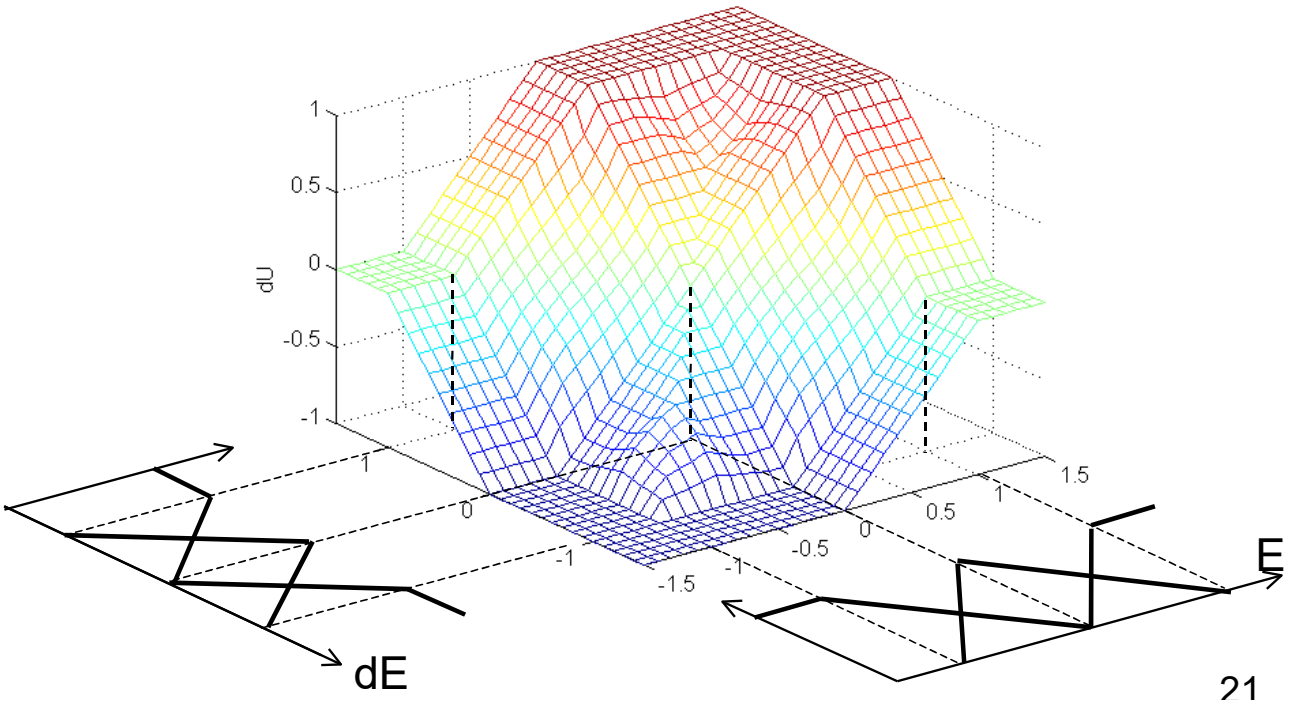
dUn \ dEn	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P



Defuzzification



Normalized control surface



21

9.4. Neural network

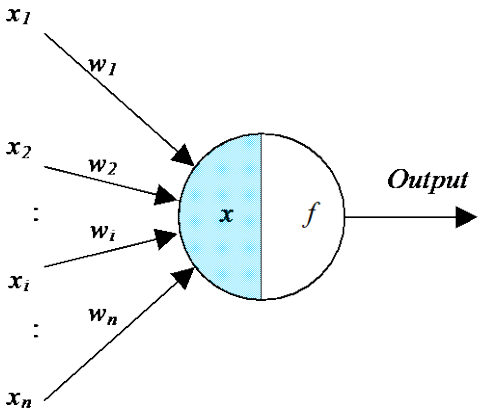
Origin: Theory McCulloch & Pitts
Modellization of a biological neuron
Non-linear adaptive Network
Modify links to adapt themselves to a particular situation

Usage
- Identification
- Optimization
- Clustering, decisions, fault detection...

Focus on a Neuro-Fuzzy controller

An elementary neurone
- Inputs
- Weights
- State
- Activation function
- Output

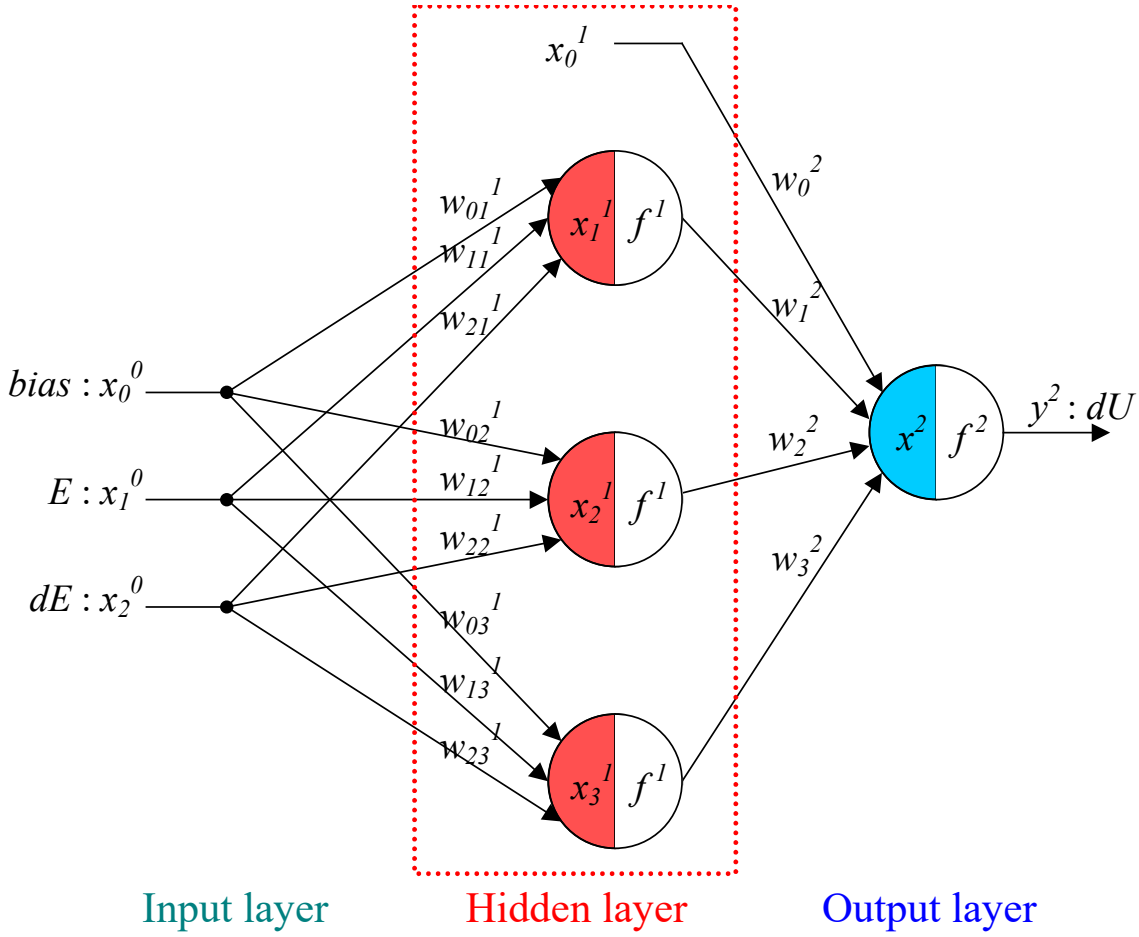
$$Output = f(x) \text{ with } x = \sum_{i=1}^{n} w_i \, x_i$$



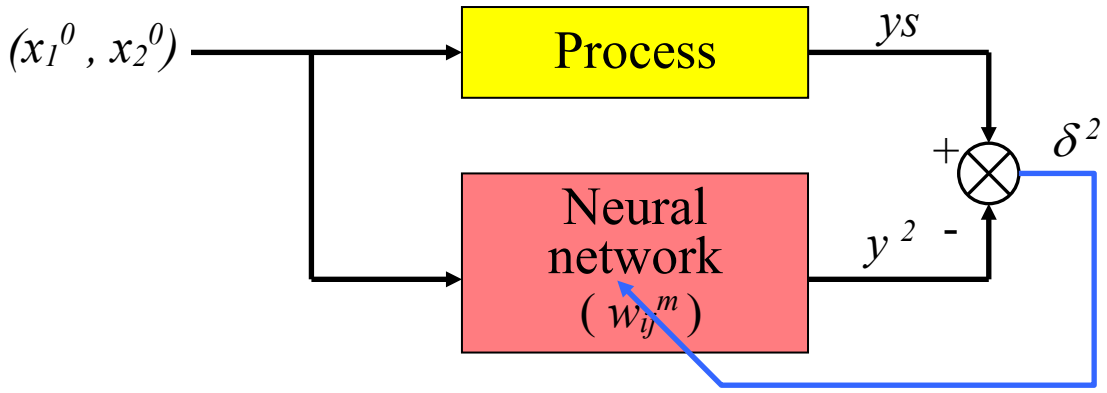
9.4.1. Perceptron

Propagation of information
1 hidden layer

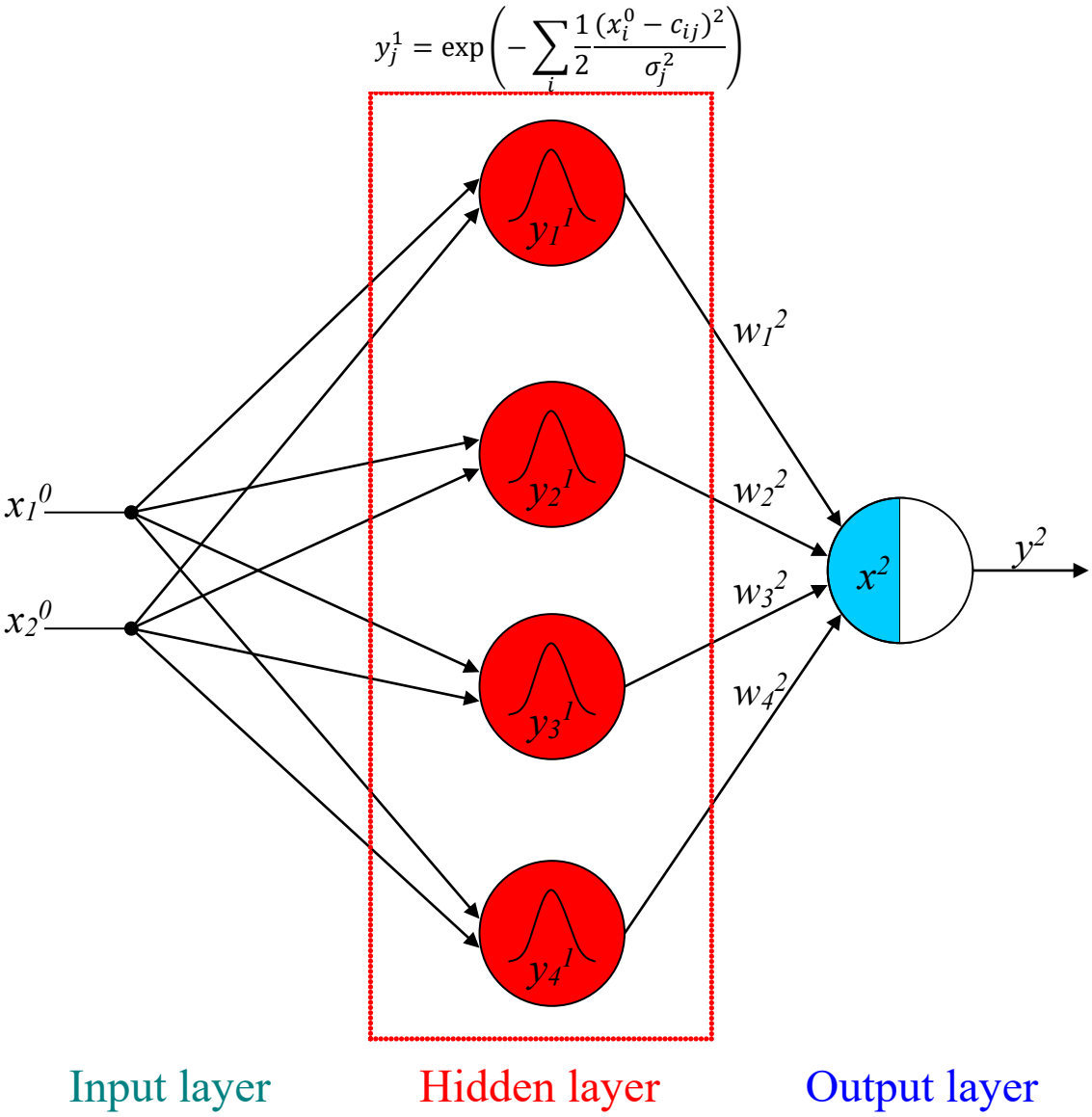
$$x_j^m = \sum_i w_{ij}^m y_i^{m-1}$$
$$y_j^m = f^m(x_j^m)$$



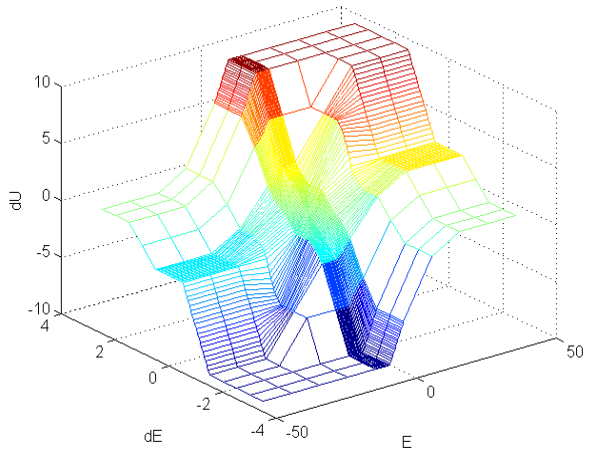
Training process



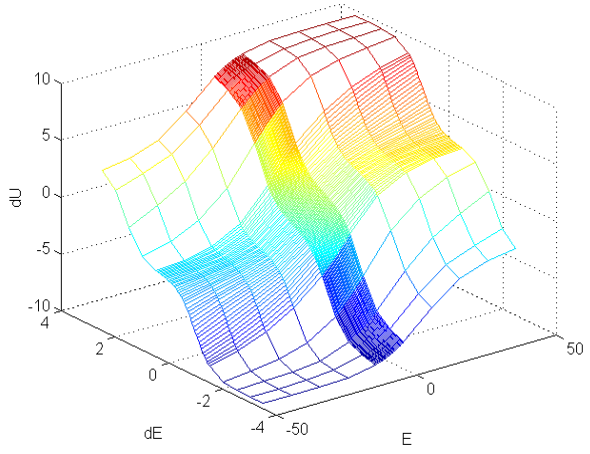
9.4.2. Radial Basis Function (RBF)



Universal approximator (perceptron)
NN mapping a fuzzy control surface



Fuzzy controller



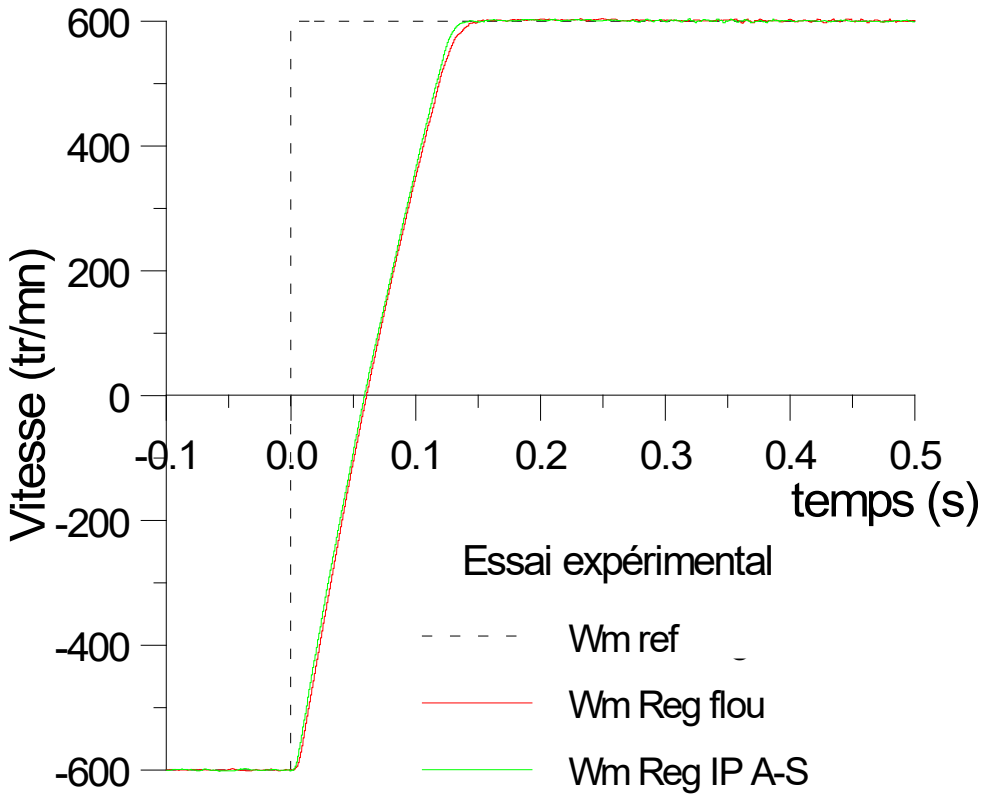
Neural network controller

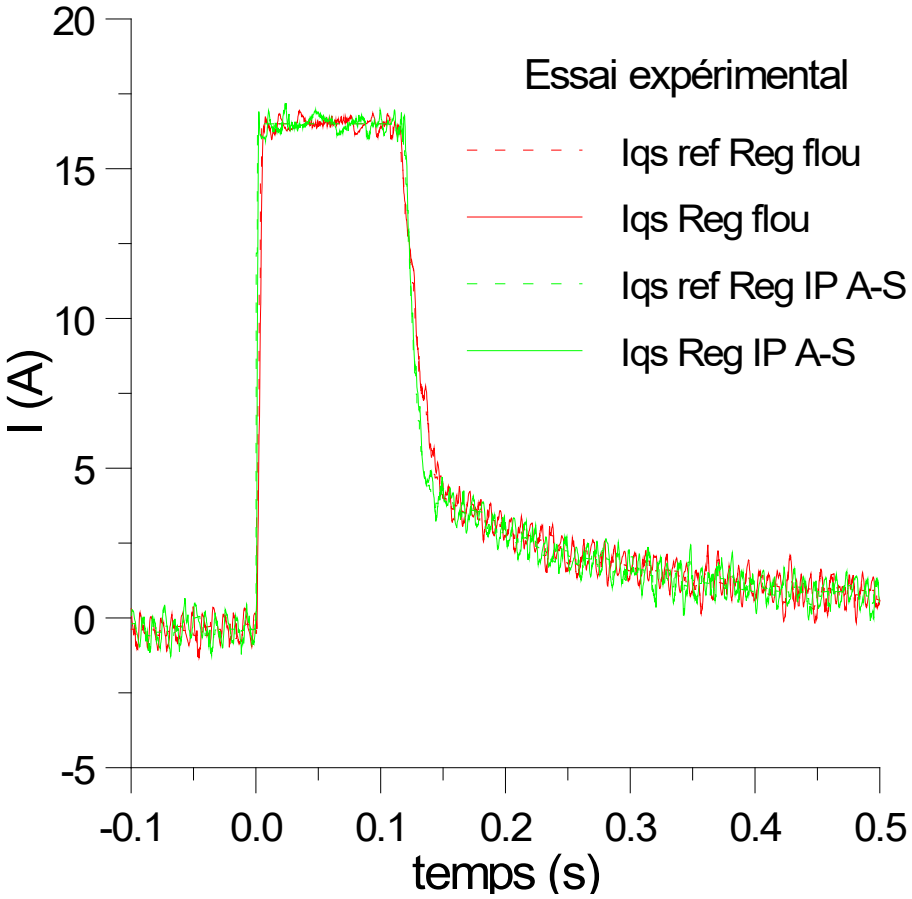
Replacement of a fuzzy controller by a neuronal network one
Structure choice

- Perceptron
- RBF

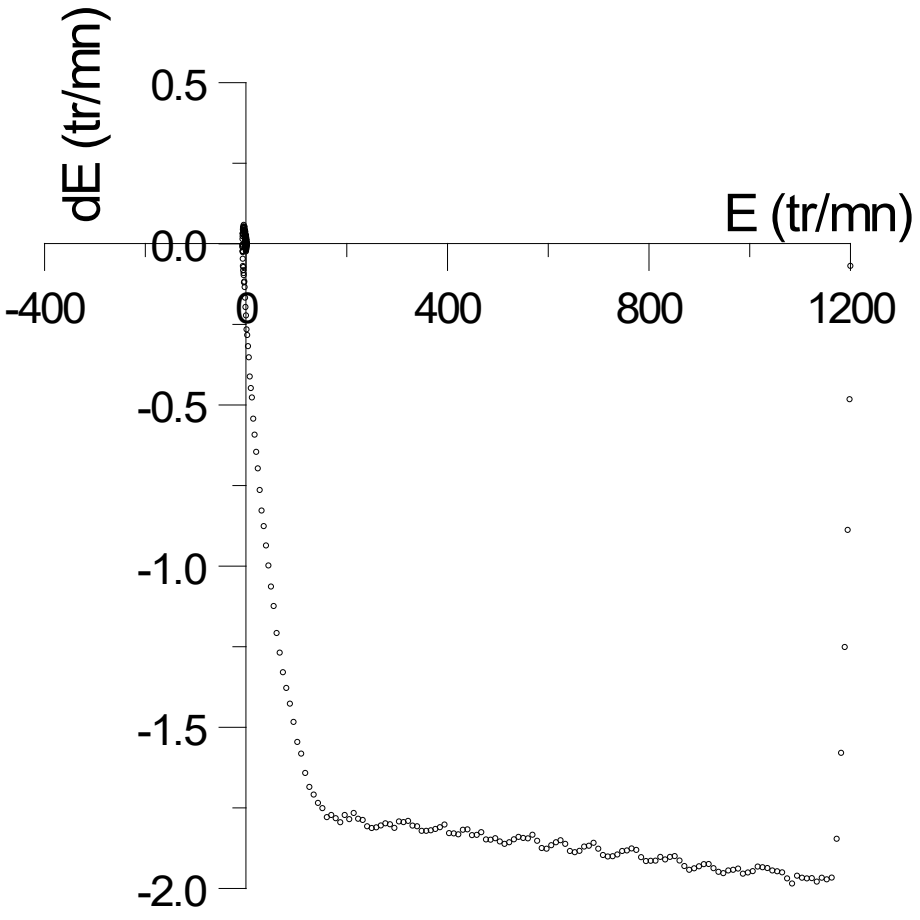
Gain in computing time
Performances?

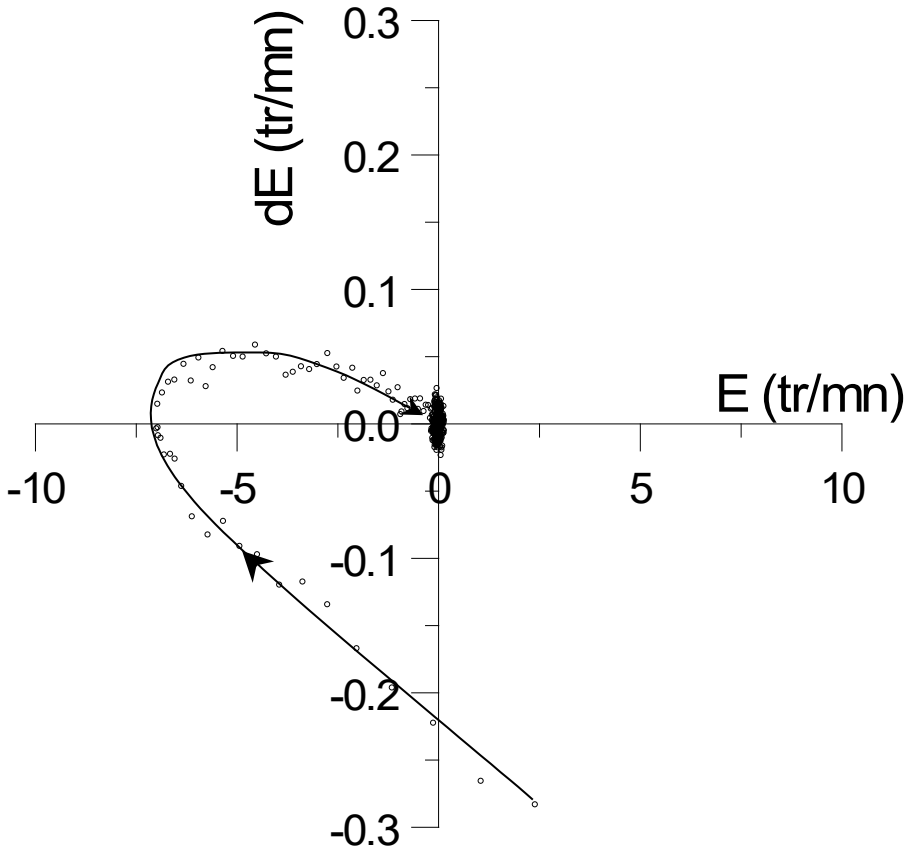
9.5. Applications





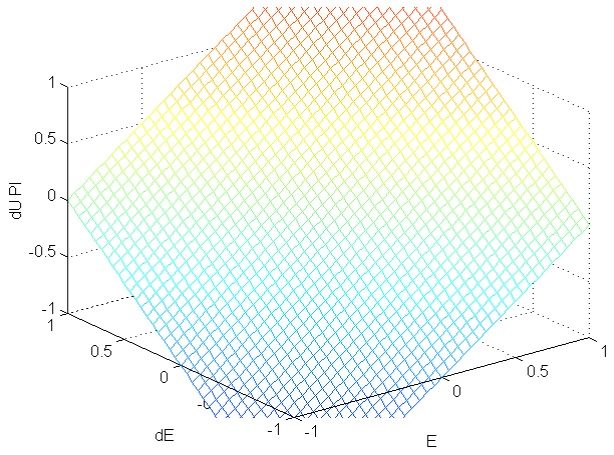
Trajectory in the phase plan


dE (tr/mn)
E (tr/mn)
0.5
0.0
-0.5
-1.0
-1.5
-2.0
-400
0
400
800
1200

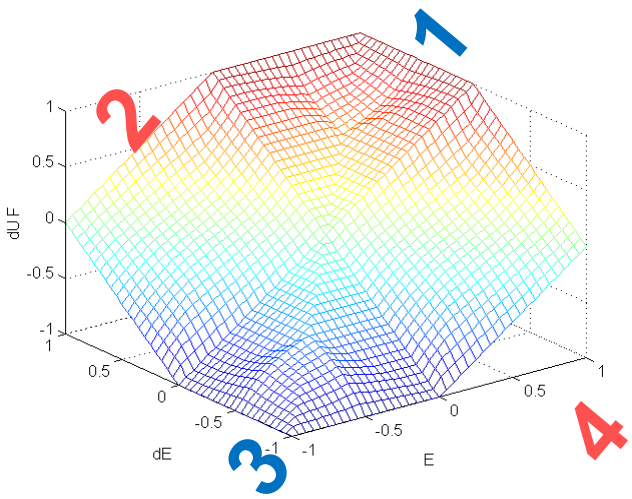


Zoom

Control surfaces in normalized units



IP controller



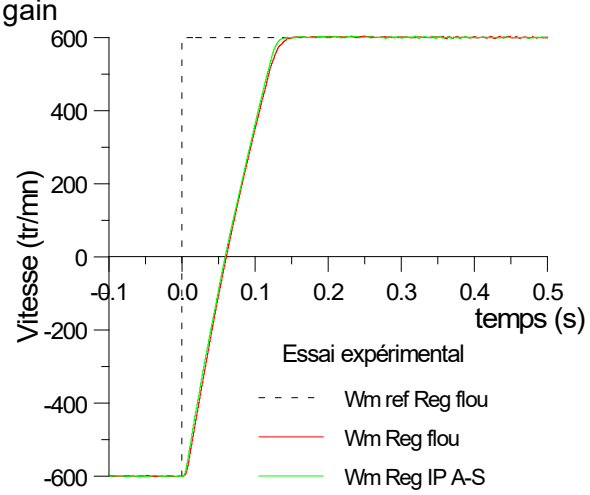
Fuzzy controller

Analyze

- Comparison criteria
- No noticeable gain in performance

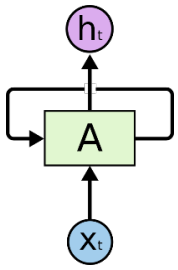
Difference and use

- Tuning method
- Feasibility, implementation
- Computing time gain

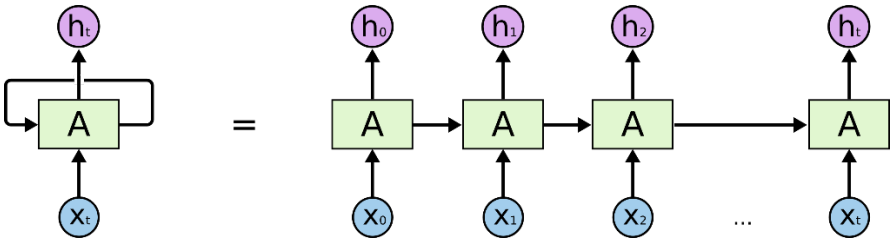


9.1. Recurrent Neural Networks

Recurrent neural networks are NN that allows the information to persist. Unlike perceptron (feed-forward) NN, Recurrent Neural Networks have loops.



A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:

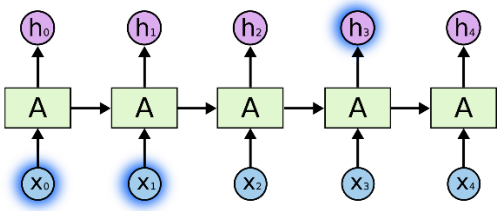


This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They're the natural architecture of neural network to use for such data. In the last few years, RNN were applied to a variety of problems: speech recognition, language modeling, translation, image captioning…

9.2. The Problem of Long-Term Dependencies

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

However, we often only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.



But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent French." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

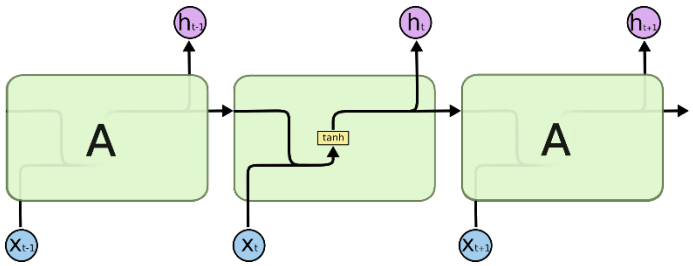
In theory, RNNs are capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them.

9.3. LSTM Networks

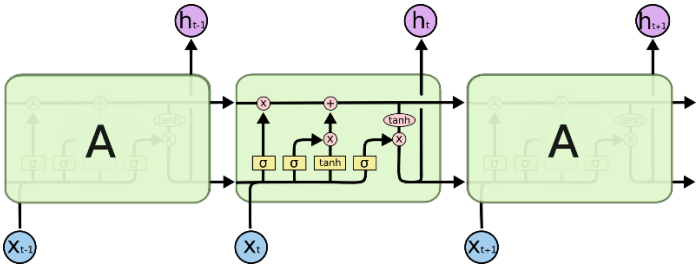
Long Short Term Memory networks (LSTMs) are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997). They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.



The repeating module in a standard RNN contains a single layer.



The repeating module in an LSTM contains four interacting layers.

In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

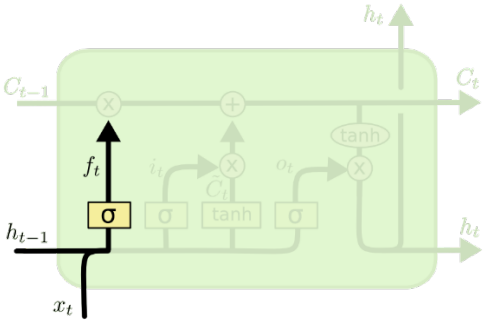
Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”

An LSTM has three of these gates, to protect and control the cell state.

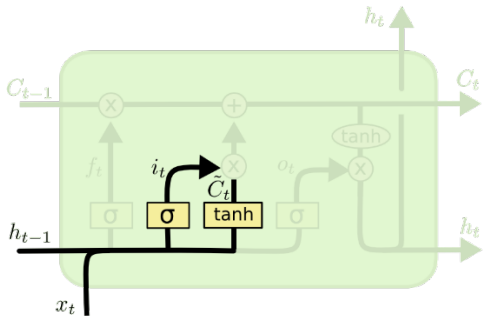
The first step in our LSTM is to decide what information we’re going to throw away from the cell state. This decision is made by a sigmoid layer called the “forget gate layer.” It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents “completely keep this” while a 0 represents “completely get rid of this.”

Let’s go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

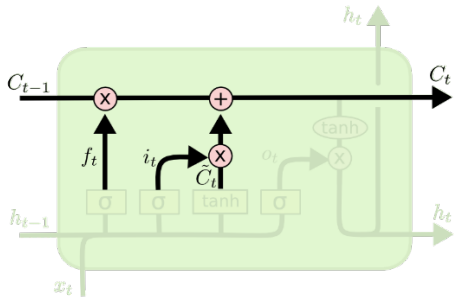
The next step is to decide what new information we’re going to store in the cell state. This has two parts. First, a sigmoid layer called the “input gate layer” decides which values we’ll update. Next, a *tanh* layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we’ll combine these two to create an update to the state.
In the example of our language model, we’d want to add the gender of the new subject to the cell state, to replace the old one we’re forgetting.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

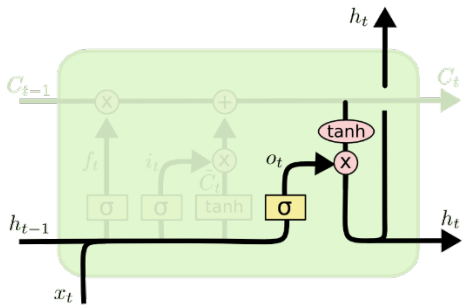
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it.
We multiply the old state by f, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.
In the case of the language model, this is where we would actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, we need to decide what we're going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through *tanh* (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.
For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

9.4. How to implement the models using open source software libraries

We can define neural network type of models using Keras, which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. For other types of models I usually use Scikit-Learn, which is a free software machine learning library, It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to inter-operate with the Python numerical and scientific libraries NumPy and SciPy.

9.5. Metaheuristics and Optimization

Optimization
Metaheuristics
- Genetic Algorithm
- PSO

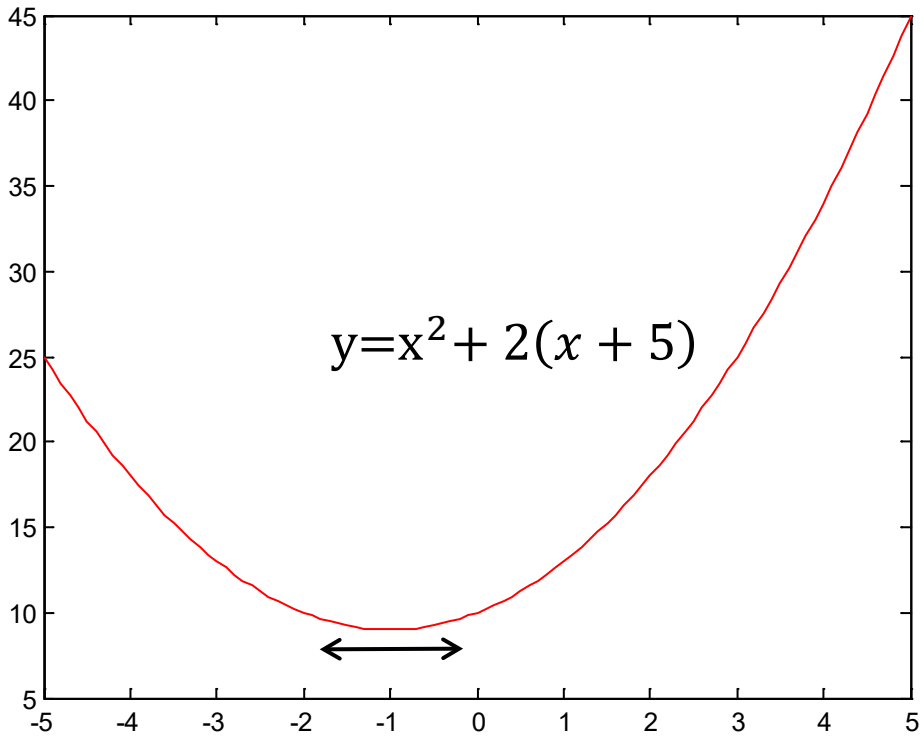
Applications
- Parameter identification of an Induction Machine
- Fourier analysis using PSO

Optimization

From Latin optimum which means the best
Approach consisting in optimizing the functioning of a system
Search for optimal conditions or configurations for various systems
Finding an extremum of a cost function with constraints

Simple example

The calculation of the derivative indicates the extremum
Where the derivative equals zero



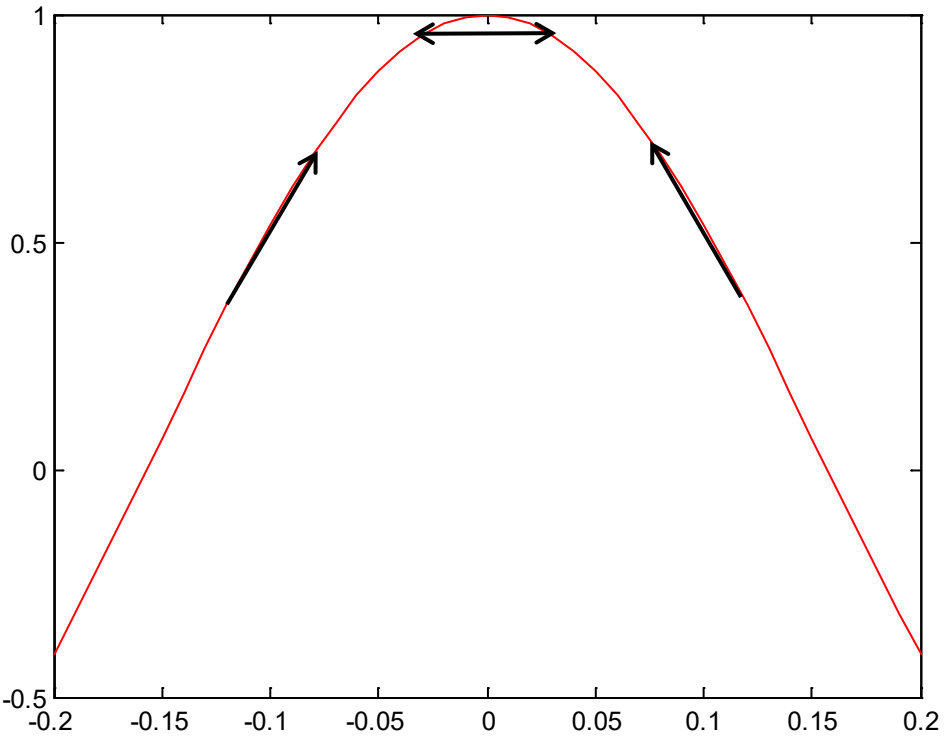
$$y' = 2x + 2 = 0$$

$$\text{for } x = -1$$

Optimization

The derivative calculation indicates the extremum (maximum)

Little by little, by successive iterations
Get close to the point where the derivative equals zero



Optimization

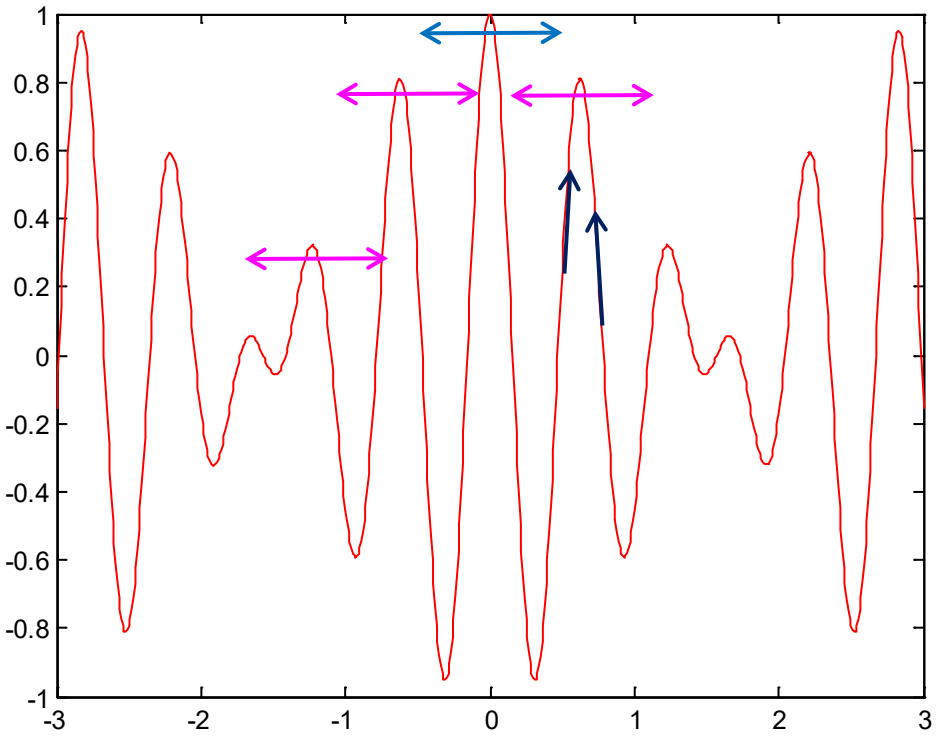
Appearances are deceptive…

$$y(x) = \cos(x) \cdot \cos(10x)$$

Presence of several local maximums

Try it on Octave

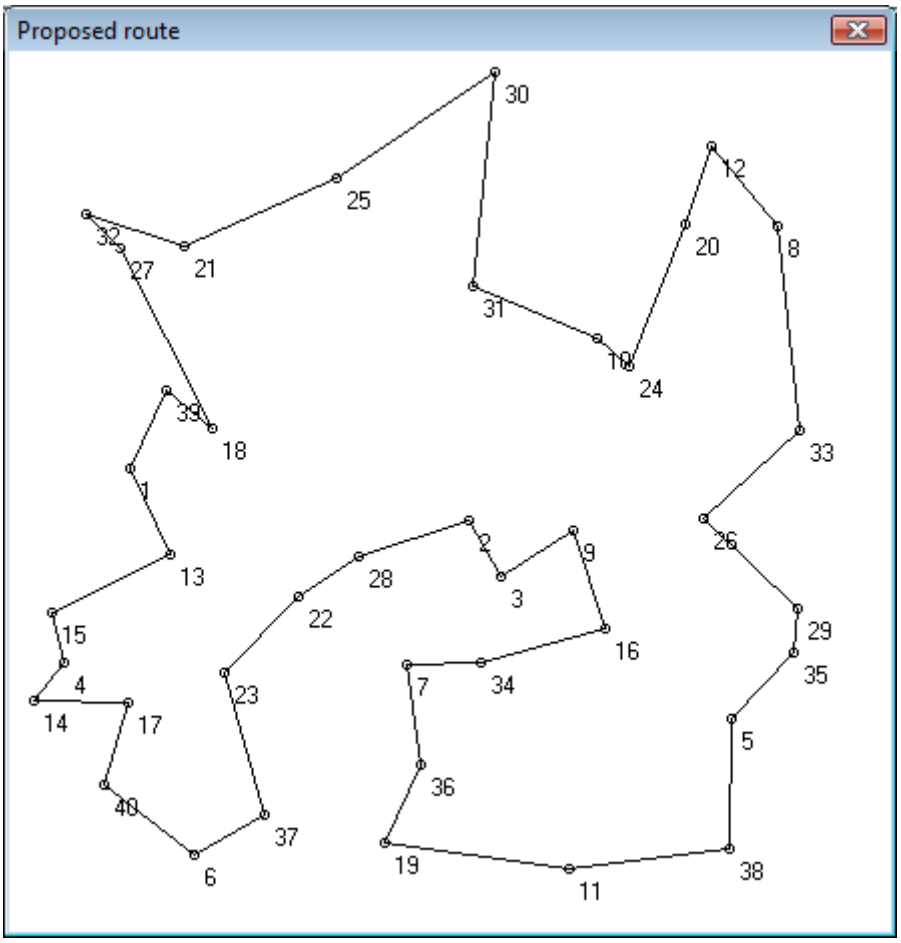
```
x=[-3:0.01:3];
y=cos(x).*cos(10*x);
figure (1); plot (x,y, 'r');

x=[-0.2:0.01:0.2];
y=cos(x).*cos(10*x);
figure (2); plot (x,y, 'r');
```



Optimization
Metaheuristics
- Genetic Algorithm
- PSO

Applications
- Parameter identification of an Induction Machine
- Fourier analysis using PSO

Example the Traveling Salesman Problem (TSP)
Given n points (“cities”) and the distances between each point, find a path of minimum total length that passes exactly once through each point and back to the starting point
This combinatorial optimization problem belongs to the class of NP-Complete problems.



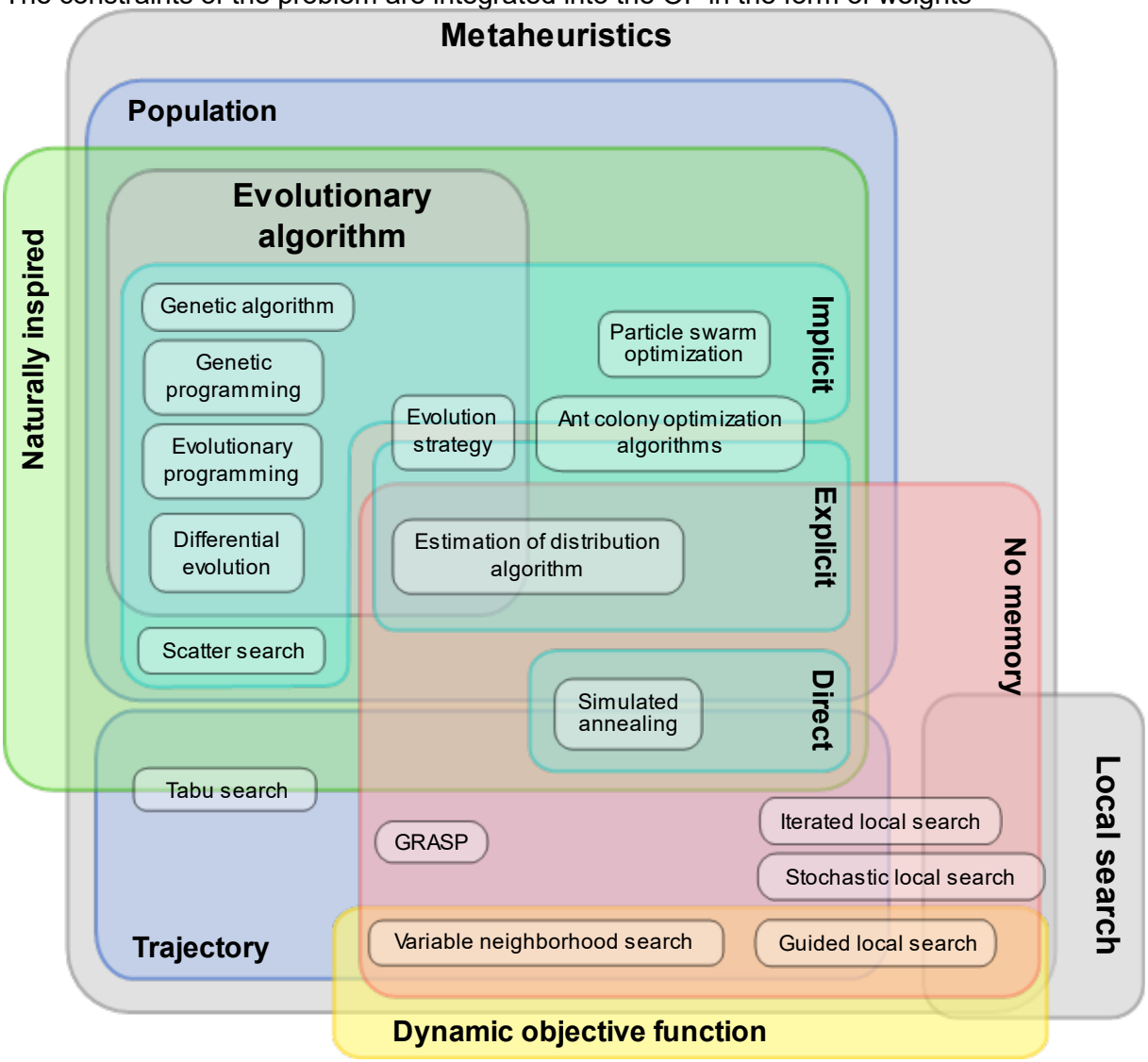
Exact algorithms
Allow finding the optimal solution in a reasonable time in most cases
Approximate algorithms
if we cannot find the best solution, we are satisfied with a “good” solution
Heuristics
Approximate algorithms specific for a type of problem
Metaheuristics
Generic methods that can optimize a wide range of different problems, without changing the algorithm too much

Metaheuristics form a family of optimization algorithms aimed at solving difficult optimization problems for which no more efficient classical method is known
Iterative stochastic algorithms, which progress towards a global optimum, that is to say the global extremum of a function, by sampling an objective function.
There are a large number of different metaheuristics, ranging from simple local search to complex global search algorithms.

Multi-Agent methods
- Genetic algorithm
- PSO

Search space for admissible solutions
Objective function (OF)

Assesses the quality of admissible solutions to the problem
Do not require any derivative calculation or knowledge of the problem (apart from the search space and the objective function).
The constraints of the problem are integrated into the OF in the form of weights

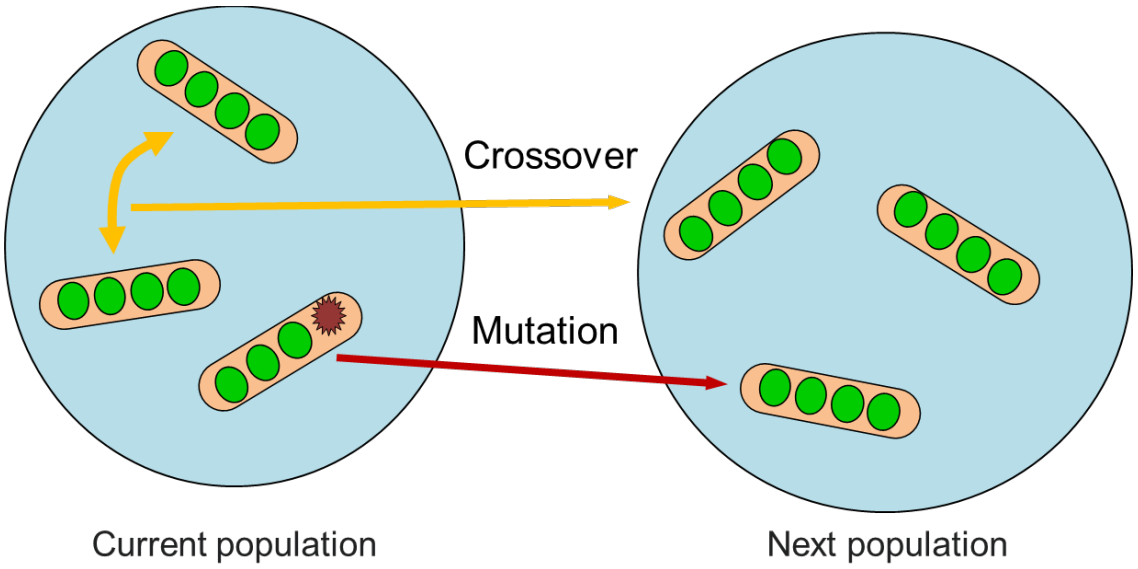


https://en.wikipedia.org/wiki/Metaheuristic

Metaheuristic methods are easy to program
Often converges without adjustment
Not specific to the problem class
Lends itself very well to parallel computation (MPI)
Hybridization (GA, PSO) - Newton Raphson
Trial and error tuning of parameters for faster convergence (population size / number of agents, algorithm coefficients)
No evidence of convergence towards the global extremum

9.6. Genetic Algorithm (GA)

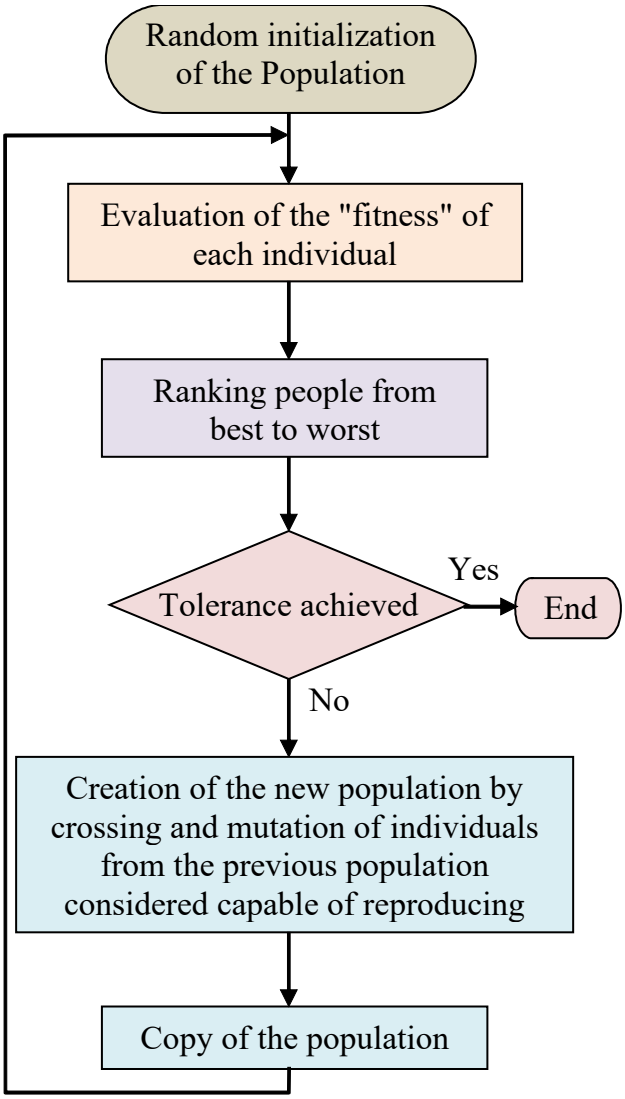
Genetic Algorithm (GA)
Graphical representation
Steps
Initialization
Genetic operators to get the next population:
Crossover and Mutation with Heuristics
Termination



9.6.1. Organizational chart.

Division of the population into sub-layers.
Binary or real encoding of information

Number of individuals	Position	Constitution of the sub-populations
1	0 to 0	Copy: We keep the best individual
9	1 to 9	Random mutation of individuals selected from the top 10 with a factor of 0.001 at most
5	10 to 14	Random mutation of individuals selected from the first 15 with a factor of at most 0.1
10	15 to 24	Random crossing of two individuals chosen from the first 15 without favoring one parent over the other
10	25 to 34	Random mutation of individuals chosen among the first 35 with a factor of at most 0.5
20	35 to 54	Random mutation of a single gene from the first individual in the population with a factor of up to 0.01



Genetic Algorithm: Classification, copy, mutation, crossover

9.6.2. GA code

```
void TGenDemWindow::Classement()
{     // Tri à bulles
double temp;
int i, j;
bool Encore;
      do {
            Encore=false;
            for ( i=0; i<GenN-1; i++)
             if ( Fitness[i]<Fitness[i+1] )
                  {
```

```
                    Encore=true;
                    for ( j=0; j<m; j++)
                          {
                          temp=X[i][j];
                          X[i][j]=X[i+1][j];
                          X[i+1][j]=temp;
                          }
                    temp=Fitness[i];
                    Fitness[i]=Fitness[i+1];
                    Fitness[i+1]=temp;
                    j=iOld[i];
                    iOld[i]=iOld[i+1];
                    iOld[i+1]=j;
                    }
              }     while(Encore);
        FitnessMin=Fitness[GenN-1];
        FitnessMax=Fitness[0];
}

void TGenDemWindow::MutationBest()
{
        for ( int i=GenNsel; i<GenNMutBest; i++)
              {
              int p;
              p=rand()%GenNMutBest;
              for ( int j=0; j<m; j++)
                    {
                    Xnew[i][j]=X[p][j]*( 1+GenamutBest*(0.5-
(rand()%5000)/5000.0) );
                    if ( Xnew[i][j]<=0 )  Xnew[i][j]=X[p][j];   // limits
                    }
              if ( Xnew[i][3]>=1.0 )      Xnew[i][3]=X[p][3];   // limits
              }
}

void TGenDemWindow::Croisement()
{
        for ( int i=GenNMutBest2; i<GenNcr; i++)
              {
              int p1, p2;
              p1=rand()%GenNMutBest2;
              p2=rand()%GenNMutBest2;
              for ( int j=0; j<m; j++)
                    Xnew[i][j]= Genacr*X[p1][j] + (1-Genacr)*X[p2][j];
              }
}
```

9.7. Particle Swarm Optimization (PSO)

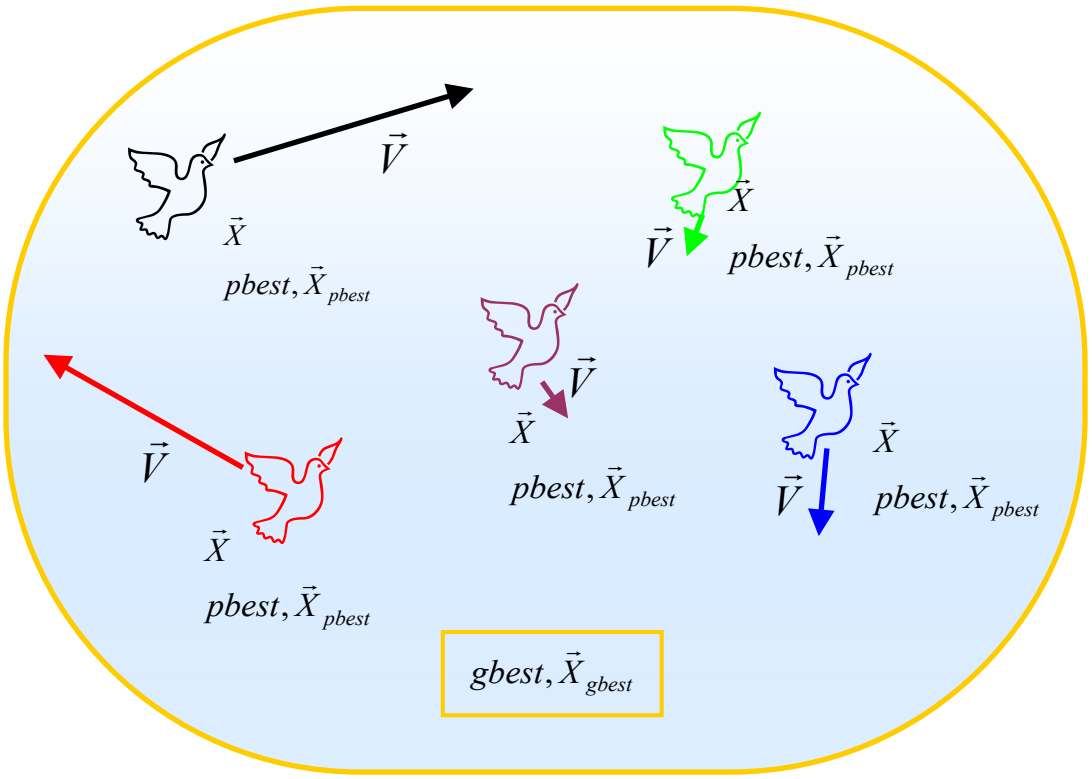
Origin
Bird flocking, fish schooling and swarming theory
N Agents
Each has M parameters (dimension of the system)
The X position in the space of each agent codes the value of the parameters to look for

Additional references

Kennedy, J.; Eberhart, R.; "Particle swarm optimization" in ***1995*** *IEEE International Conference on Neural Networks Proceedings (Cat. No.95CH35828), 1995, V4, pp 1942-1948*

Kennedy, J.; Spears, W. M.; "Matching algorithms to problems : an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator" in *Proceedings of the **1998** International Conference on Evolutionary Computation, pp 78-83*

L. BAGHLI , GREEN – Nancy Université, *Particle Swarm and Genetic Algorithms applied to the identification of Induction Machine Parameters*, EPE 2003



Random initialization of the position of the agents **i** (particles) **X** and of their speed **VX**

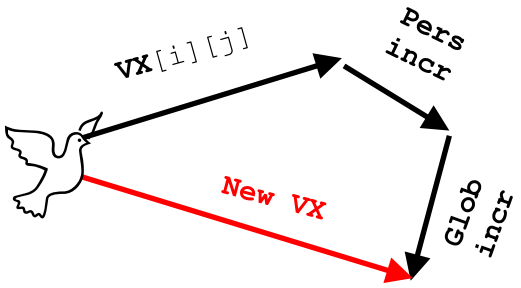
```
// initialisation aléatoire entre 0 et 1
for (i=0; i<N; i++)          // N : Nombre d'agents
      {
      for (j=0; j<M; j++)    // M : Nombre de paramètres
            {
            VX[i][j]=0.0;
```

```
            X[i][j]=Random(eps, 1-eps);
            }
      pbest[i]=1e30;   // Eval de FO de l'agent i
      }
Epoch=0;    // Epoque (génération)
gbest=0;    // numéro de l'agent ayant la meilleure Eval
```

Evaluation of the OF of agent **i**

```
double Min=1e50;
for (i=0; i<N; i++)          // Pour chaque agent i
      {
      Eval[i]=ErreurQuadFx( X[i]);
      if (Eval[i] < pbest[i])    // Est-ce un nouveau personnel best
      ?
            {
      pbest[i]=Eval[i];      // Sauve son Eval et ses coordonnées
      for (j=0; j<M; j++)   pbestX[i][j]=X[i][j];
      }
      if (Min > pbest[i])   // Est-ce un nouveau global best ?
            {
      Min= pbest[i];
      gbest=i;
            }
      }
```

Calculation of the new **VX** speed of agent **i** and its new position **X**



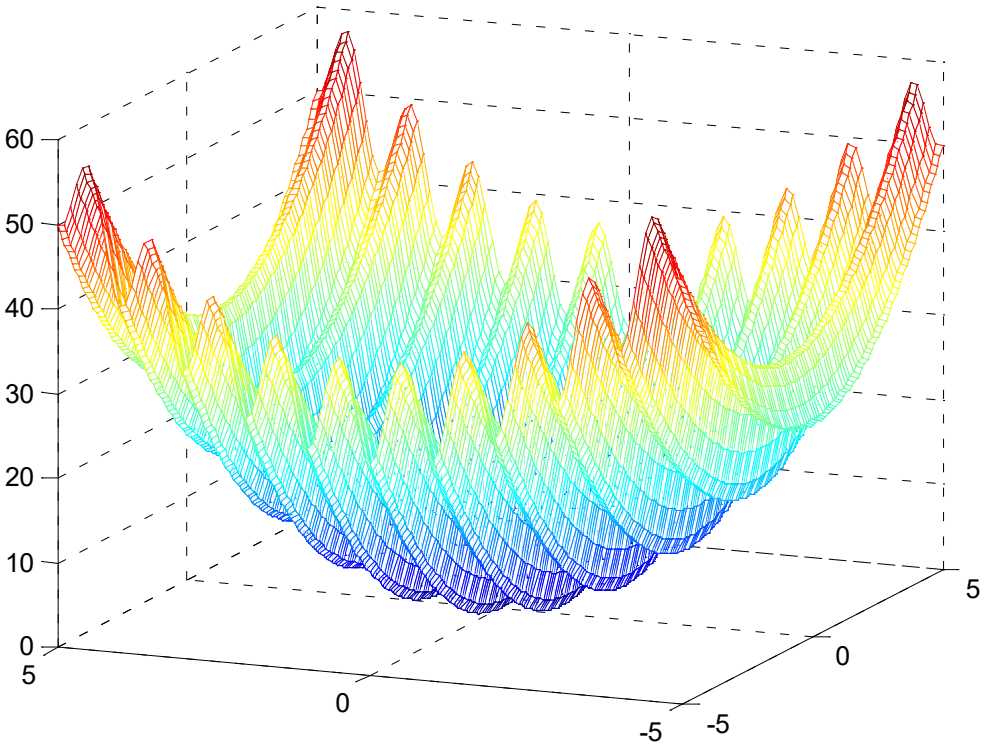

```
// Vitesses
for (i=0; i<N; i++)
      for (j=0; j<M; j++)
      {
      VX[i][j] = w*VX[i][j]
               +(rand()%1000)/1000.0*pincrement*(pbestX[i][j]-X[i][j])
        +(rand()%1000)/1000.0*gincrement*(pbestX[gbest][j]-X[i][j]);
      }
// Positions
for (i=0; i<N; i++)          // Pour chaque agent i
      for (j=0; j<M; j++)    // Pour chaque paramètre j
      {
```

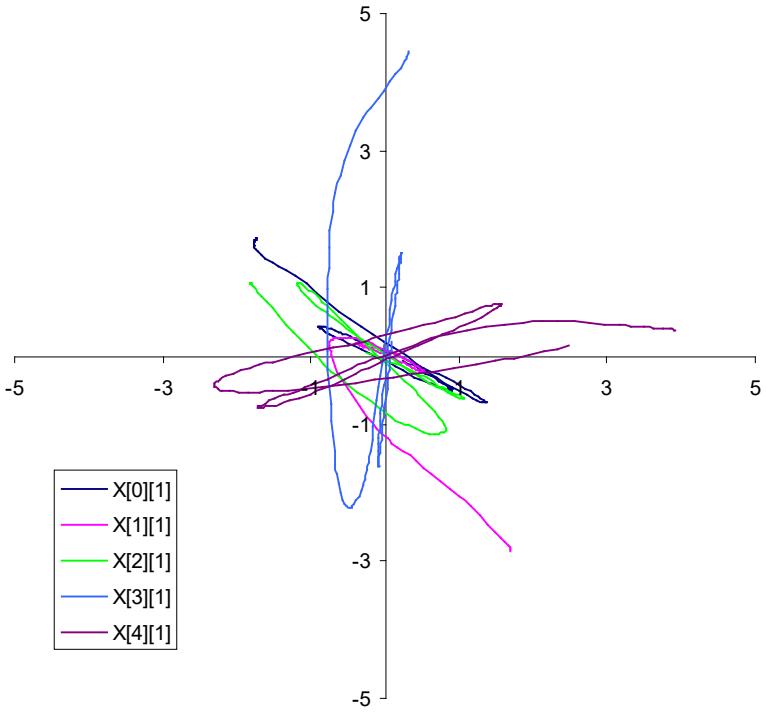
```
X[i][j] += VX[i][j]*dt;    // Intègre la vitesse
// Vérifie si l'on sort de l'espace de recherche
// ...
}
```

Evolution of the 5 agents (optimization with 2 parameters)

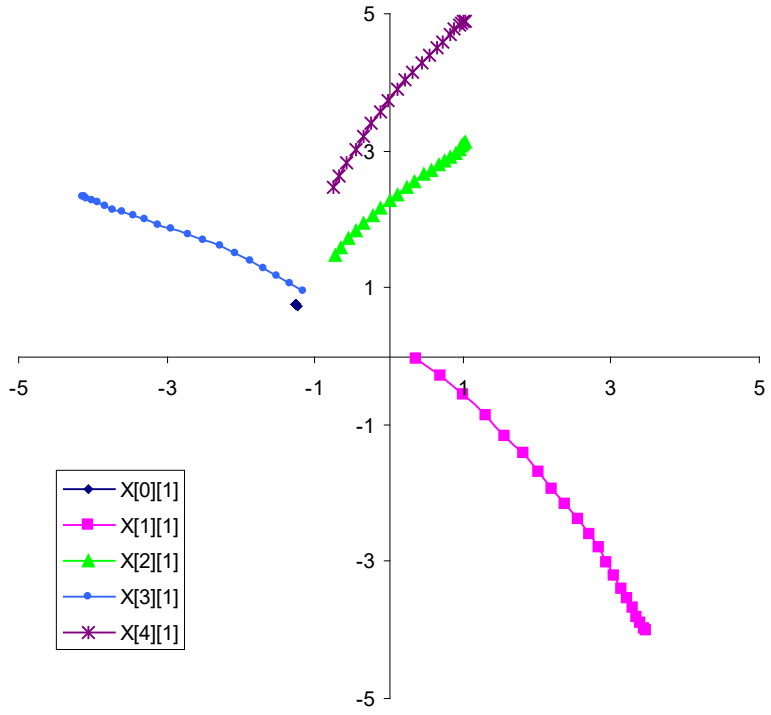
Rastrigin function of dimension *m*

$$Eval_i = 3m + \sum_{j=1}^{m} \left(x_{ij}^2 - 3\cos\left(2\pi x_{ij}\right)\right)$$

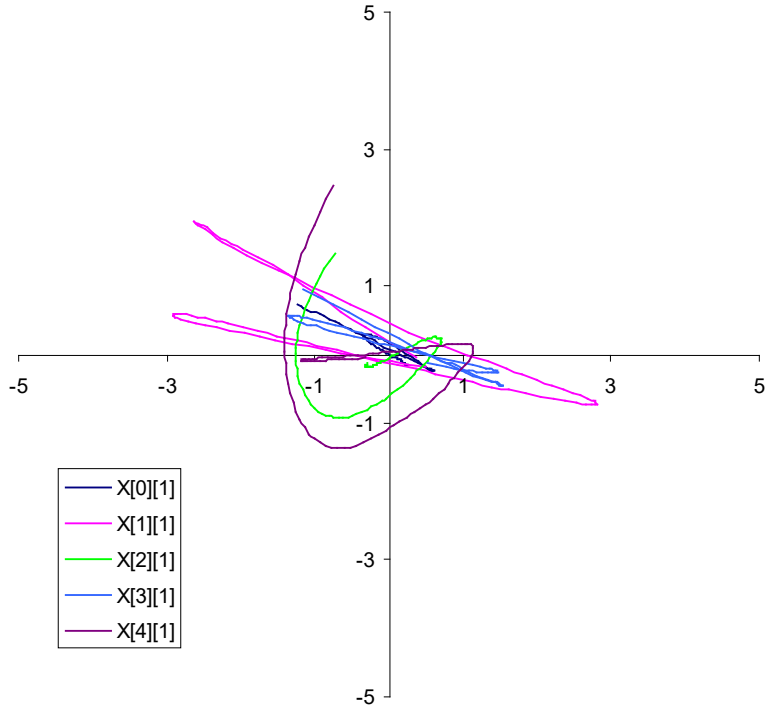




Evolution of the 5 agents (optimization with 2 parameters)



Epoch 0 to 20



Epoch 20 to 120

9.8. Practical application, example 1

Identification of the parameters of a MAS

It is necessary to excite the maximum of modes

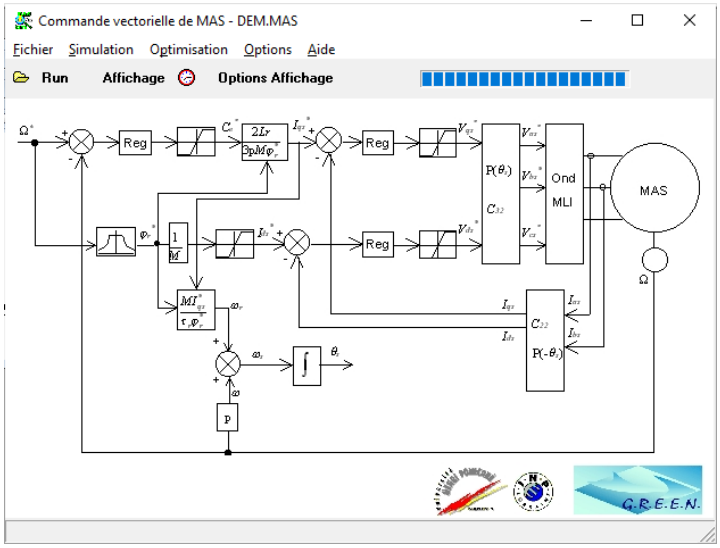
Direct start: acquire experimental mechanical speed and phase current I_{as}

Transient model of MAS in unsaturated regime, simulated under MASVECT, for each agent

Electromagnetic parameters: R_s , τ_s , τ_r , σ

Mechanical parameters: J, a_1 , a_2 , a_3

$$\begin{cases} \dfrac{dI_{ds}}{dt} = \dfrac{1}{\sigma L_s} V_{ds} - \dfrac{1}{\sigma}\left(\dfrac{1}{\tau_s} + \dfrac{1}{\tau_r}\right) I_{ds} - \mathrm{p}\Omega I_{qs} + \dfrac{1}{\sigma L_s \tau_r} \phi_{ds} + \dfrac{1}{\sigma L_s} \mathrm{p}\Omega \phi_{qs} \\ \dfrac{dI_{qs}}{dt} = \dfrac{1}{\sigma L_s} V_{qs} + \mathrm{p}\Omega I_{ds} - \dfrac{1}{\sigma}\left(\dfrac{1}{\tau_s} + \dfrac{1}{\tau_r}\right) I_{qs} - \dfrac{1}{\sigma L_s} \mathrm{p}\Omega \phi_{ds} + \dfrac{1}{\sigma L_s \tau_r} \phi_{qs} \\ \dfrac{d\phi_{ds}}{dt} = V_{ds} - R_s I_{ds} \\ \dfrac{d\phi_{qs}}{dt} = V_{qs} - R_s I_{qs} \\ \dfrac{d\Omega}{dt} = \dfrac{1}{J}\left(\dfrac{3}{2}\mathrm{p}(\phi_{ds} I_{qs} - \phi_{qs} I_{ds}) - a_1 \Omega^2 - a_2 \Omega - a_3\right) \end{cases}$$

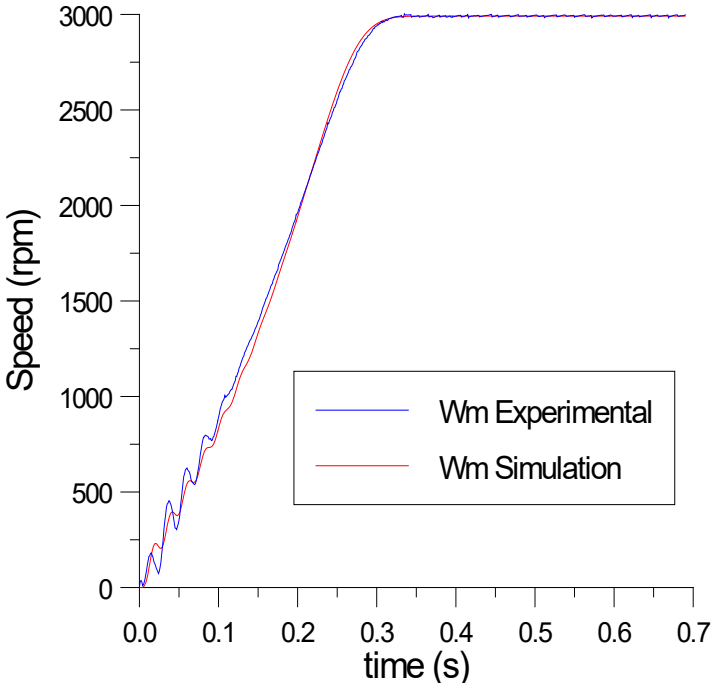


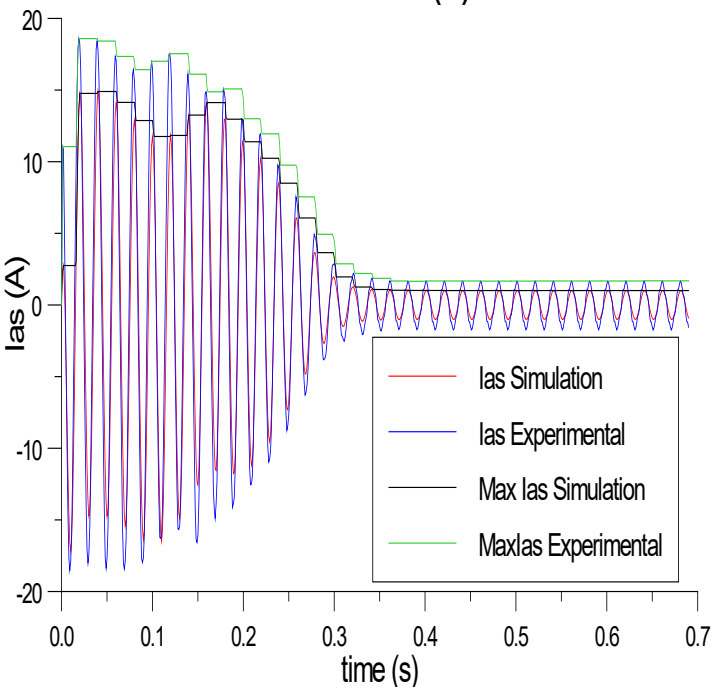
Identification of an IM parameters

Cost function

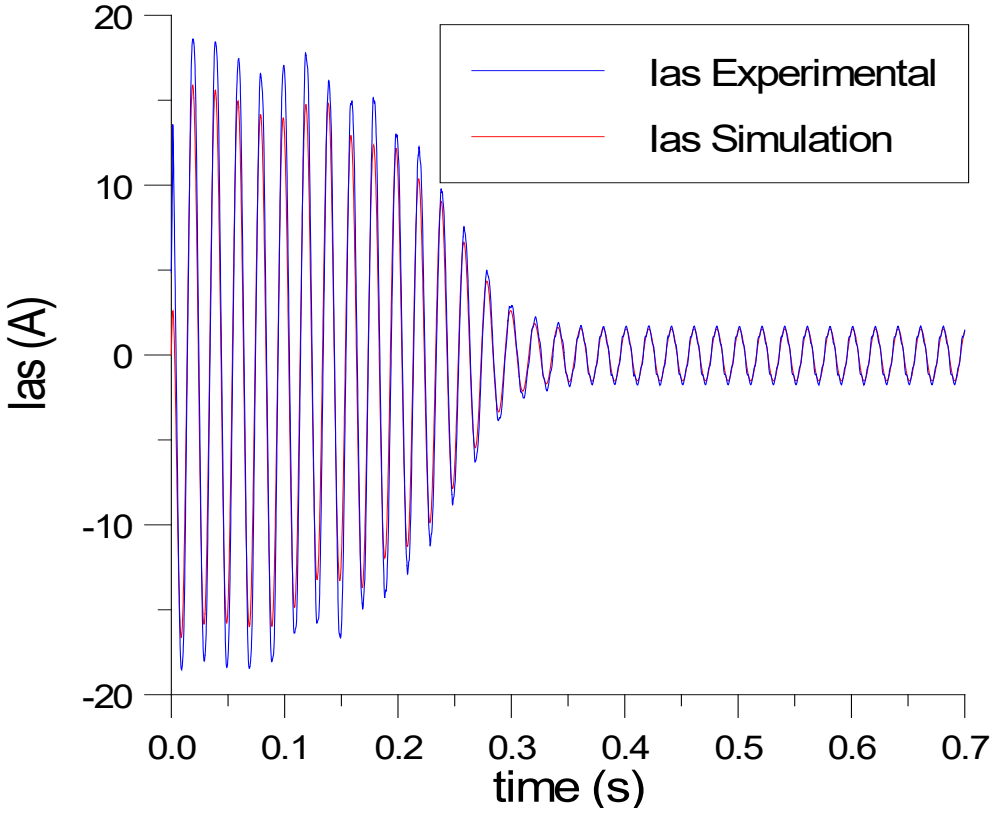
$$E_r = \int \left(\frac{1}{1 + e^{gA(gt0 - t)}} \left[(\Omega sim - \Omega exp)^2 + \lambda (I_{as} sim - I_{as} exp)^2 \right] \right) dt$$

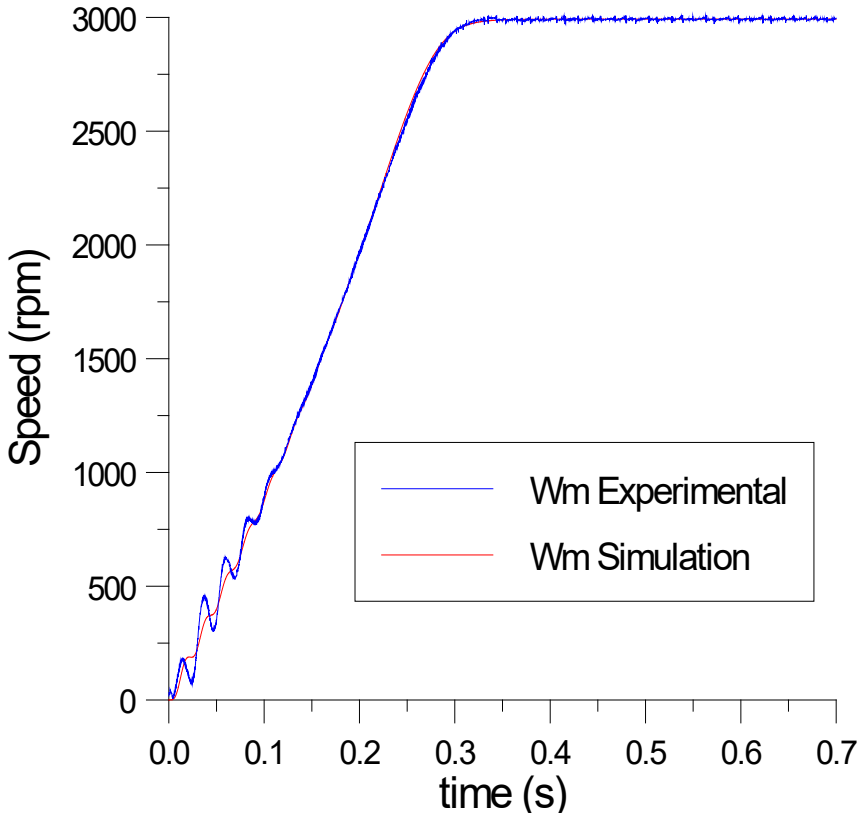
Intermediate result (PSO: Epoch 80)





Final result (GA)





9.9. Practical application, example 2

Decomposition of a square wave in Fourier series
Parameters of Fourier coefficients: B1, B3, B5 ...
Same problem 5.6 can be solved using PSG or GA

agent *i*:

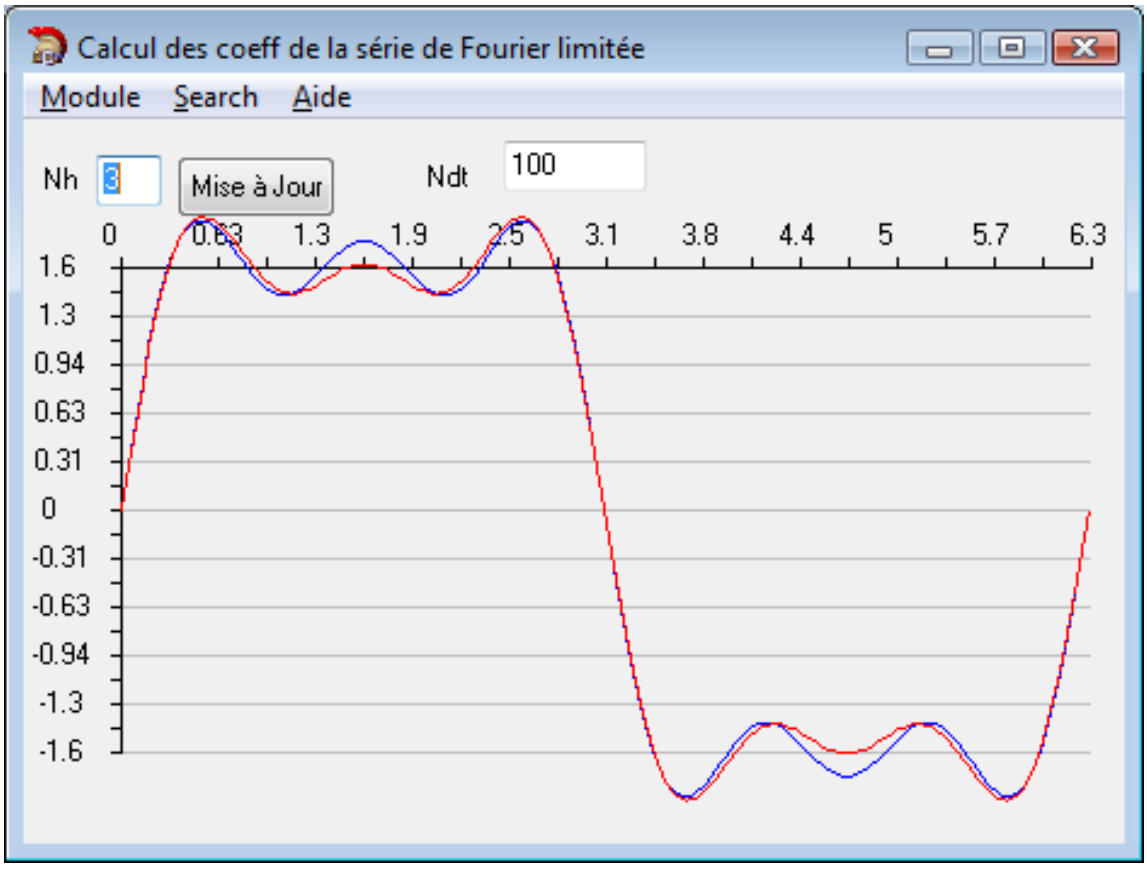
$$u_i(x) = \sum_{k=0}^{M-1} B_{i,k}\, sin\big((2k+1)x\big)$$

Theory:

$$u_F(x) = \sum_{k=0}^{M-1} \frac{2}{2k+1} sin\big((2k+1)x\big)$$

OF:

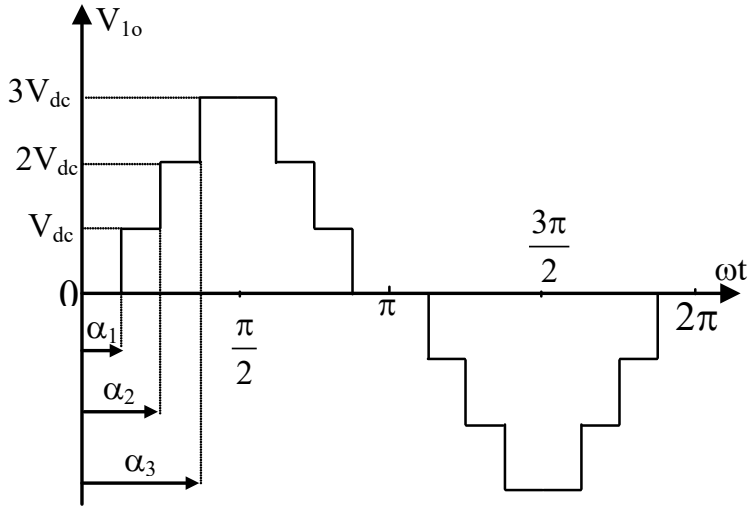
$$Err = \frac{2}{\pi}\int_0^{\frac{\pi}{2}} \Big(\frac{\pi}{2} - u(x)\Big)\, dx$$



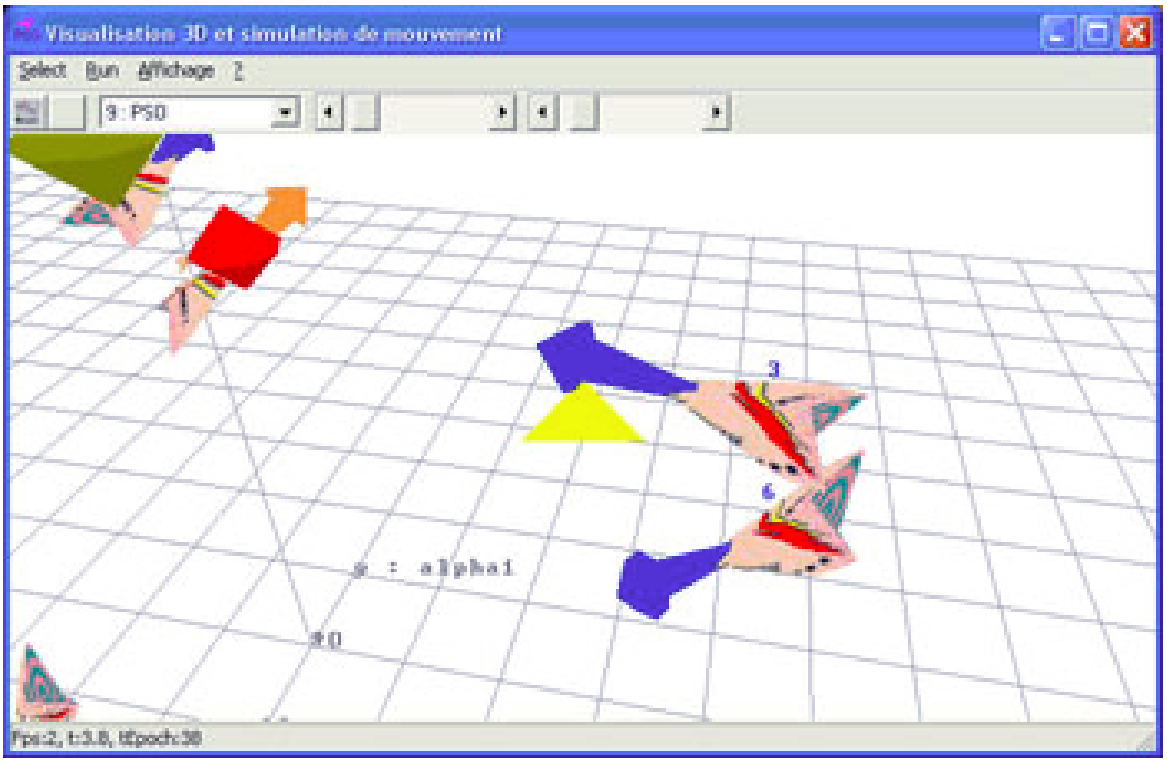
9.10. Practical application, example 3

Finding the switching angles of a multi-level inverter to eliminate voltage harmonics

Parameters: α_1, α_2, α_3



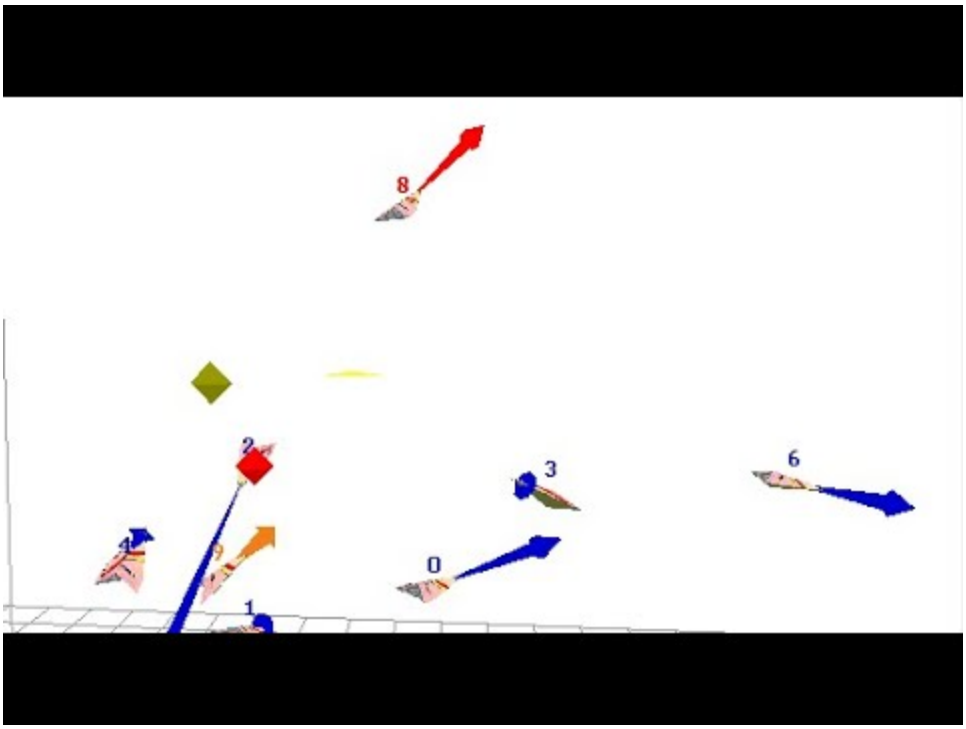
Animation: https://youtu.be/uDPy-6DThXw



A 3D visualization of an optimization progress using Particle Swarm Optimization (PSO) algorithm

This correspond to searching an optimal configuration (3 switching angles) for a PWM Pulse Width Modulation to eliminate harmonics for a 3-phases multilevel inverter

Research papers concerning this study
https://www.researchgate.net/publication/222686244_Harmonic_elimination_in_diode-clamped_multilevel_inverter_using_evolutionary_algorithms
https://www.researchgate.net/publication/229036332_Particle_Swarm_and_Genetic_Algorithms_applied_to_the_identification_of_Induction_Machine_Parameters



9.11. PSO vs GA

GA, Individuals (agents) carry the information
PSO, the information is in memory (personal best) and the agent continues to search for a better solution
GA takes longer to converge than the PSO
GA and PSO are suitable for parallel computing
GA and PSO used as a starting point for N-R or gradient-conjugate algorithms
No mathematical proof or guarantee to find the global optimum

9.12. Summary

Modern optimization methods use metaheuristics like PSO and GA
They do not require the derivative of the cost function and have less chance to get stuck in local optimum
Artificial intelligence can be implemented using neural networks and fuzzy logic inference

10. Machine Learning

Machine Learning

Clustering, classification and regression. Applications: Electricity demand forecasting from weather data, Identifying home appliances with non-intrusive load monitoring

Examples of the importance of prediction in energy systems
Application: Electricity demand forecasting from weather data
Linear regression models and the least squares method

References

C. Cogos, G. Gerogiou, Appliance Identification Through Nonintrusive Load Monitoring in Residences , in book Computational Intelligence and Optimization Methods for Control Engineering, pp 227–244, 2019, Springer, DOI: 10.1007/978-3-030-25446-9_10
Reddy, T. A. (2011). Applied data analysis and modeling for energy engineers and scientists. New York: Springer. DOI:10.1007/978-1-4419-9613-8
Luca Lorenzoni,Paolo Cherubini,Davide Fioriti,Davide Poli,Andrea Micangeli,Romano Giglioli, Classification and modeling of load profiles of isolated mini-grids in developing countries: A data-driven approach, Energy for Sustainable Development, Elsevier, December 2020, DOI: 10.1016/j.esd.2020.10.001

10.1. Introduction

Machine learning algorithms use computational methods to "learn" information directly from data without relying on a predetermined equation as a model. Machine learning build a model based on sample data (training data), in order to make predictions or decisions without being explicitly programmed to do so.

The algorithms adaptively improve their performance as the number of samples available for learning increases.

They are used every day to make critical decisions in medical diagnosis, stock trading, energy load forecasting, and more.

Media sites rely on machine learning to sift through millions of options to give you song or movie recommendations. Retailers use it to gain insight into their customers' purchasing behavior.

Real-World Applications

With the rise in big data, machine learning has become particularly important for solving problems in areas like these:

• Computational finance, for credit scoring and algorithmic trading

• Image processing and computer vision, for face recognition, motion detection, and object detection

• Computational biology, for tumor detection, drug discovery, and DNA sequencing

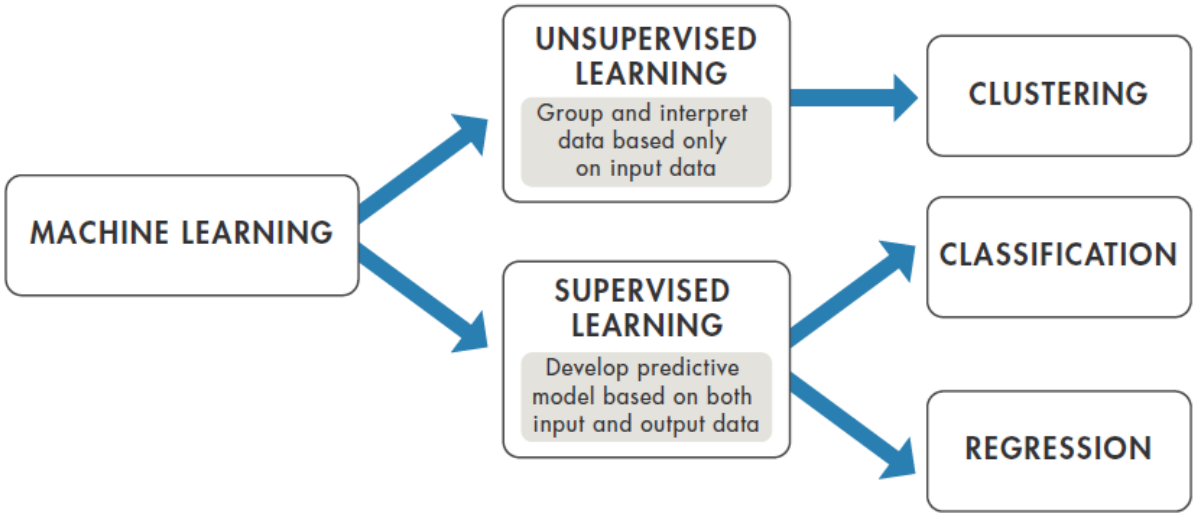
• Energy production, for price and load forecasting

• Automotive, aerospace, and manufacturing, for predictive maintenance

• Natural language processing

10.2. Machine learning categories

We can divide machine learning techniques in supervised and unsupervised learning.



Supervised learning trains a model on known input and output data so that it can predict future outputs.

Unsupervised learning finds hidden patterns or intrinsic structures in input data. It is doing **clustering**.

10.3. Supervised Learning

The aim of supervised machine learning is to build a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data.

Supervised learning uses classification and regression techniques to develop predictive models.

Classification techniques predict discrete responses—for example, whether an email is genuine or spam, or whether a tumor is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.

Regression techniques predict continuous responses— for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

Example: Using Supervised Learning to Predict Heart Attacks

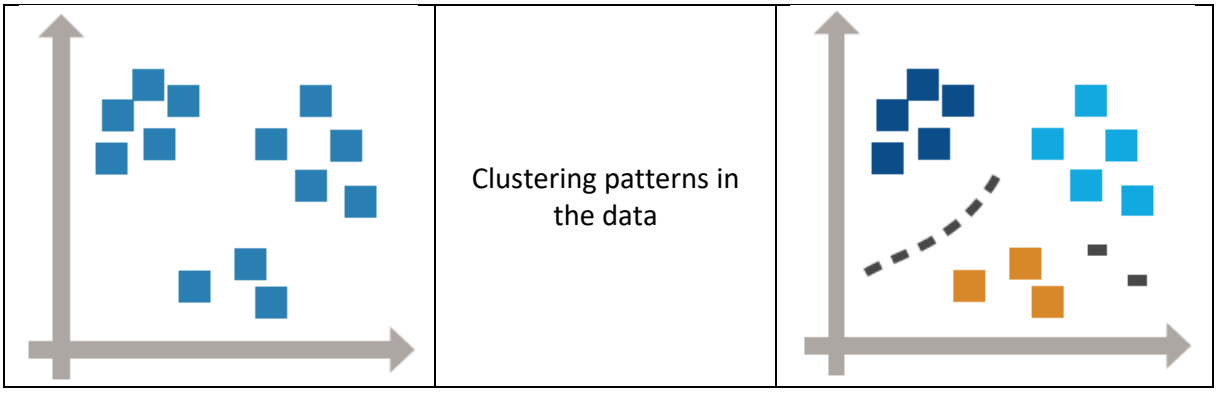
Suppose clinicians want to predict whether someone will have a heart attack within a year. They have data on previous patients, including age, weight, height, and blood pressure. They know whether the previous patients had heart attacks within a year. So, the problem is combining the existing data into a model that can predict whether a new person will have a heart attack within a year.

10.4. Unsupervised Learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labeled responses.

Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data.

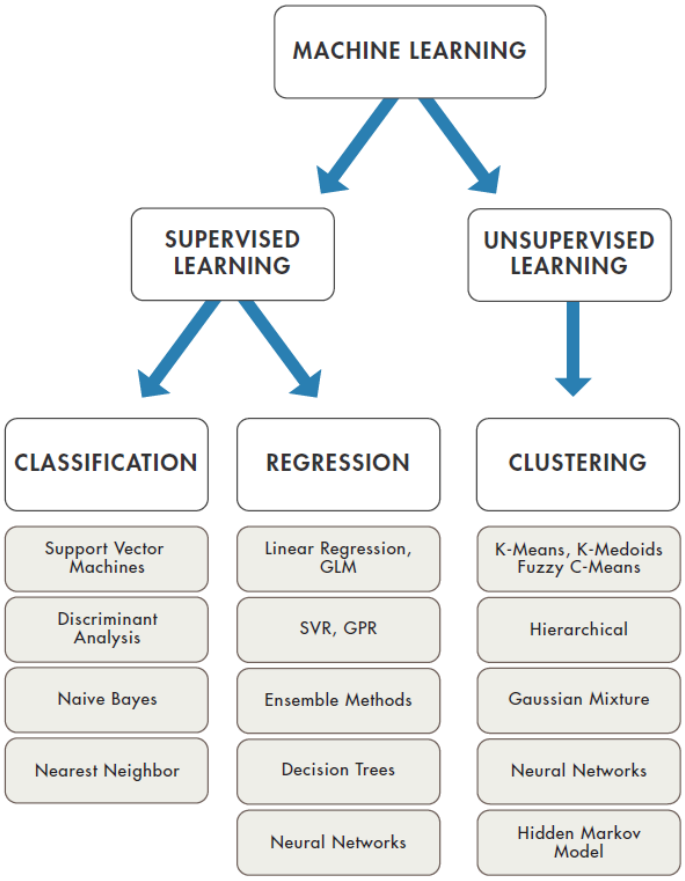
Applications for clustering include gene sequence analysis, market research, and object recognition.



10.5. Choosing the right algorithm

Choosing the right algorithm can seem overwhelming—there are dozens of supervised and unsupervised machine learning algorithms, and each takes a different approach to learning.

There is no best method or one size fits all. Finding the right algorithm is partly just trial and error. Even highly experienced data scientists can't tell whether an algorithm will work without trying it out. But algorithm selection also depends on the size and type of data you're working with, the insights you want to get from the data, and how those insights will be used.



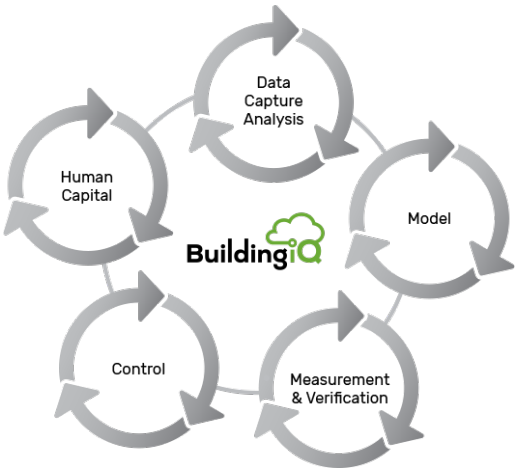
10.6. Usage of Machine Learning

We should use machine learning if we face a complex task or problem involving a large amount of data and lots of variables, but no existing formula or equation. For example, machine learning is a good option to handle situations like:

- Hand-written rules and equations are too complex—as in face recognition and speech recognition.
- The rules of a task are constantly changing—as in fraud detection from transaction records.
- The nature of the data keeps changing, and the program needs to adapt—as in automated trading, energy demand forecasting, and predicting shopping trends.

Energy examples

Optimizing HVAC Energy Usage in Large Buildings



The heating, ventilation, and air-conditioning (HVAC) systems in office buildings, hospitals, and other large-scale commercial buildings are often inefficient because they do not take into account changing weather patterns, variable energy costs, or the building's thermal properties.

Building IQ's cloud-based software platform addresses this problem. The platform uses advanced algorithms and machine learning methods to continuously process gigabytes of information from power meters, thermometers, and HVAC pressure sensors, as well as weather and energy cost. In particular, machine learning is used to segment data and determine the relative contributions of gas, electric, steam, and solar power to heating and cooling processes. The building IQ platform reduces HVAC energy consumption in large-scale commercial buildings by 10% - 25% during normal operation.

10.7. Machine Learning Challenges

Most machine learning challenges relate to handling data and finding the right model.

Data comes in all shapes and sizes. Real-world datasets can be messy, incomplete, and in a variety of formats. We might just have simple numeric data. But sometimes it is a combination of several different data types, such as sensor signals, text, and streaming images from a camera.

Preprocessing your data might require specialized knowledge and tools. For example, to select features to train an object detection algorithm requires specialized knowledge of image processing. Different types of data require different approaches to preprocessing.

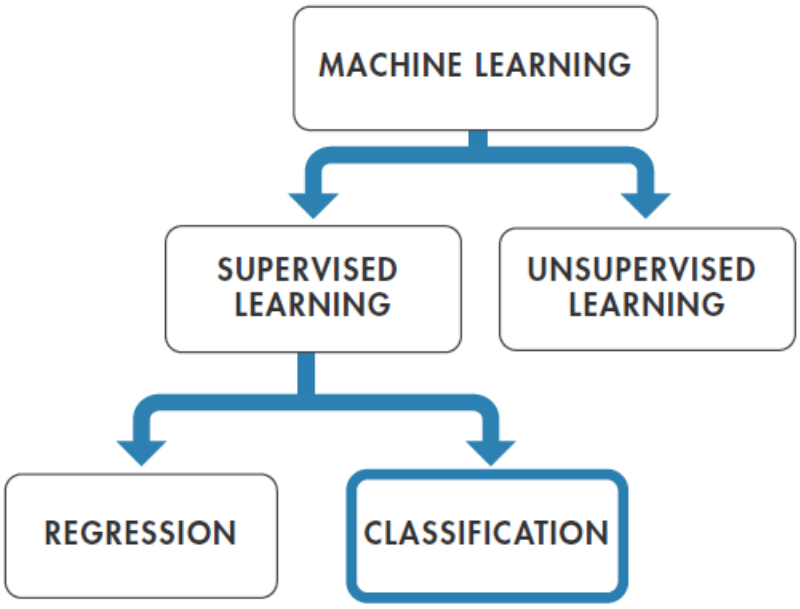
It takes time to find the best model to fit the data. Choosing the right model is a balancing act. Highly flexible models tend to overfit data by modeling minor variations that could be noise. On the other hand, simple models may assume too much. There are always tradeoffs between model speed, accuracy, and complexity.

A systematic workflow is needed to know how to perform the tasks.

Every machine learning workflow begins with three questions:

- What kind of data are you working with?
- What insights do you want to get from it?
- How and where will those insights be applied?

Answers to these questions help to decide whether to use supervised or unsupervised learning.



Choose supervised learning if we need to train a model to make a prediction--for example, the future value of a continuous variable, such as temperature or a stock price, or a classification—for example, identify makes of cars from webcam video footage.

Choose unsupervised learning if we need to explore your data and want to train a model to find a good internal representation, such as splitting data up into clusters.

Workflow:

1. ACCESS and load the data.

2. PREPROCESS the data.

3. DERIVE features using the preprocessed data.

4. TRAIN models using the features derived in step 3.

5. ITERATE to find the best model.

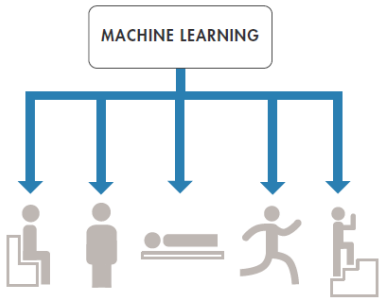
6. INTEGRATE the best-trained model into a production system.

10.8. Training a Model to Classify Physical Activities

This example is based on a cell phone health-monitoring app. The input consists of three-axial sensor data from the phone's accelerometer and gyroscope. The responses, (or output), are the activities performed–walking, standing, running, climbing stairs, or lying down.

We want to use the input data to train a classification model to identify these activities. Since our goal is classification, we'll be applying supervised learning.

The trained model (or classifier) will be integrated into an app to help users track their activity levels throughout the day.



10.8.1. Step One: Load the Data

To load data from the accelerometer and gyroscope we do the following:

1. Sit down holding the phone, log data from the phone, and store it in a text file labeled “Sitting.”

2. Stand up holding the phone, log data from the phone, and store it in a second text file labeled “Standing.”

3. Repeat the steps until we have data for each activity we want to classify.

We store the labeled data sets in a text file. A flat file format such as text or CSV is easy to work with and makes it straightforward to import data.

Machine learning algorithms aren’t smart enough to tell the difference between noise and valuable information. Before using the data for training, we need to make sure it’s clean and complete.

10.8.2. Step Two: Preprocess the Data

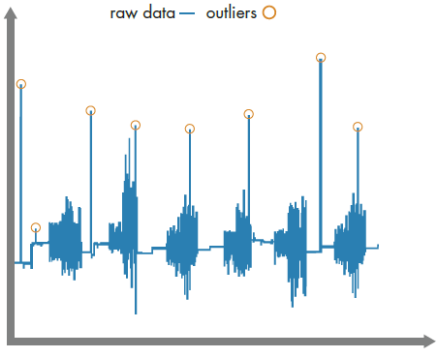
We import the data into MATLAB and plot each labeled set. To preprocess the data we do the following:

1. Look for outliers–data points that lie outside the rest of the data.

We must decide whether the outliers can be ignored or whether they indicate a phenomenon that the model should account for. In our example, they can safely be ignored (it turns out that we moved unintentionally while recording the data).

2. Check for missing values (perhaps we lost data because the connection dropped during recording).

We could simply ignore the missing values, but this will reduce the size of the data set. Alternatively, we could substitute approximations for the missing values by interpolating or using comparable data from another sample.



Outliers in the activity-tracking data.

In many applications, outliers provide crucial information. For example, in a credit card fraud detection app, they indicate purchases that fall outside a customer's usual buying patterns.

3. Remove gravitational effects from the accelerometer data so that our algorithm will focus on the movement of the subject, not the movement of the phone. A simple high-pass filter such as a biquad filter is commonly used for this.

4. Divide the data into two sets. We save part of the data for testing (the test set) and use the rest (the training set) to build models. This is referred to as holdout and is a useful cross-validation technique.

By testing your model against data that wasn't used in the modeling process, you see how it will perform with unknown data.

10.8.3. Step Three: Derive Features

Deriving features (also known as feature engineering or feature extraction) is one of the most important parts of machine learning. It turns raw data into information that a machine learning algorithm can use.

For the activity tracker, we want to extract features that capture the frequency content of the accelerometer data. These features will help the algorithm distinguish between walking (low frequency) and running (high frequency). We create a new table that includes the selected features.

Use feature selection to:

- Improve the accuracy of a machine learning algorithm
- Boost model performance for high-dimensional data sets

- Improve model interpretability
- Prevent overfitting

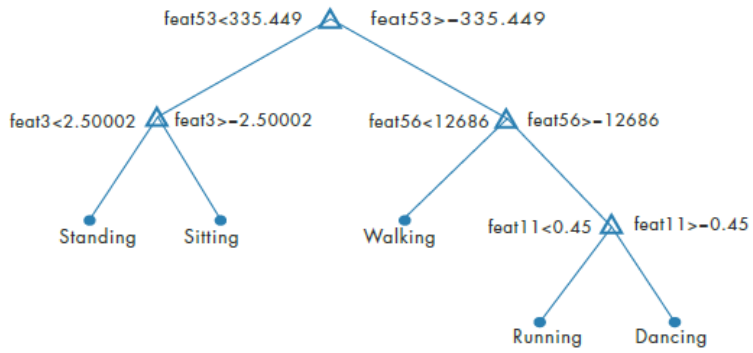
The number of features that you could derive is limited only by your imagination. However, there are a lot of techniques commonly used for different types of data.

Data Type	Feature Selection Task	Techniques
Sensor data	Extract signal properties from raw sensor data to create higher-level information	**Peak analysis** – perform an fft and identify dominant frequencies **Pulse and transition metrics** – derive signal characteristics such as rise time, fall time, and settling time **Spectral measurements** – plot signal power, bandwidth, mean frequency, and median frequency
Image and video data	Extract features such as edge locations, resolution, and color	**Bag of visual words** – create a histogram of local image features, such as edges, corners, and blobs **Histogram of oriented gradients (HOG)** – create a histogram of local gradient directions **Minimum eigenvalue algorithm** – detect corner locations in images **Edge detection** – identify points where the degree of brightness changes sharply
Transactional data	Calculate derived values that enhance the information in the data	**Timestamp decomposition** – break timestamps down into components such as day and month **Aggregate value calculation** – create higher-level features such as the total number of times a particular event occurred

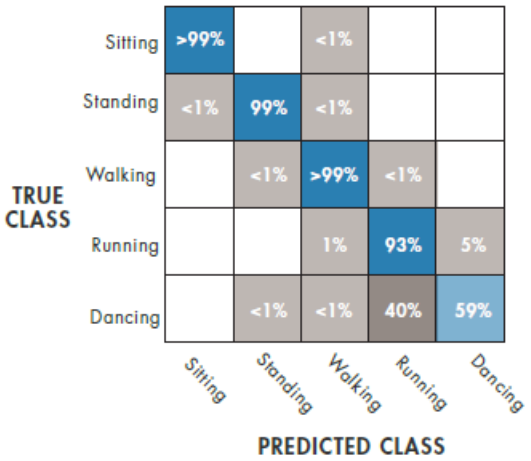
10.8.4. Step Four: Build and Train the Model

When building a model, it's a good idea to start with something simple; it will be faster to run and easier to interpret.

We start with a basic decision tree:

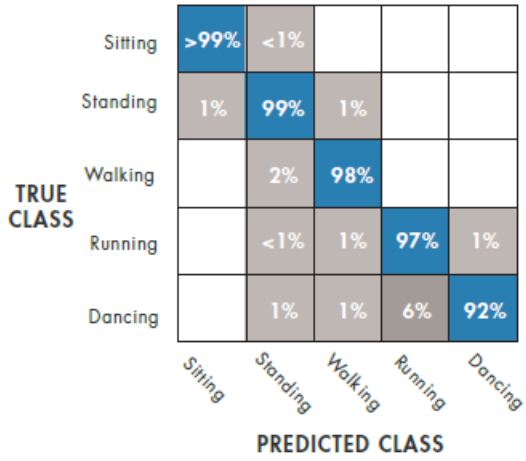


To see how well it performs, we plot the confusion matrix, a table that compares the classifications made by the model with the actual class labels that we created in step 1.



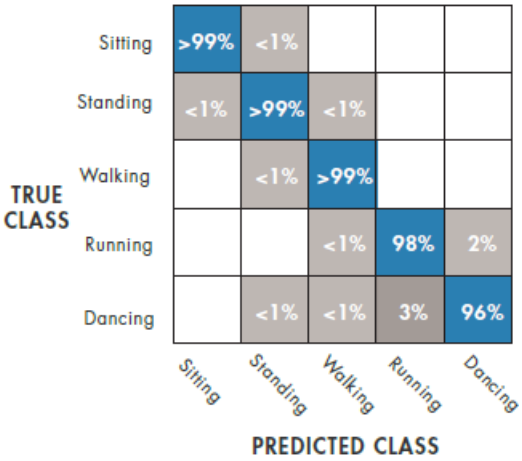
The confusion matrix shows that our model is having trouble distinguishing between dancing and running. Maybe a decision tree doesn't work for this type of data. We'll try a few different algorithms.

We start with a K-nearest neighbors (KNN), a simple algorithm that stores all the training data, compares new points to the training data, and returns the most frequent class of the "K" nearest points. That gives us 98% accuracy compared to 94.1% for the simple decision tree. The confusion matrix looks better, too:



However, KNNs take a considerable amount of memory to run, since they require all the training data to make a prediction.

We try a linear discriminant model, but that doesn't improve the results. Finally, we try a multiclass support vector machine (SVM). The SVM does very well—we now get 99% accuracy:



We achieved our goal by iterating on the model and trying different algorithms. If our classifier still couldn't reliably differentiate between dancing and running, we'd look into ways to improve the model.

10.8.5. Step Five: Improve the Model

Improving a model can take two different directions: make the model simpler or add complexity.

Simplify

First, we look for opportunities to reduce the number of features. Popular feature reduction techniques include:

- Correlation matrix – shows the relationship between variables, so that variables (or features) that are not highly correlated can be removed.
- Principal component analysis (PCA) – eliminates redundancy by finding a combination of features that captures key distinctions between the original features and brings out strong patterns in the dataset.
- Sequential feature reduction – reduces features iteratively on the model until there is no improvement in performance.

Next, we look at ways to reduce the model itself. We can do this by:

- Pruning branches from a decision tree
- Removing learners from an ensemble

A good model includes only the features with the most predictive power. A simple model that generalizes well is better than a complex model that may not generalize or train well to new data.

In machine learning, as in many other computational processes, simplifying the model makes it easier to understand, more robust, and more computationally efficient.

Add Complexity

If our model can't differentiate dancing from running because it is over-generalizing, then we need find ways to make it more fine-tuned. To do this we can either:

Use model combination – merge multiple simpler models into a larger model that is better able to represent the trends in the data than any of the simpler models could on their own.

Add more data sources – look at the gyroscope data as well as the acceleromter data. The gyroscope records the orientation of the cell phone during activity. This data might provide unique signatures for the different activities; for example, there might be a combination of acceleration and rotation that's unique to running.

Once we've adjusted the model, we validate its performance on the test data that we set aside during preprocessing.

If the model can reliably classify activities on the test data, we're ready to move it to the phone and start tracking.

10.9. Applying Unsupervised Learning

Unsupervised learning is useful when you want to explore your data but don't yet have a specific goal or are not sure what information the data contains. It's also a good way to reduce the dimensions of your data.

Unsupervised Learning Techniques

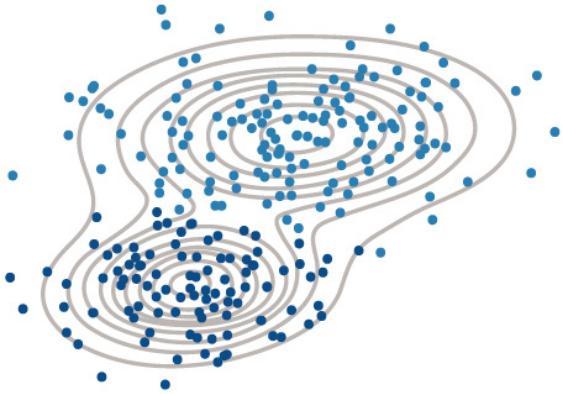
As we saw in section 1, most unsupervised learning techniques are a form of cluster analysis.

In cluster analysis, data is partitioned into groups based on some measure of similarity or shared characteristic. Clusters are formed so that objects in the same cluster are very similar and objects in different clusters are very distinct.

Clustering algorithms fall into two broad groups:

- Hard clustering, where each data point belongs to only one cluster
- Soft clustering, where each data point can belong to more than one cluster

You can use hard or soft clustering techniques if you already know the possible data groupings.



Gaussian mixture model used to separate data into two clusters.

you don't yet know how the data might be grouped:

Use self-organizing feature maps or hierarchical clustering to look for possible structures in the data.

Use cluster evaluation to look for the "best" number of groups for a given clustering algorithm.

10.9.1. Common Hard Clustering Algorithms

k-Means

How it Works Partitions data into k number of mutually exclusive clusters. How well a point fits into a cluster is determined by the distance from that point to the cluster's center.

Best Used...

When the number of clusters is known

For fast clustering of large data sets

k-Medoids

How It Works Similar to k-means, but with the requirement that the cluster centers coincide with points in the data.

Best Used...

When the number of clusters is known

For fast clustering of categorical data

To scale to large data sets

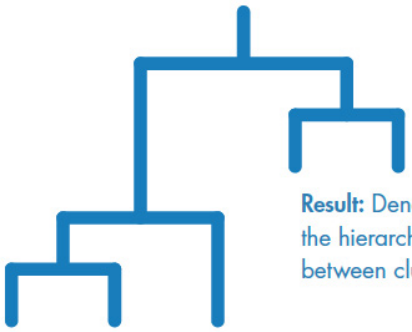
Hierarchical Clustering

How it Works Produces nested sets of clusters by analyzing similarities between pairs of points and grouping objects into a binary, hierarchical tree.

Best Used...

When you don't know in advance how many clusters are in your data

You want visualization to guide your selection



Result: Dendrogram showing the hierarchical relationship between clusters

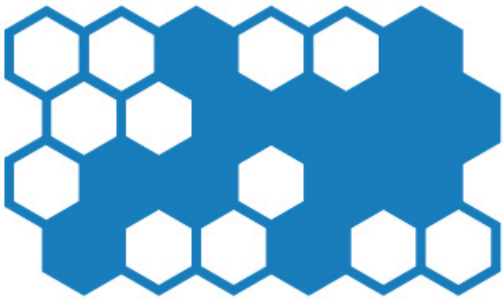
Self-Organizing Map

How It Works Neural-network based clustering that transforms a dataset into a topology-preserving 2D map.

Best Used...

To visualize high-dimensional data in 2D or 3D

To deduce the dimensionality of data by preserving its topology (shape)



Result: Lower-dimensional (typically 2D) representation

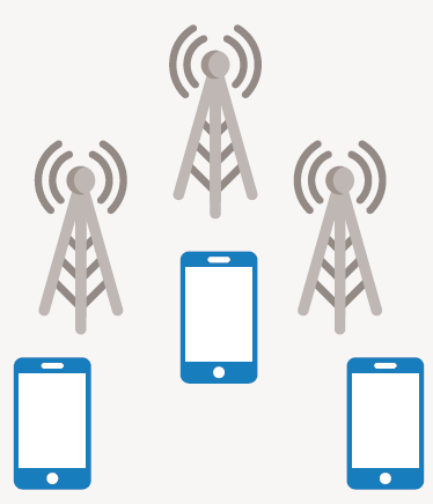
Example: Using k-Means Clustering to Site Cell Phone Towers

A cell phone company wants to know the number and placement of cell phone towers that will provide the most reliable service. For optimal signal reception, the towers must be located within clusters of people.

The workflow begins with an initial guess at the number of clusters that will be needed. To evaluate this guess, the engineers compare service with three towers and four towers to see how well they're able to cluster for each scenario (in other words, how well the towers provide service).

A phone can only talk to one tower at a time, so this is a hard clustering problem. The team uses k-means clustering because k-means treats each observation in the data as an object having a location in space. It finds a partition in which objects within each cluster are as close to each other as possible and as far from objects in other clusters as possible.

After running the algorithm, the team can accurately determine the results of partitioning the data into three and four clusters.



10.9.2. Common Soft Clustering Algorithms

Fuzzy c-Means

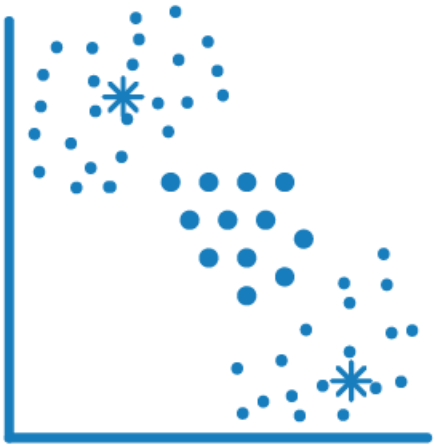
How it Works Partition-based clustering when data points may belong to more than one cluster.

Best Used...

When the number of clusters is known

For pattern recognition

When clusters overlap



Result: Cluster centers (similar to k-means) but with fuzziness so that points may belong to more than one cluster

Gaussian Mixture Model

How It Works Partition-based clustering where data points come from different multivariate normal distributions with certain probabilities.

Best Used...

When a data point might belong to more than one cluster

When clusters have different sizes and correlation structures within them

Result: A model of Gaussian distributions that give probabilities of a point being in a cluster

Example: Using Fuzzy c-Means Clustering to Analyze Gene Expression Data

A team of biologists is analyzing gene expression data from microarrays to better understand the genes involved in normal and abnormal cell division. (A gene is said to be “expressed” if it is actively involved in a cellular function such as protein production.)

The microarray contains expression data from two tissue samples. The researchers want to compare the samples to determine whether certain patterns of gene expression are implicated in cancer proliferation.

After preprocessing the data to remove noise, they cluster the data. Because the same genes can be involved in several biological processes, no single gene is likely to belong to one cluster only. The researchers apply a fuzzy c-means algorithm to the data. They then visualize the clusters to identify groups of genes that behave in a similar way.

10.9.3. Improving Models with Dimensionality Reduction

Machine learning is an effective method for finding patterns in

big datasets. But bigger data brings added complexity.

As datasets get bigger, you frequently need to reduce the

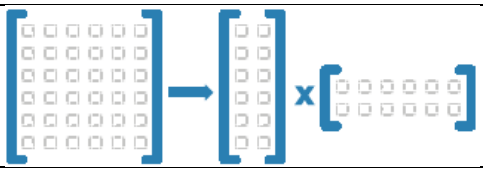
number of features, or dimensionality.

Example: EEG Data Reduction

Suppose you have electroencephalogram (EEG) data that captures electrical activity of the brain, and you want to use this data to predict a future seizure. The data was captured using dozens of leads, each corresponding to a variable in your original dataset. Each of these variables contains noise. To make your prediction algorithm more robust, you use dimensionality reduction techniques to derive a smaller number of features. Because these features are calculated from multiple sensors, they will be less susceptible to noise in an individual sensor than would be the case if you used the raw data directly.

10.9.4. Common Dimensionality Reduction Techniques

The three most commonly used dimensionality reduction techniques are:

Principal component analysis (PCA)—performs a linear transformation on the data so that most of the variance or information in your high-dimensional dataset is captured by the first few principal components. The first principal component will capture the most variance, followed by the second principal component, and so on.	
Factor analysis—identifies underlying correlations between variables in your dataset to provide a representation in terms of a smaller number of unobserved latent, or common, factors.	
Nonnegative matrix factorization—used when model terms must represent nonnegative quantities, such as physical quantities.	

10.9.5. Using Principal Component Analysis

In datasets with many variables, groups of variables often move together. PCA takes advantage of this redundancy of information by generating new variables via linear combinations of the original variables so that a small number of new variables captures most of the information.

Each principal component is a linear combination of the original variables. Because all the principal components are orthogonal to each other, there is no redundant information.

Example: Engine Health Monitoring

You have a dataset that includes measurements for different sensors on an engine (temperatures, pressures, emissions, and so on). While much of the data comes from a healthy engine, the sensors have also captured data from the engine when it needs maintenance.

You cannot see any obvious abnormalities by looking at any individual sensor. However, by applying PCA, you can transform this data so that most variations in the sensor measurements are captured by a small number of principal components. It is easier to distinguish between a healthy and unhealthy engine by inspecting these principal components than by looking at the raw sensor data.

10.9.6. Using Factor Analysis

Your dataset might contain measured variables that overlap, meaning that they are dependent on one another. Factor analysis lets you fit a model to multivariate data to estimate this sort of interdependence.

In a factor analysis model, the measured variables depend on a smaller number of unobserved (latent) factors. Because each factor might affect several variables, it is known as a common factor. Each variable is assumed to be dependent on a linear combination of the common factors.

Example: Tracking Stock Price Variation

Over the course of 100 weeks, the percent change in stock prices has been recorded for ten companies. Of these ten, four are technology companies, three are financial, and a further three are retail. It seems reasonable to assume that the stock prices for companies in the same sector will vary together as economic conditions change. Factor analysis can provide quantitative evidence to support this premise.

10.9.7. Using Nonnegative Matrix Factorization

This dimension reduction technique is based on a low-rank approximation of the feature space. In addition to reducing the number of features, it guarantees that the features are nonnegative, producing models that respect features such as the nonnegativity of physical quantities.

Example: Text Mining

Suppose you want to explore variations in vocabulary and style among several web pages. You create a matrix where each row corresponds to an individual web page and each column corresponds to a word (“the”,”a”,”we”, and so on). The data will be the number of times a particular word occurs on a particular page.

Since there more than a million words in the English language, you apply nonnegative matrix factorization to create an arbitrary number of features that represent higher-level concepts rather than individual words. These concepts make it easier to distinguish between, say, news, educational content, and online retail content.

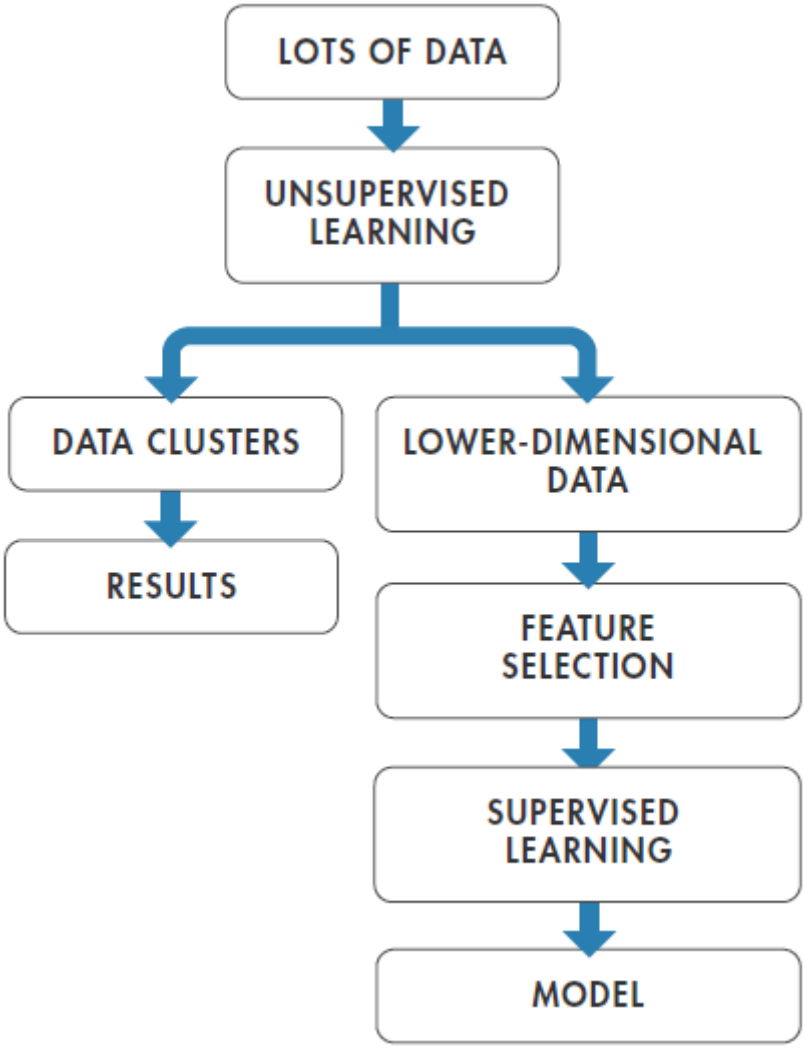
10.9.8. Next Steps

In this section we took a closer look at hard and soft clustering algorithms for unsupervised learning, offered some tips on selecting the right algorithm for your data, and showed how reducing the number of features in your dataset improves model performance. As for your next steps:

Unsupervised learning might be your end goal. For example, if you are doing market research and want to segment consumer groups to target based on web site behavior, a clustering algorithm will almost certainly give you the results you’re looking for.

On the other hand, you might want to use unsupervised learning as a preprocessing step for supervised learning. For example, apply clustering techniques to derive a smaller number of features, and then use those features as inputs for training a classifier.

In section 4 we’ll explore supervised learning algorithms and techniques, and see how to improve models with feature selection, feature reduction, and parameter tuning.



10.10. Application: Daily load curves analysis and clustering

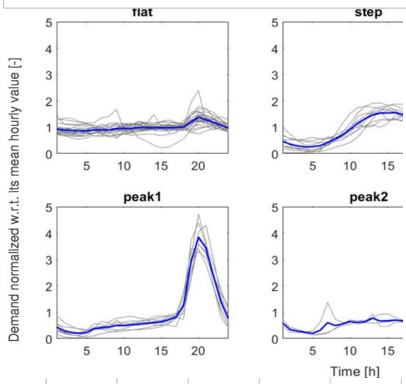
For that we need the “Statistics and Machine Learning Toolbox” of Matlab.
Let’s take the data given in the reference Lorenzoni et all, 2020, DOI: 10.1016/j.esd.2020.10.001
They present the load curves of different installations in Honduran communities from various studies.
The load curves gives power consumption over 24 hours.

Data from Lorenzoni L., Cherubini P., Fioriti D., Poli D., Micangeli A., Giglioli R., Classification and modeling of load profiles of isolated mini-grids in developing countries: a data-driven appr

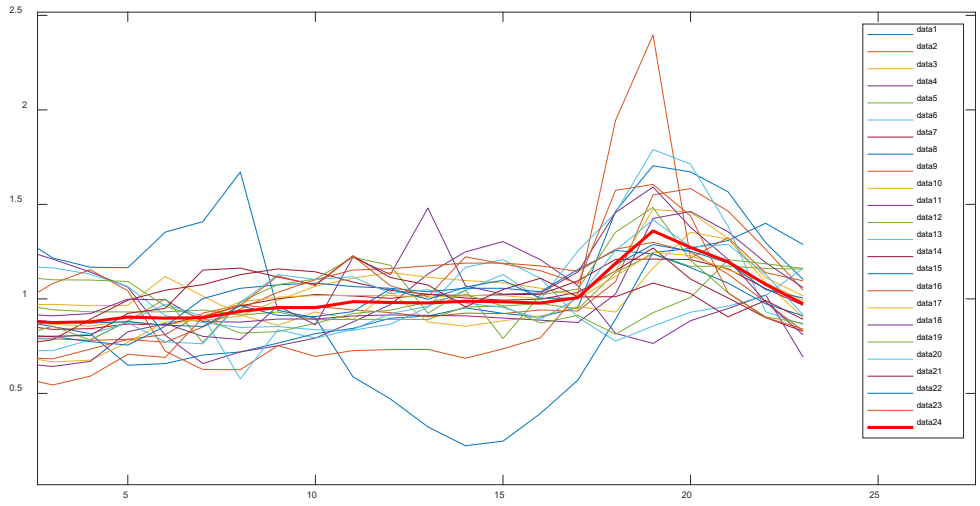
Mean and standard deviation of each cluster average profile [p.u.]

Time	flat		step		step-peak		Peak		outliers	
	mean	std	mean	std	mean	std	mean	std	mean	std
00:00	0.934	0.161	0.461	0.287	0.461	0.167	0.494	0.208	1.153	0.241
01:00	0.900	0.184	0.356	0.229	0.366	0.158	0.315	0.148	0.826	0.061
02:00	0.881	0.186	0.295	0.191	0.346	0.158	0.257	0.121	0.633	0.220
03:00	0.871	0.177	0.256	0.187	0.332	0.147	0.219	0.095	0.597	0.247
04:00	0.875	0.159	0.263	0.199	0.356	0.120	0.228	0.083	0.587	0.260
05:00	0.904	0.126	0.302	0.174	0.459	0.148	0.357	0.153	0.616	0.239
06:00	0.899	0.145	0.420	0.148	0.531	0.132	0.469	0.331	0.733	0.204
07:00	0.905	0.160	0.551	0.140	0.640	0.167	0.465	0.142	0.717	0.118
08:00	0.941	0.209	0.708	0.216	0.756	0.101	0.535	0.148	0.601	0.101
09:00	0.956	0.121	0.915	0.291	0.907	0.133	0.565	0.164	0.573	0.164
10:00	0.959	0.126	1.152	0.335	1.091	0.271	0.570	0.153	0.615	0.212
11:00	0.987	0.164	1.320	0.314	1.151	0.317	0.584	0.140	0.683	0.220
12:00	0.980	0.149	1.462	0.290	1.171	0.271	0.642	0.176	0.794	0.183
13:00	0.974	0.193	1.537	0.278	1.131	0.186	0.648	0.188	0.892	0.100
14:00	0.987	0.198	1.541	0.266	1.104	0.139	0.670	0.184	0.962	0.132
15:00	0.985	0.200	1.550	0.212	1.118	0.157	0.715	0.132	1.042	0.223
16:00	0.985	0.158	1.481	0.200	1.086	0.230	0.758	0.146	1.010	0.147
17:00	1.006	0.128	1.626	0.503	1.177	0.248	1.077	0.406	0.917	0.124
18:00	1.194	0.264	1.584	0.484	1.707	0.453	2.246	1.130	0.828	0.268
19:00	1.364	0.328	1.663	0.300	2.391	0.221	3.226	0.953	0.909	0.247
20:00	1.280	0.209	1.556	0.345	2.284	0.233	3.634	0.550	1.527	0.394
21:00	1.190	0.166	1.362	0.440	1.689	0.263	2.737	0.704	2.383	0.352
22:00	1.075	0.127	0.984	0.474	1.095	0.302	1.665	0.565	2.519	0.655
23:00	0.968	0.140	0.654	0.375	0.653	0.193	0.925	0.365	1.887	0.679

Each load profile reported in the other sheets of this excel
demand a day; then all the normalized profiles have been c
method, using the Euclidean metric and the Ward linkage r
the profile and the clusters, as explained in the paper.
Here, the average profiles for each cluster, with the relative
are reported, as per the figure below (Figure 1 of the paper



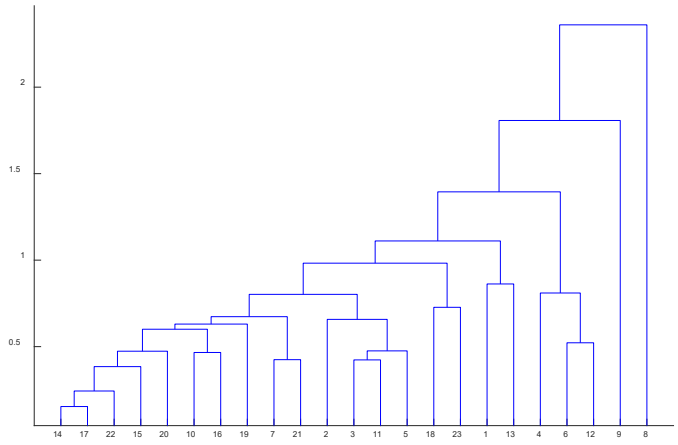
The idea here is to apply some of the methods we studied in order to extract information from these curves.
First, we will load the data from CSV files, put them in a certain form (matrix), normalize using mean value.
We will display them and try to figure out if there are common features.
Then we will try to cluster them in families.

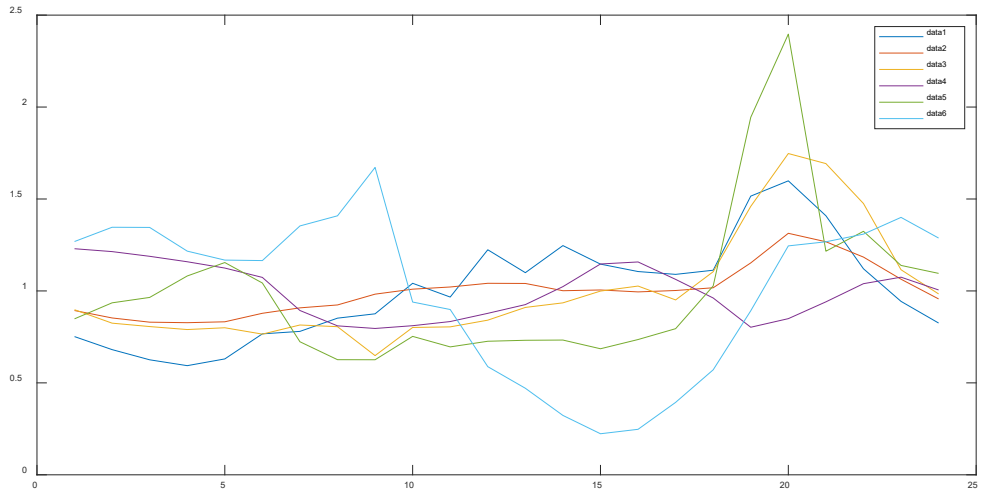


23 data normalized of load curve and their mean curve

Creating dendrogram using

generates a dendrogram plot of the hierarchical binary cluster tree. A dendrogram consists of many *U*-shaped lines that connect data points in a hierarchical tree. The height of each *U* represents the distance between the two data points being connected.





Clustering data into 6 clusters using hierarchical clustering

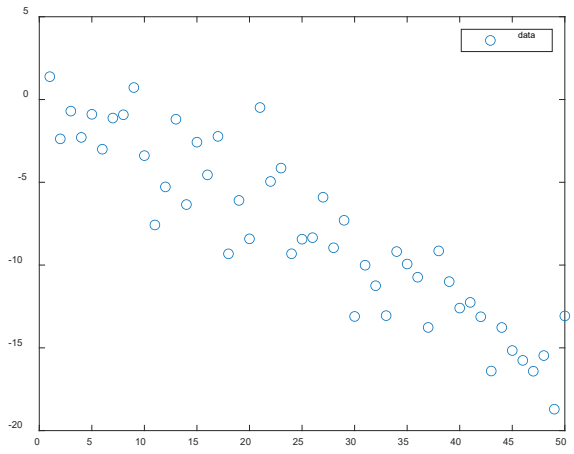
```
close all, clear all;
crv_table = readtable("flat.csv");
crv = table2array(crv_table(:,2:24));
%plot (crv(2:24,1), crv(2:,2))
h=[0:23];
% y=crv(:,2);
% figure (1)
% plot (h,y);
figure (2)
stackedplot(crv);
crvn = crv./mean(crv);
figure(3)
plot(h, crvn);
crv_mean = mean(crvn')';
hold on
plot(h, crv_mean, 'r', 'LineWidth',2);
legend()

% create dendogram from data
```

```
data_for_clustering = crvn';
Y = pdist(data_for_clustering);
Z = linkage(Y,"average");
dendrogram(Z)

% get clusters
n_clusters = 6;
T = cluster(Z,"maxclust",n_clusters);

clusters = zeros(size(crvn, 1), n_clusters);

for c = 1:n_clusters
    clusters(:, c) = mean(crvn(:, T==c), 2);
end

figure
plot(clusters)
legend()
```

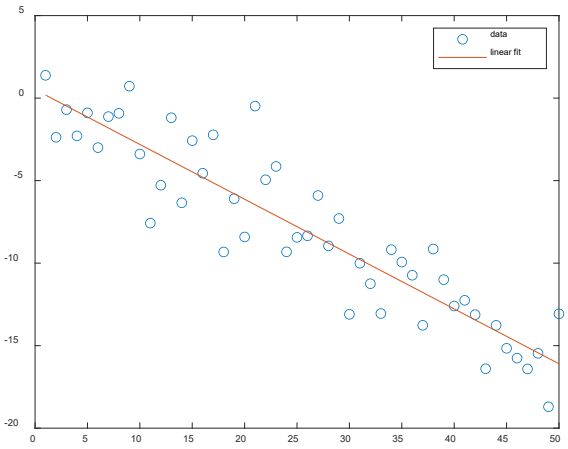
10.1. Linear regression models and the least squares method

10.1.1. Introduction example

Given an amount of data we can try to figure out a model that represent them. Many physics model have linear variation. Thus, a simple linear regression can be applied.



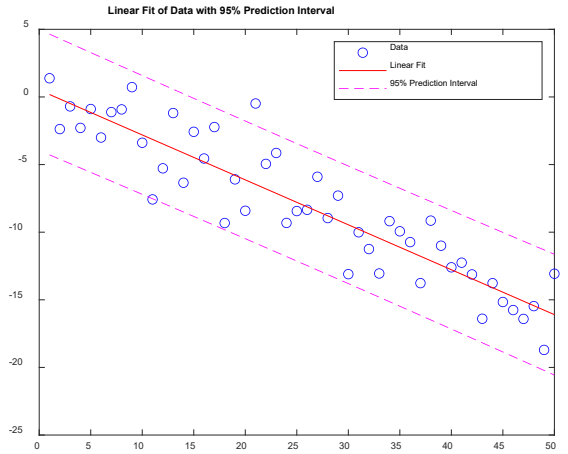
Data seems to have linear variation. So, if we apply a linear regression:




```
close all, clear all;
rng(1);
x = 1:50;
y = -0.3*x + 2*randn(1,50);
p = polyfit(x,y,1);
f = polyval(p,x);
figure
plot(x,y,'o')
legend('data')
figure
plot(x,y,'o',x,f,'-')
legend('data','linear fit')
```

We notice it can be a good representative of the system. But how good?

We need to know if 95% of the points are close to the approximation, by how much




```
[p,S] = polyfit(x,y,1);
[y_fit,delta] = polyval(p,x,S);plot(x,y,'bo')
hold on
plot(x,y_fit,'r-')
plot(x,y_fit+2*delta,'m--',x,y_fit-2*delta,'m--')
```

```
title('Linear Fit of Data with 95% Prediction Interval')
legend('Data','Linear Fit','95% Prediction Interval')
```

10.1.2. Why regression

The analysis of observational data or data obtained from designed experiments often requires the identification of a statistical model or relationship which captures the underlying structure of the system from which the sample data was drawn. A model is a relation between the variation of one variable (called the dependent or response variable) against that of other variables (called independent or regressor variables). If observations (or data) are taken of both response and regressor variables under various sets of conditions, one can build a mathematical model from this information which can then be used as a predictive tool under different sets of conditions.

How to analyze the relationships among variables and determine a (if not "the") optimal relation, falls under the realm of regression model building or regression analysis.

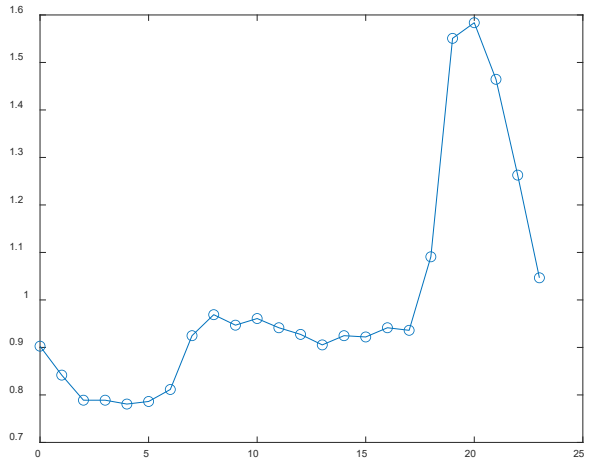
Models can be of two forms:

- Parametric models which can be a single function (or a set of functions) capturing the variation of the response variable in terms of the regressors. The intent is to identify both the model function and determine the values of the parameters of the model along with some indication of their uncertainty
- Nonparametric models where the relationship between response and regressors is such that a mathematical model in the conventional sense is inadequate. Non-parametric models can be treated in the framework of time series models dealing with artificial neural network models (RNN, Long Short Term Memory networks _LSTMs_).

The parameters appearing in parametric models can be estimated in a number of ways, of which ordinary least squares (OLS) is the most common and historically the oldest.
There is a direct link between how the model parameters are estimated and the underlying joint probability distributions of the variables.

10.1. Application: Fitting Daily load curves

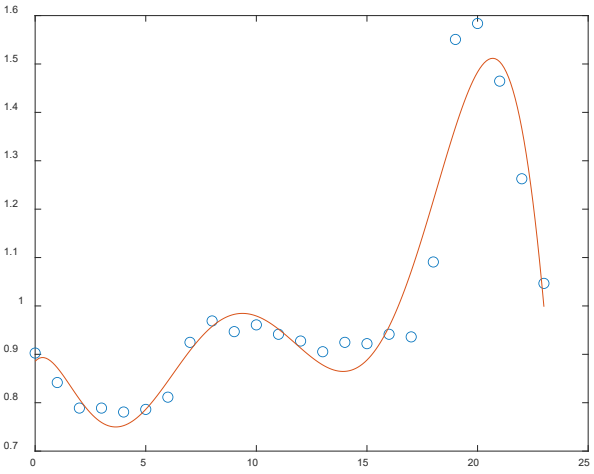
Let us now consider only the curve n°2 of the flat curves file.



Normalized data curve 2

Using polyfit/polyval functions find a Polynomial that fits the curve the better.
Try different Polynomial orders, under 7 and much beyond.

What do you notice? In terms of accuracy, inside and outside the hours range.
How this fitting works?



7th order Polynomial fit

```
close all, clear all;
crv_table = readtable("flat.csv");
crv = table2array(crv_table(:,2:24));
%plot (crv(2:24,1), crv(2:,2))
h=[0:23];
hc=[0:0.1:23];
figure
%stackedplot(crv);
crvn = crv./mean(crv);
figure
```

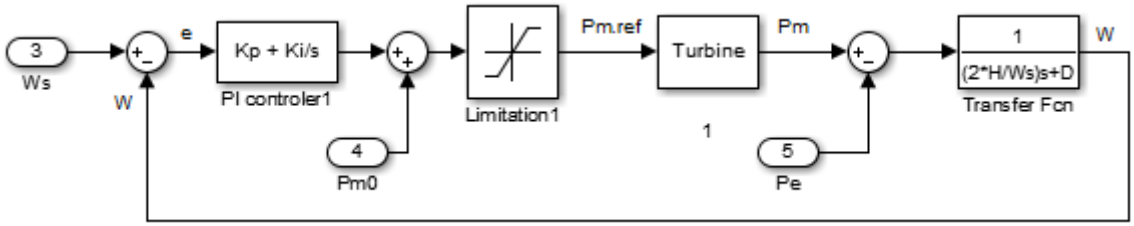
```
y=crvn(:,2);
plot(h, y);
p = polyfit(h,y,7);
y1 = polyval(p,hc);
figure
plot(h,y,'o')
hold on
plot(hc,y1)
hold off
```

11.Control and planning

Control and planning

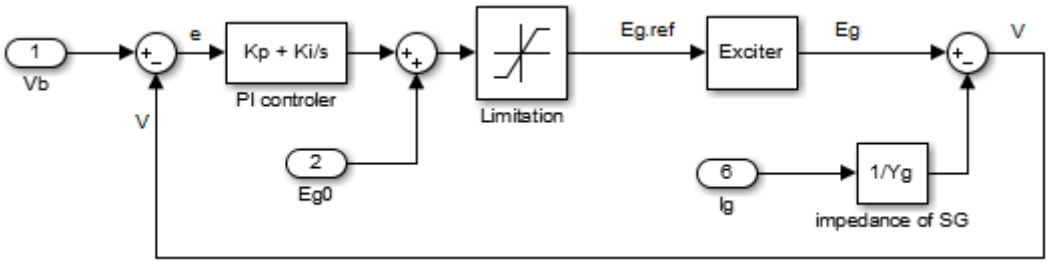
11.1. Control of energy production systems

There are different productions systems. Traditional ones are based on steam turbines that drives Synchronous Generators which are connected to the power grid through power transformers (MV-HV).
The steam turbines take their primer energy from water heated by nuclear power, gas or coil.
The control is done thanks to the different schemes. The driving torque (or mechanical power) through the steam valve.
This allow to control the frequency of the synchronous generator as shown on the figure below.

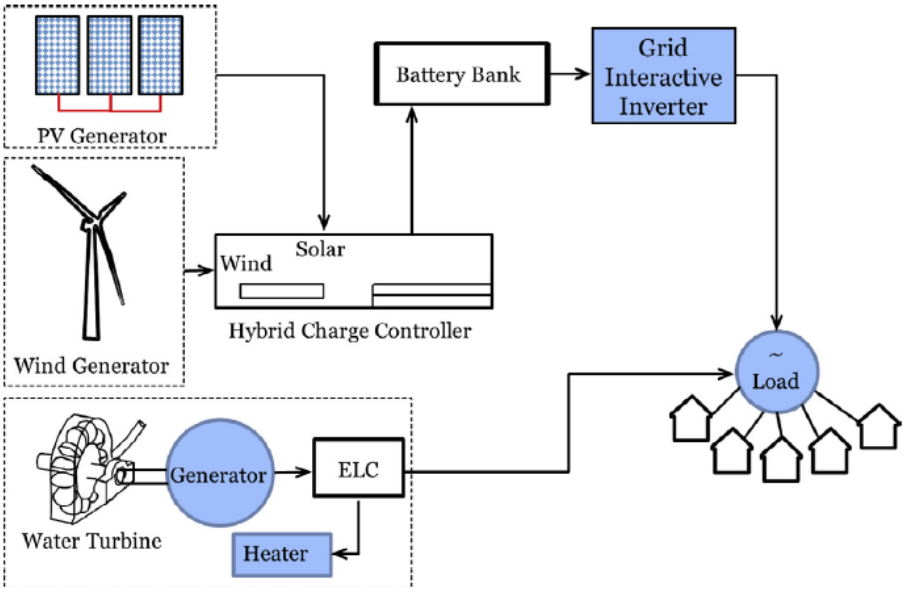


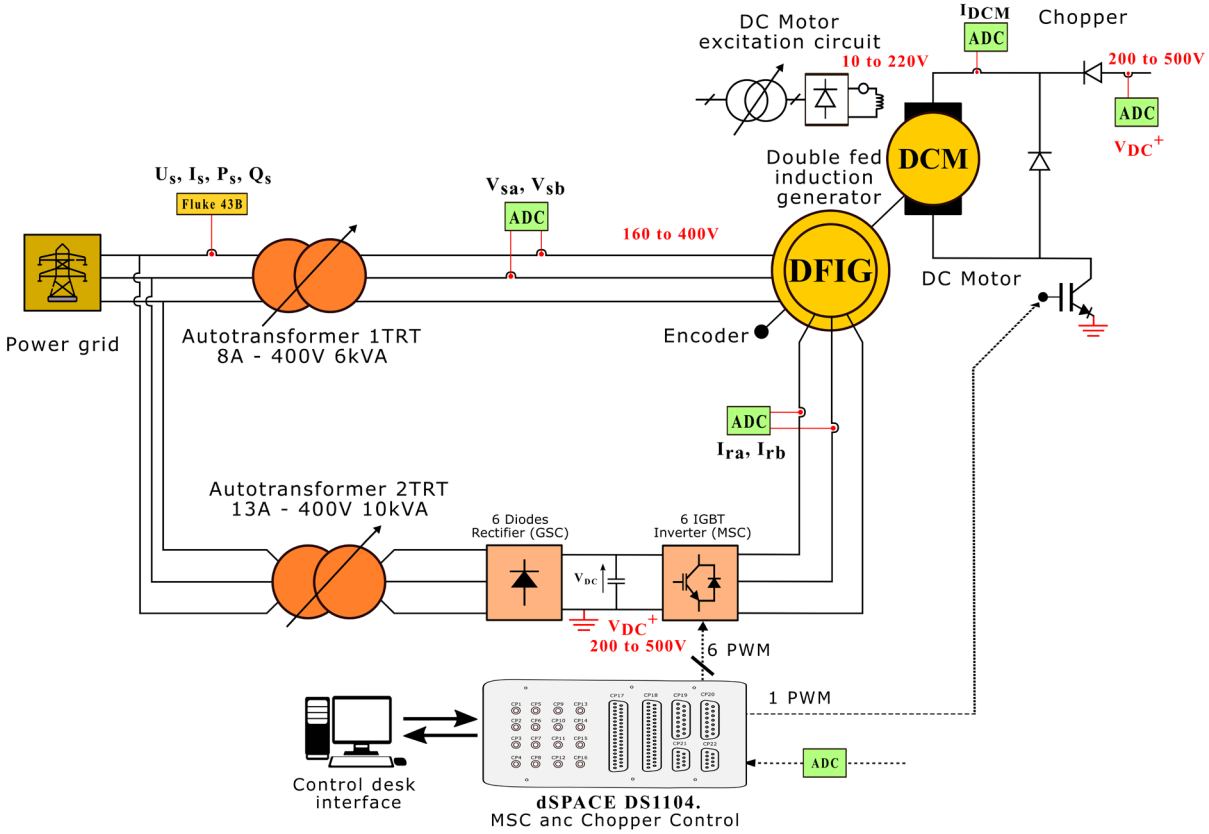
This is called Speed Governor Loop.
It is used to control the frequency of each generator of conventional power plants.

To control the terminal voltage, we need to control the EMF of the generator throur the excitation current Ig. We use the following control scheme.

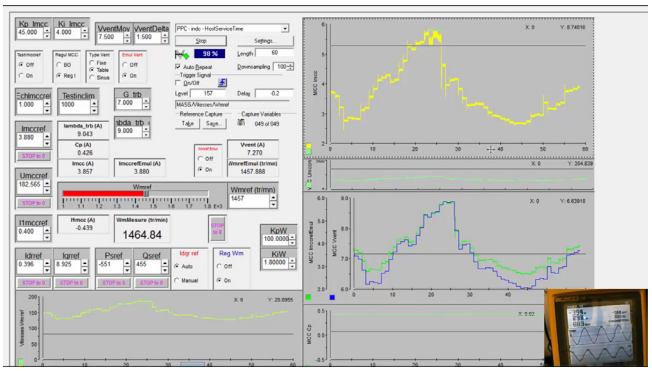


It is called the Automatic Voltage Regulator (AVR). It reacts faster than the Speed Governor Loop.
Both controllers are called P**ower System Stabilizers (PSS)**. They help to keep the balance between the power production and the power demand through the stabilization of the voltage in magnitude and frequency.
As one can see, these are **conventional PI controllers**. They are well used for many decades now in all power plants.

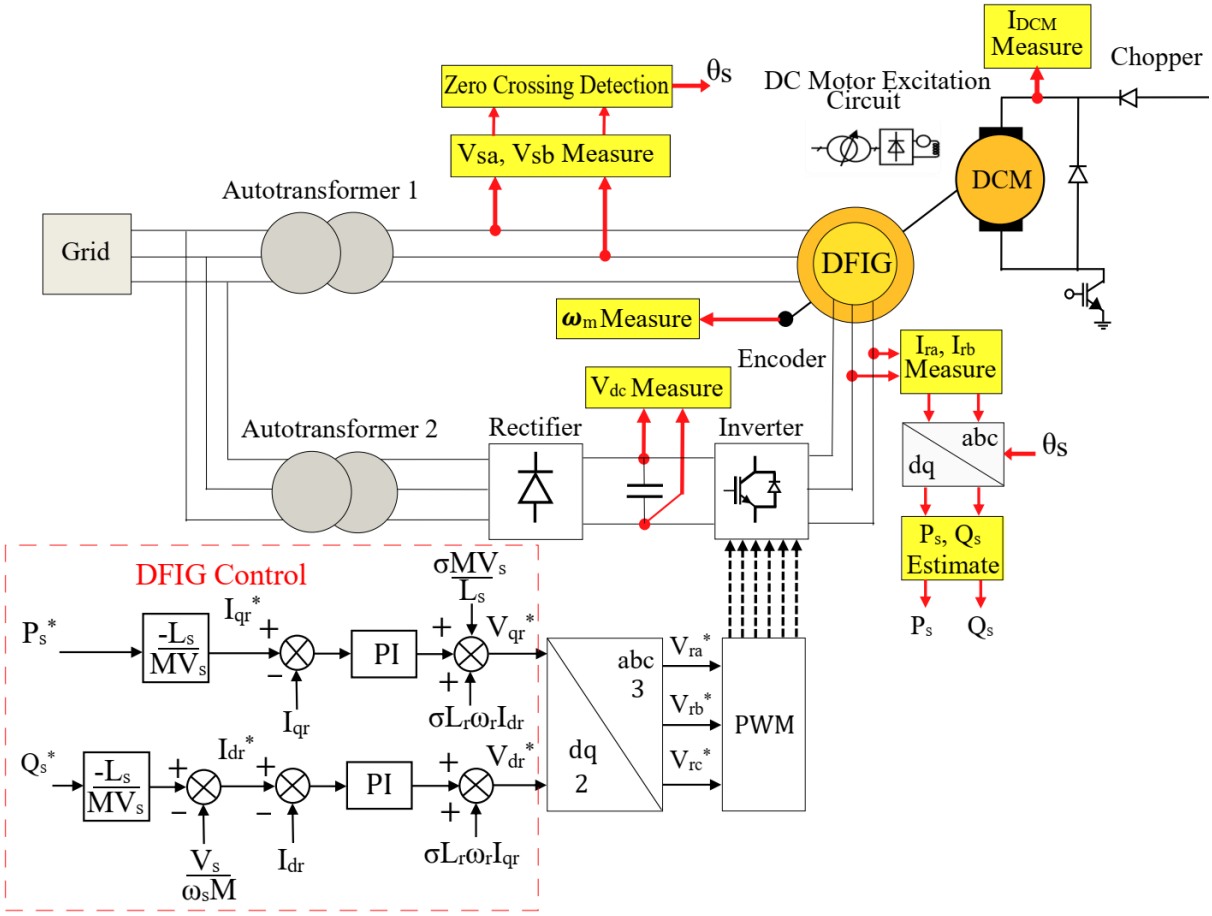




Wind Turbine Emulator, Hardware In th Loop (HIL) DFIG – DCM



- Electrical micro-grids
  - Models and test benches of MPP (Micro Power Plants) and ESS (Energy Storage Systems)
  - DC or AC bus interconnection, interfacing, control, low inertia
- Develop prototypes and their models
- MPP: DFIG and SG with back-to-back converters, single-phase PV GTI, SiC, Grid synchronization
- ESS: Batteries and BMS, micro Pumped-Storage Hydroelectricity (PSH)

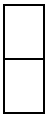


Wind Turbine Emulator, Hardware In th Loop (HIL) DFIG – DCM

Control of active and reactive power injected to the grid

- Electrical micro-grids
- Model parameters Identification and Control (Linear, Non-Lineat)
- Emulate some models in RT (DSP, RPI4): SOC, SOH, study slow/fast dynamic behavior
- Interconnection to DC or AC bus, interfacing thanks to Power Electronics, active filtering
- Control strategies with constraints
    - Low inertia (low ESS)
    - Powerful connection point (V,f)
    - Grid forming / islanding

References

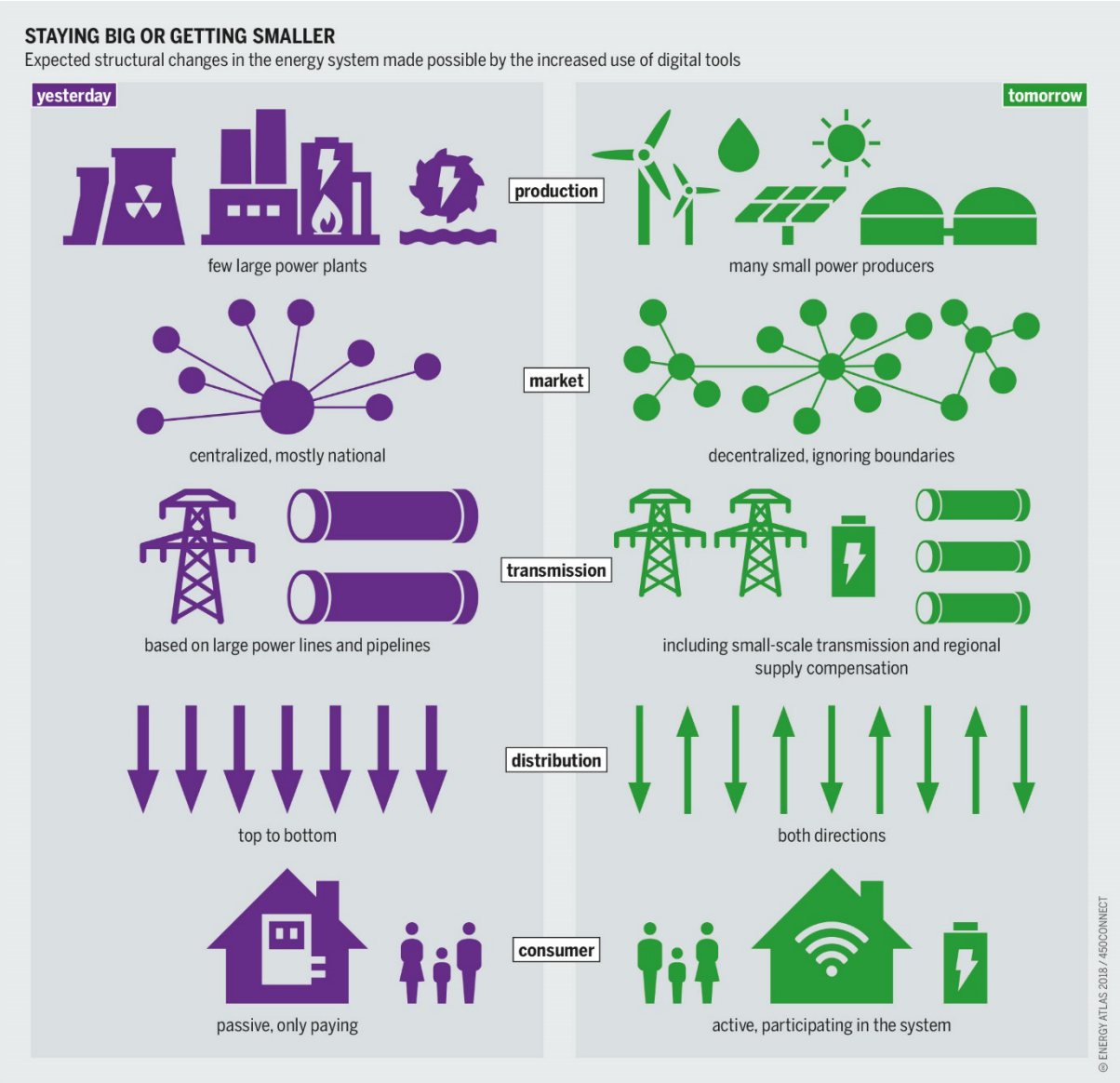


12. Current and future research directions

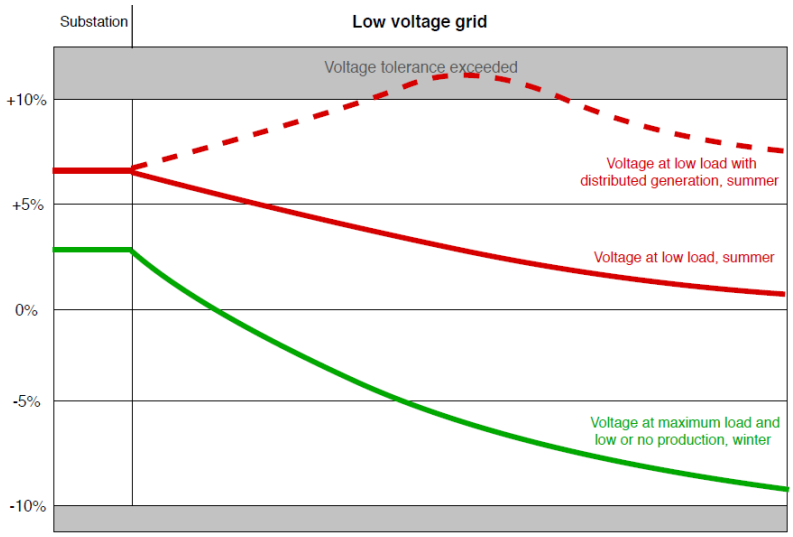
Current and future research directions

12.1. Smart grid, intelligent buildings, generation planning and other innovations

We are moving from traditional centralized energy production systems to distributed ones.
TSO: Transmission System Operators
DSO: Distribution System Operators.
Traditionally, power flows from large scale power plants to users through transmission line then distribution lines.
With consumers being producers (prosumers), the power can flow both ways leading to voltage increase problems that the TSO will not see by its actual metering systems (illustration drawn on whiteboard)



Some aspects of a smart grid that differ from standard power grid *(i-SCOOP, 2018).*



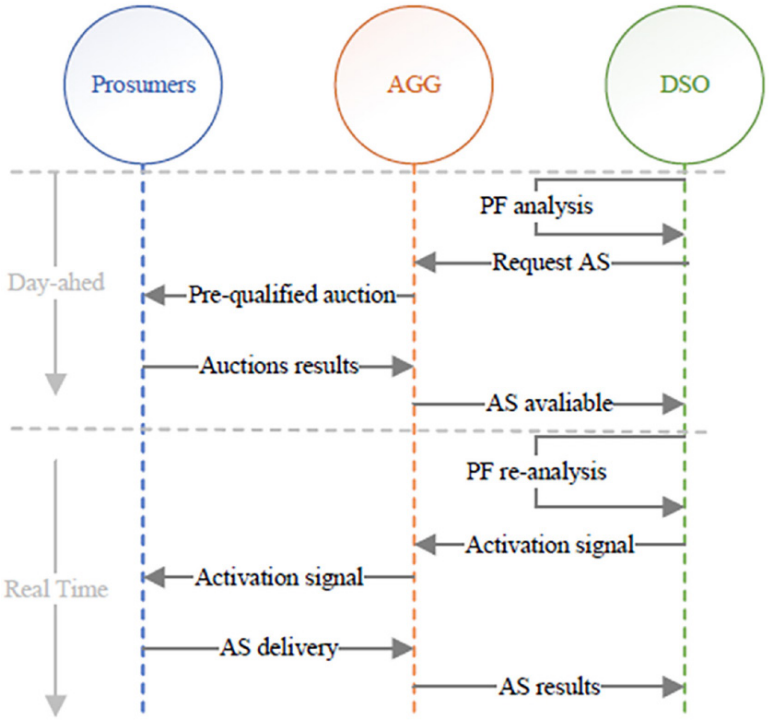
The voltage drop along a transmission line [Ringheim 2020]:

$$\Delta V = \frac{P \cdot R - Q \cdot X}{V}$$

If there is a production of power (negative active power) then the voltage will raise and not drop!

The TSO can't see it from its production side (left).

Interaction scheme [Ricardo 2022]



References

i-SCOOP, 2018. Smart grids: electricity networks and the grid in evolution, URL https://www.i-scoop.eu/industry-4-0/smart-grids-electrical-grid/
Faia, Ricardo & Pinto, Tiago & Lezama, Fernando & Vale, Zita & Corchado, Juan & González Briones, Alfonso. (2022). Prosumers Flexibility as Support for Ancillary Services in Low Voltage Level. ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal. 11. 65-80. 10.14201/adcaij.27896.
Liv Ringheim, Grid Impact from Increased Prosumer Penetration in the Norwegian Distribution Grid, Master's thesis in Energy and Environment, Norwegian University of Science and Technology, February 2020

13. Case studies: Group activity

Work in group, brainstorming to study one of the proposed project
1. Building a smart energy meter
2. DC small connected micro grids with low power transfer and energy / money metering
3. Vehicle counter to measure traffic

13.1. Introduction

Internet of Things applications
- Very useful to study in group
- Learn by making
- Easy and low cost
- Funny, entertainment
- Multiple possibilities

Choice for this course
- Choose 1 among 3 projects to be studied in group of students
- Students virtually meet and cooperate using Slack, Google meet or Microsoft Teams
- Project sliced into technological, managerial and financial aspects
- PDF report to be sent with solutions, different options, part of code

13.2. Study content

Choose one of the proposed project as the case study of the group
You need to brainstorm in group in order to develop the solution
Students meets virtually using Slack, Google meet or Microsoft Teams
Studied aspects:

- Introduction
- Aim, principle
- Targeted people, community state
- Why the project is important
- Is it low cost, affordable, must rise money, big investment
- What aspect of connectivity, IoT and infrastructure it requires
- What are the hardware components needed for the prototype and for the final release, market ready
- What is the software needed for the IoT device, the server, the handled App
- What policies it may affect or require
- What are the impact on the environment, its carbon footprint, sustainability, compare with other solutions

Technological implementation:

- Choose the suitable components
- Point out the internal peripherals and microcontroller used for the IoT device
- What sensors, adaptation board, amplifier, filters, circuitry the device requires
- How to power the embedded device, study the energy scenario, use Lipo batteries, how to charge them, small 6V PV or intermittent +5V flyback power supply of modern phone charger
- What about the case, box, encapsulation, can you make it using a 3D printer, laser cut wood/metal plates, or you need plastic injection molding
- Data transmission, protocol, frequency, distance, communication traffic jam, EMI, multiple addresses and simultaneous connections
- Database, type of DB, instant notification, storage cost, amount of daily data, per month, per year, permanent storage or can be erased on a periodic scheme
- Data access, by the developer, the client (who uses the solution), the user (who is the data provider), platform for data access, web client interface, smartphone App.
- In the report, provide part of the code for the database (PHP, C, MySQL), the microcontroller C code, gather data, loops, counters, send data, receive instructions/data, any relevant part of the solution

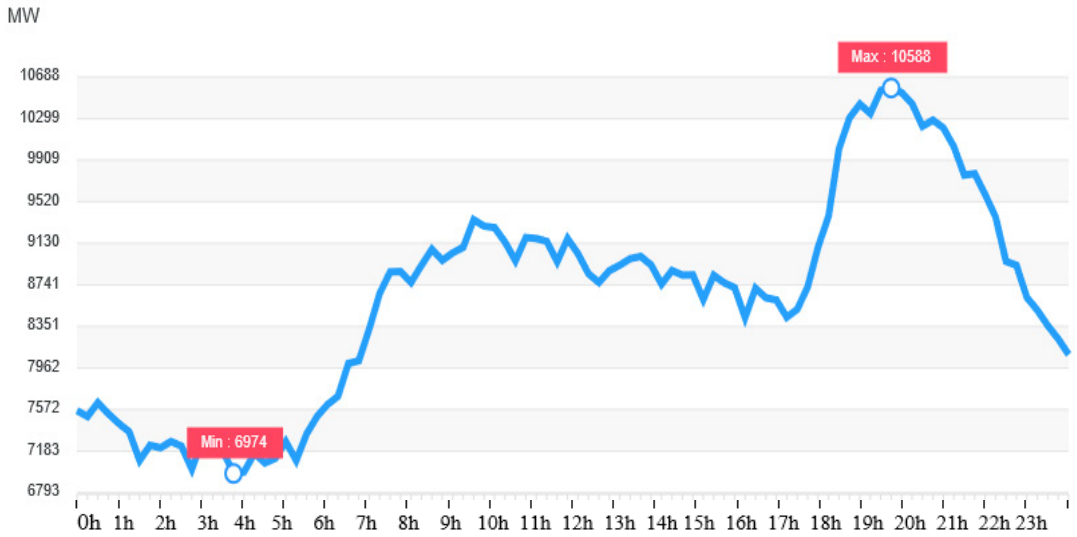
Management of the project

- Team, leader, individual competences, divide the tasks
- Cooperative work
- Quality control, QoS
- Divide the project into work packages WP, schedule them with a Gantt chart

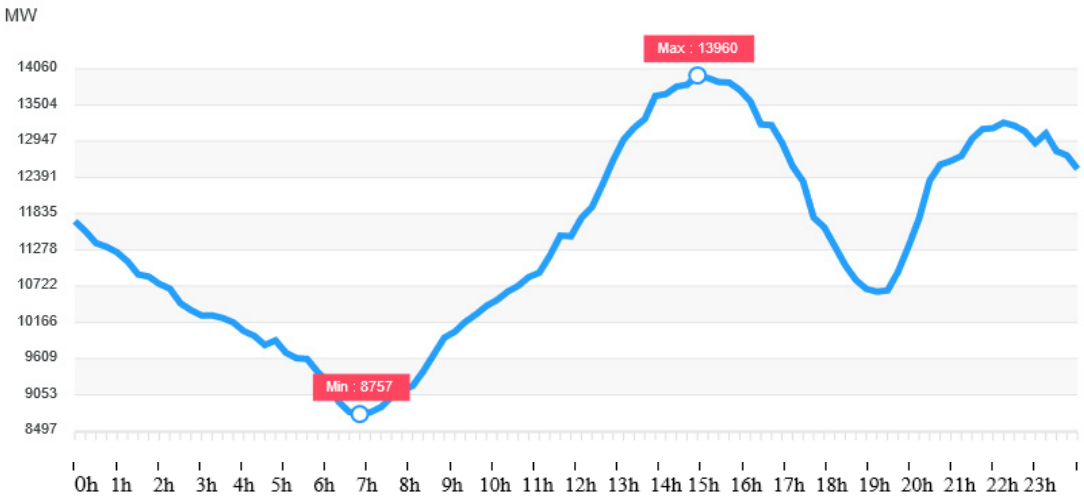
13.3. Case study 1: smart energy meter

Need to rethink the energy metering
Load curves depends on daily factors and season/weather factors
Load curves are for Algeria power grid operator
In summer, peek demand in the afternoon 15h, in the winter, the peak is at 20h
Force the user to shift the time slots of power consumption
Smart energy meter

- Sense the power and compute energy consumption per every 15 min slot
- Get daily updated prices from the grid operator DB
- Report the consumption and price to user and to grid operator.
- Store in the DB
- Monitor the past consumption and forecast the future use or trend



Load curve during a week day in January



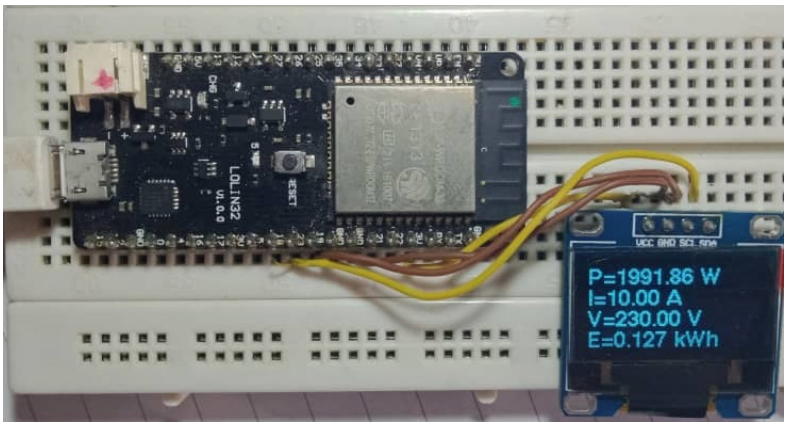
Load curve during a week day in August

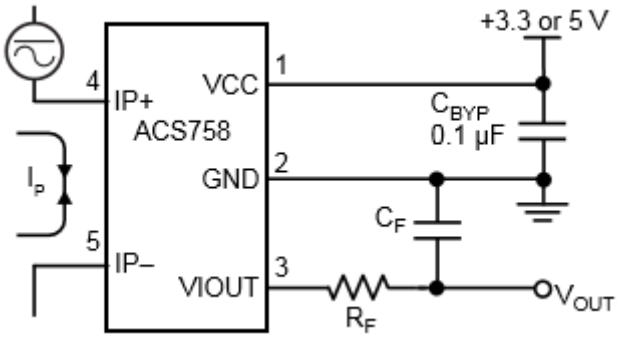
Hardware

- Hall effect current and voltage IC sensors
- Core-less transformers, filters
- Instantaneous measurement, ADC of microcontroller
- Wi-Fi or 4G connection to internet / MySQL database
- Oled display for end user
- Control other appliances: water heater tank, dish washer, electric car charger

Software

- Server-side PHP, JavaScript, HTML, MySQL code
- Embedded C code on microcontroller
- Java on Android App. for end user (MHI) Machine to Human Interface





Current sensor ACS758

13.4. Case study 2: DC small connected micro grids

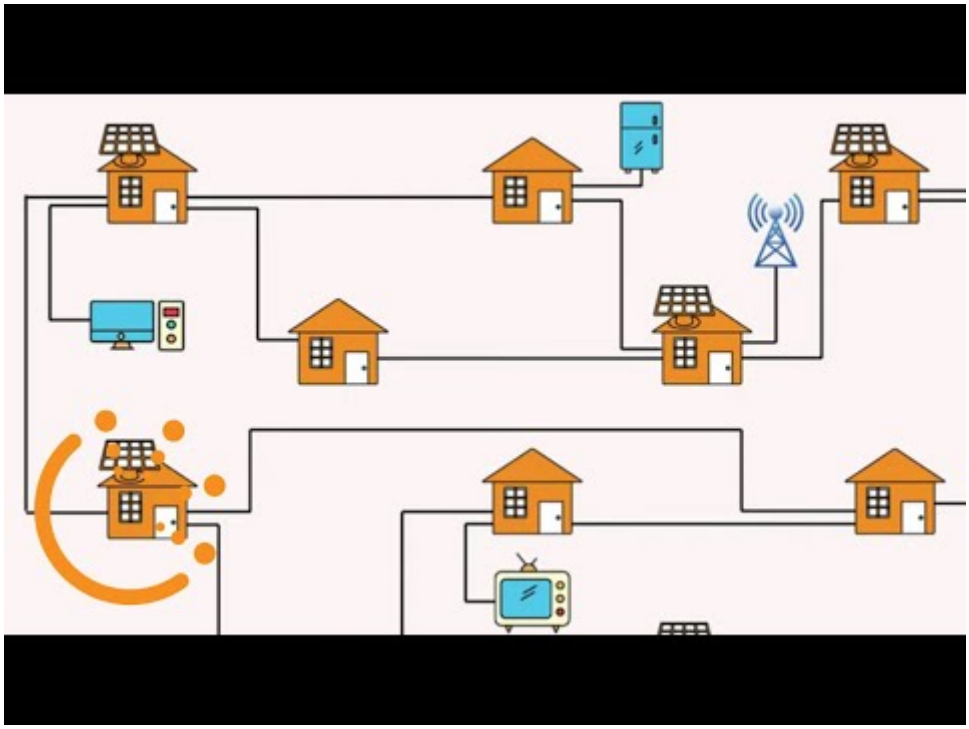
DC small connected micro grids with low power transfer and energy / money metering

The idea is not new but is very interesting for small community in villages

Produce low amounts of power, on a DC grid distributed community
PV panels, batteries, energy sharing boxes
Compute the net energy transfer from/to neighbors
Charge the account by paying real money to ask neighbors that produces the energy to send some of it
Automatically ruled by the box

No connection to the grid operator, no policy

https://me-solshare.com
Remote-monitoring solutions https://mekarana.jimdofree.com/



13.5. Summary

Present 3 case studies
Brainstorming, divide the tasks, study in group
Key points and technical aspects that must be addressed in an IoT project
Send all the assessment work in pdf files by uploading them on the platform
https://lms.pauwes.dz/course/view.php?id=116