

# Tokamak GOLEM

## Data processing

---

P. Macha, V. Svoboda

# OUTLINE

- Motivation
- Data format and access
- WGET
- EXCEL
- GNUPLOT
- MATLAB
- OCTAVE
- PYTHON

# Why do we need data processing?

- Data processing is the most important part of every experiment.
- Discharge produces various raw data.
- The data needs to be processed.
- Several processing tools can be used:
  - Programming languages - Python, MATLAB, GNU PLOT, Octave, etc.
  - Spreadsheets - Excel (not recommended).
- Main output - figures, tables, etc.

# What is data processing?

1. Data collection - gather data from a source.
2. Sorting data - sort the raw data (remove incorrect data, remove offset, ...)
3. Data input - prepare the data for processing
4. Data processing - processing the data input (integrate, differentiate, sum, ...)
5. Data output - creating visualisation, figures, ...
6. Data storage - save the data (for the next use)

# DATA ACCESS

- All the discharge outputs are available the GOLEM website

***[http://golem.fjfi.cvut.cz/shots/<#shot\\_no>/](http://golem.fjfi.cvut.cz/shots/<#shot_no>)***

- The most recent discharge

***<http://golem.fjfi.cvut.cz/shots/0>***

- Data <CHANNEL> from DAS <DAS\_name>

***[http://golem.fjfi.cvut.cz/shots/<#shot\\_no>/<DAS\\_name>/<CHANNEL>.csv](http://golem.fjfi.cvut.cz/shots/<#shot_no>/<DAS_name>/<CHANNEL>.csv)***

# SHOT HOMEPAGE

1. Discharge number
2. Discharge command
3. Technological parameters
4. Plasma parameters
5. Diagnostics

## Tokamak GOLEM - Shot Database #40151

Basic Diagnostics
🔍 📄 🏠

**The date of discharge execution** 22.11.2020 20:32:50 [Shot logbook]

**The session mission** Golem -> Training course for NUCE.psu. Additional session for Camila et al. Pressure scan.

**The session ID** 40132

**The discharge comment** Pressure scan - 28 mPa

**Discharge command**  
`./Dirigent.sh --discharge --UBI 800 --Tbt 0 --Ucd 500 --Tcd 1000 --preionization 1 --gas H --pressure 28 --ScanDefinition "40146 40147 40148 40149 40150" --comment "Pressure scan - 28 mPa"`

**Technological parameters**

- Working Gas:  $p_{\text{chamber}}^{\text{discharge, before}} = 0,85 \text{ mPa}$ ;  $p_{\text{chamber}}^{\text{discharge, pre}} = 0,85 \text{ mPa}$  ( $p_{\text{WG}}^{\text{request}} = 28 \text{ mPa}$  @  $X_{\text{WG}}^{\text{request}} = \text{H}$ )
- Toroidal magnetic field:  $U_{B_i}^{\text{request}} = 800 \text{ V}$  @  $t_{B_i}^{\text{request}} = 0,0 \text{ us}$
- Current drive field:  $U_{E_{cd}}^{\text{request}} = 500 \text{ V}$  @  $t_{E_{cd}}^{\text{request}} = 1000,0 \text{ us}$

**Plasma:** 📄 📄 📄

- Plasma: yes or no:
- Time parameters:  $\Delta t_p = 11,20 \text{ ms}$  (from:  $t_{\text{start}} = 3,33 \text{ ms}$ , to:  $t_{\text{end}} = 14,53 \text{ ms}$ )

**Plasma parameters:** 📄 📄 📄

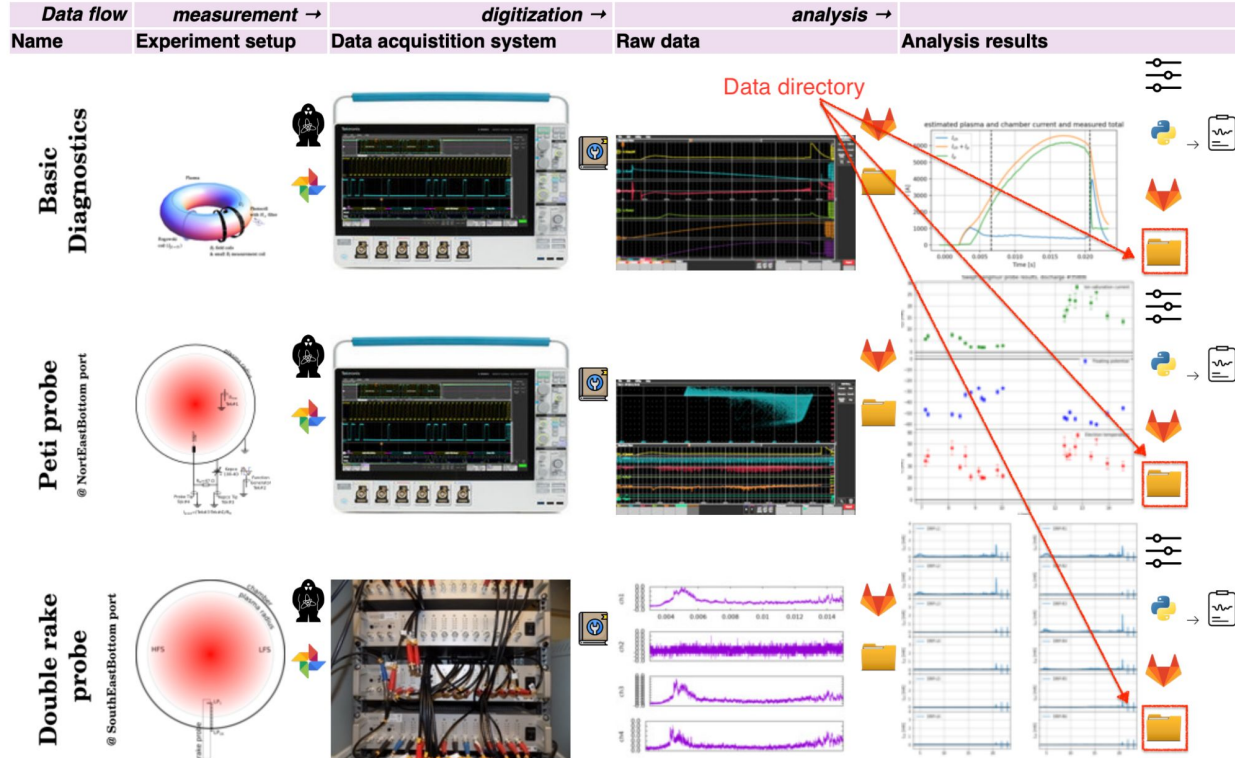
- Loop voltage:  $\overline{U_{\text{loop}}} = 7,26 \text{ V}$ ;  $\max_{r \in [\text{discharge}]} U_{\text{loop}} = 35,31 \text{ V}$ ;  $U_{\text{breakdown}} = 11,92 \text{ V}$
- Toroidal magnetic field:  $\overline{B_i} = 0,26 \text{ T}$ ;  $\max_{r \in [\text{discharge}]} B_i = 0,38 \text{ T}$
- Plasma current:  $I_p = 3,73 \text{ kA}$ ;  $\max_{r \in [\text{discharge}]} I_p = 4,64 \text{ kA}$ ;  $t_{I_p}^{\text{max}} = 13,13 \text{ ms}$

**On stage diagnostics**

|                                                           | Data flow → measurement → | digitization →          | analysis →                     |
|-----------------------------------------------------------|---------------------------|-------------------------|--------------------------------|
| Name                                                      | Experiment setup          | Data acquisition system | Raw data      Analysis results |
| <b>Basic Diagnostics</b><br>                              |                           |                         |                                |
| <b>Interferometry</b><br><small>@ SouthWest ports</small> |                           |                         |                                |














# Where to find data on homepage?

## On stage diagnostics



# Basic Diagnostics data directory

## Index of /shots/40745/Diagnostics/BasicDiagnostics/Basic/Results

| <a href="#">Name</a>                                                                                               | <a href="#">Last modified</a> | <a href="#">Size</a> | <a href="#">Description</a> |
|--------------------------------------------------------------------------------------------------------------------|-------------------------------|----------------------|-----------------------------|
|  <a href="#">Parent Directory</a> |                               | -                    |                             |
|  <a href="#">Bt.csv</a>           | 2023-02-14 19:20              | 792K                 |                             |
|  <a href="#">Bt_max</a>           | 2023-02-14 19:20              | 5                    |                             |
|  <a href="#">Bt_mean</a>          | 2023-02-14 19:20              | 5                    |                             |
|  <a href="#">Ich.csv</a>          | 2023-02-14 19:20              | 735K                 |                             |
|  <a href="#">Ich_max</a>          | 2023-02-14 19:20              | 5                    |                             |
|  <a href="#">Ich_mean</a>         | 2023-02-14 19:20              | 5                    |                             |
|  <a href="#">Ip.csv</a>           | 2023-02-14 19:20              | 721K                 |                             |
|  <a href="#">Ip_max</a>           | 2023-02-14 19:20              | 6                    |                             |
|  <a href="#">Ip_mean</a>          | 2023-02-14 19:20              | 6                    |                             |
|  <a href="#">U_loop.csv</a>       | 2023-02-14 19:20              | 562K                 |                             |
|  <a href="#">U_loop_max</a>       | 2023-02-14 19:20              | 6                    |                             |
|  <a href="#">U_loop_mean</a>      | 2023-02-14 19:20              | 5                    |                             |

Right click selected .csv file to find URL.

URL will be used later for data downloading.



# DATA FORMAT

- Data stored in **csv** format (columns separated by "," symbol).
- Several ways to download data:
  - **Manually** download by clicking on it (not recommended, slow, not automatized).
  - Using **wget** command (Linux, Mac users, Windows WSL).
  - Automated downloading using programming languages (**Python**, MATLAB, ...).
- DAS temporal resolution 1  $\mu$ s.
- DAS produces **N** columns x **40000** rows.
- 1st column represents time.
- 2nd column represents measured data.

# GOLEM - basic Data Acquisition System DAS

- An example of data representation.
- Csv files can vary in number of columns.
- Time axis units can vary (s or ms).
- Separators can vary.
- Separator must be specified to load the data.
- **Check data before analyzing!**

| time [ms]          | Bt [T]              |
|--------------------|---------------------|
| 9.99335744         | 0.2602015123779187  |
| 9.99435744         | 0.26022505899274373 |
| 9.99535744         | 0.260248611329229   |
| 9.99635744         | 0.26027216366571426 |
| 9.99735744         | 0.2602957195781975  |
| 9.998357440000001  | 0.2603192726298683  |
| 9.99935744         | 0.2603428235359473  |
| 10.0003574         | 0.2603663749303391  |
| 10.0013574         | 0.26038992583641807 |
| 10.0023574         | 0.26041349033133887 |
| 10.003357399999999 | 0.26043706197829086 |
| 10.0043574         | 0.2604606236123992  |
| 10.0053574         | 0.2604841802400679  |
| 10.0063574         | 0.26050773686773676 |
| 10.0073574         | 0.2605312999322512  |
| 10.0083574         | 0.26055487014879697 |
| 10.0093574         | 0.26057844251093454 |
| 10.0103574         | 0.26060201415788653 |
| 10.0113574         | 0.2606255822288406  |

# WGET

- Wget is used on Linux, Mac, or Windows WSL for data downloading.

*Wget is the non-interactive network downloader used to download files from the server.*

- For cycles can be used in **bash** to download data from multiple discharges:

```
#!/bin/bash
```

```
for shot in {40740..40745}
```

```
do wget "http://golem.fjfi.cvut.cz/shots/$shot/Diagnostics/BasicDiagnostics/Basic/Results/U_loop.csv"
```

```
done
```

- The script downloads Bt data from discharges 40740 to 40745.

# EXCEL

- Download data using wget and open one after another.
- Open data directly from URL:

[How to Import Web Data Into Excel on PC or Mac \(with Pictures\)](#)

- **Excel is tolerated but not recommended.**
- Usable for easy tasks, not suitable for more complex tasks.
- Disadvantages: difficult to load data, handle it, and process
- Advantages: no need of programming

# GNUPLOT

- Command-line driven utility generating two and three dimensional plots.
- Can be used for automatized data visualization.
- Useful and versatile.
- It is not programming language.
- Not very intuitive.
- No graphical interface available.
- The data processing must be done outside GNUPLOT.

# GNU PLOT - code example

```
# Create a path to the data
identifier = "U_loop.csv"; # signal name
ShotNo = "40745" # discharge number
DAS="Diagnostics/BasicDiagnostics/Basic/Results/" # diagnostics path
baseURL="http://golem.fjfi.cvut.cz/shots/" # base URL
DataURL= baseURL.ShotNo."/".DAS.identifier # connect together

set datafile separator ","; # define separator

# figure setup
set title "Uloop for #".ShotNo; # title
set xrange [0:25];set xlabel "Time [ms]";set ylabel "U_l [V]" # labels

# make the plot
plot '< wget -q -O - DataURL' u 1:2 w l t 'Uloop';
```

# MATLAB

- Easy to use.
- Great for matrix data manipulation and visualization.
- Predefined functions and complex graphical interface.
- Interpreted language - can be slow.
- Is not free software.
- Syntax not very similar to other programming languages.

# MATLAB - code example

```
% Create a path to the data
identifier = 'U_loop.csv';
ShotNo = '40745';
DAS = 'Diagnostics/BasicDiagnostics/Basic/Results/';
baseURL = 'http://golem.fjfi.cvut.cz/shots/';
DataURL = strcat(baseURL, ShotNo, '/', DAS, identifier);

% Write data from GOLEM server to a local file
websave('data', DataURL); % download the data from URL to local file called data.csv

% Load data
data = load('data.csv', 's,'); % load data from the local data.csv file

% Plot and save the graph
figure;
plot(data(:,1), data(:,2), '.');
title(strcat('U loop for #', ShotNo));
xlabel('time [ms]');
ylabel('U_loop [V]');
```



# OCTAVE

- Similar syntax to MATLAB.
- Free software.
- Both GUI and no-GUI interface.
- No all of the fancy libraries and packages MATLAB offers.
- Not a general purpose programming language.
- Interpreted language => slow.

# OCTAVE - code example

```
% Create a path to the data
identifier = 'U_loop.csv';
ShotNo = '40745';
DAS = 'Diagnostics/BasicDiagnostics/Basic/Results/';
baseURL = 'http://golem.fjfi.cvut.cz/shots/';
DataURL = strcat(baseURL, ShotNo, '/', DAS, identifier);

% Write data from GOLEM server to a local file
urlwrite(DataURL, 'data_octave'); % download the data from URL to local file called data

% Load data
data = load('data_octave', 's,'); % load data from the local data.csv file

% Plot and save the graph
figure;
plot(data(:,1), data(:,2), '.');
title(strcat('U loop for #', ShotNo));
xlabel('time [ms]');
ylabel('U_loop [V]');
```

# PYTHON - recommended programming language

- One of the best programming languages to be used generally.
- Offers simple syntax, powerful modules.
- Relatively easy to learn, some parts of syntax similar to MATLAB (matplotlib).
- Free to use.
- Can be very fast when using correctly.
- Jupyter Notebook - great IDE allowing divide the code into cells.
- Almost no disadvantages :)

# PYTHON - IDEs to use

- Script in .py file - *python3 script.py* - not optimal.
- pyCharm
- SPYDER
- **Jupyter Notebook - Anaconda3**
  - **Advantages:** all in one place, easy to convert (HTML, PDF, ...), easy to share, interactive, code divided into cells, markdown included, easy to use custom python environment, great for simpler data processing tasks.
  - **Disadvantages:** not best for long projects, less security, no code asistation.

<https://docs.anaconda.com/anaconda/install/index.html>

## Comments

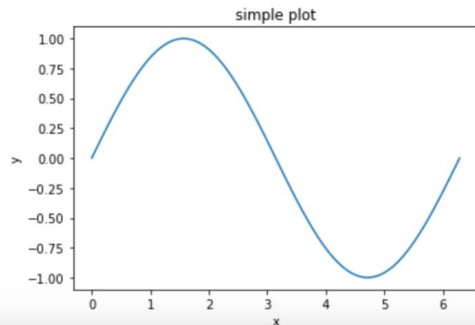
- comments can be written using markdown language
- the best way to comment your code
- equations can be written using LaTeX -  $R = \frac{U}{T}$

```
In [10]: 1 # Module imports
         2 import numpy as np
         3 import matplotlib.pyplot as plt
         4 import pandas as pd
```

```
In [11]: 1 x = np.linspace(0, 2 * np.pi, 100)
         2 y = np.sin(x)
```

```
In [12]: 1 plt.figure()
         2 plt.plot(x, y)
         3 plt.title('simple plot')
         4 plt.xlabel('x')
         5 plt.ylabel('y')
```

Out[12]: Text(0, 0.5, 'y')



## Basic operations

```
In [151]: 1 # anything after '#' is a comment
          2 a = 5 # assign value 5 (integer) to name 'a'
          3 b = a * 2.5 - 3.7 + 9.7 / 3 + 2**4 # automatic casting to float, standard math operators (** is exponentiat
          4 b # last (unassigned) value in cell is returned and printed
```

Out[151]: 28.03333333333333

```
In [153]: 1 description = 'b is now:' # strings are delimited by ' ' or ""
          2 print(description, b) # the print function (called with 'func()') prints all arguments
```

b is now: 28.03333333333333

```
In [154]: 1 def radius(x, y, x0=0, y0=0): # 'def' starts a function definition, arguments can have default values arg=valu
          2     """Return radius of point [x,y] from [x0,y0]"""
          3
          4     r = (x-x0)**2 + (y-y0)**2 # commands within the function are indented by 4 spaces
          5     return r**0.5 # this is the value returned by the function
```

```
In [155]: 1 vec = [1, 5, 6, 7] # a list of values in square brackets
```

```
In [158]: 1 for v in vec: # iterate over elements in list
          2     r = radius(v, y=1) # parameters specified by position or name=value
          3     print('r = ', round(r, 2)) # print value rounded to 2 decimal points
```

r = 1.41  
r = 5.1  
r = 6.08  
r = 7.07

```
In [157]: 1 rs = [radius(v, 5) for v in range(10)] # list generator comprehension using range(n) -> [0, 1, ..., n-1]
          2 rs
```

Out[157]: [5.0,  
5.0990195135927845,  
5.385164807134504,  
5.830951894845301,  
6.4031242374328485,  
7.0710678118654755,  
7.810249675906654,  
8.602325267042627,  
9.433981132056603,  
10.295630140987]

# PYTHON - collections

- List:
  - Defined by square bracelets.
  - A collection of ordered data, mutable.
- Tuple:
  - Defined by round bracelets.
  - Ordered collection of data, immutable, higher performance, used as indices for dicts, have defined shape.
- Dictionary:
  - Defined by curly bracelets.
  - Unordered collection of data that stores data in key-value pair.

In [171]:

```
1 # array
2 my_array = [1, 2, 3, 4, 5]
3 print(my_array)
4 print('value of my_array on third position: ', my_array[2])
```

```
[1, 2, 3, 4, 5]
value of my_array on third position: 3
```

In [172]:

```
1 # tuple
2 my_tuple = (1, 2, 3, 4, 5)
3 print(my_tuple)
4 print('value of my_tuple on second position: ', my_tuple[1])
```

```
(1, 2, 3, 4, 5)
value of my_tuple on second position: 2
```

In [173]:

```
1 # dictionary
2 my_dictionary = {'A': [1, 2, 3], 'B': 'test', 'C': 5.234}
3 print(my_dictionary)
4 print('value of my dictionary in key A: ', my_dictionary['A'])
```

```
{'A': [1, 2, 3], 'B': 'test', 'C': 5.234}
value of my dictionary in key A: [1, 2, 3]
```



# PYTHON - working with strings

- An 'f' strings can be used for inserting variables into strings.
- Insert 'f' letter at the beginning of the string.
- Use curly braces '{', '}' and insert the variable inside.
- The string will be concatenated.

```
In [76]: 1 shot_no = '40745'  
2 identifier = 'U_loop.csv'  
3 url = f'http://golem.fjfi.cvut.cz/shots/{shot_no}/Diagnostics/BasicDiagnostics/Basic/Results/{identifier}'  
4  
5 print(url)
```

[http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/Results/U\\_loop.csv](http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/Results/U_loop.csv)

# PYTHON - modules

- Numpy
  - Multi-dimensional arrays and matrices equipped with high-level mathematical operations.
  - In our case is used for basic data manipulation.
- Matplotlib
  - Plotting library used for creating static, animated, or interactive visualizations.
  - In our case is used for basic or complex data plotting.
- Pandas
  - Module used for data manipulation and analysis in form of data structured (various tables).
  - In our case is used for data manipulation in form of data frames.

# PYTHON - numpy

```
In [10]: 1 import numpy
```

## Important numpy functions

array, linspace, arange, empty, eye, zeros, ones, load, loadtxt, save, random.rand

```
In [15]: 1 A = [1, 2, 3, 4, 5] # standard Python array
2 A_np = np.array([1, 2, 3, 4, 5])
3
4 print('Standart Python list type: ', type(A))
5 print('Numpy Python list type: ', type(A_np))
```

```
Standart Python list type: <class 'list'>
Numpy Python list type: <class 'numpy.ndarray'>
```

```
In [17]: 1 matrix = np.random.rand(10, 10) # create matrix 10x10 filled by random numbers
```

```
In [19]: 1 print(matrix[:, 0]) # indexing the first column of the matrix
```

```
[0.70609552 0.76054409 0.46304194 0.95361304 0.16183893 0.02872066
 0.4457452  0.35962072 0.52477787 0.70726263]
```

```
In [26]: 1 # Basic operations
2 B = matrix + 2 # add 2 to each element of matrix
3 C = matrix * B # multiply each element of matrix by element of B
```

# PYTHON - numpy

## Loading data from server using numpy

```
In [33]: 1 # Import urlopen from urllib.request library
          2 # The library is used for opening data on servers
          3 from urllib.request import urlopen
```

```
In [35]: 1 # Define path to the data
          2 url = 'http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/Results/U_loop.csv'
          3
          4 # Use urlopen to open the URL, the load opened URL with np.loadtxt
          5 # Delimiter must be defined!
          6 U_loop = np.loadtxt(urlopen(url), delimiter = ',')
```

```
In [38]: 1 # Shape of the data can be displayed using shape function
          2 # First column is time axis, second the loop voltage
          3 print('Data shape: ', U_loop.shape)
```

Data shape: (24000, 2)

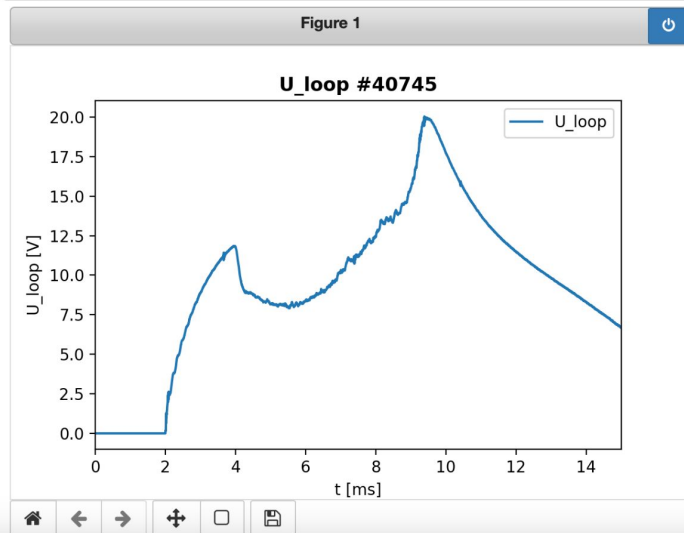
```
In [40]: 1 # Optional step (not recommended to do this), better to use in one variable U_loop
          2 t = U_loop[:, 0] # this is how to load time axis into variable
          3 U_loop_data = U_loop[:, 1] # this is how to load data into variable
```

# Python - matplotlib

## Data visualization

```
In [49]: 1 # Import matplotlib library for data visualization
2 import matplotlib.pyplot as plt
3 # Highly recommended to use notebook format - allows interactive plots
4 %matplotlib notebook
```

```
In [58]: 1 plt.figure(figsize = (6, 4)) # create a figure first
2 plt.plot(U_loop[:, 0], U_loop[:, 1], label = 'U_loop') # plot the data (time and data columns), create label
3 plt.title('U_loop #40745', fontweight = 'bold') # create title
4 plt.xlabel('t [ms]') # create x label
5 plt.ylabel('U_loop [V]') # create y label
6 plt.xlim(0, 15) # adjust x-axis range
7 plt.legend()
```



# Python - pandas

## Using pandas

```
In [61]: 1 # Import pandas module  
2 import pandas as pd
```

```
In [64]: 1 url = 'http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/Results/U_loop.csv'  
2 # Easy way to load the data into pandas DataFrame  
3 U_loop_pd = pd.read_csv(url, names = ['t', 'data'])
```

```
In [66]: 1 U_loop_pd.head(10)
```

Out[66]:

|   | t         | data |
|---|-----------|------|
| 0 | -0.719643 | 0.0  |
| 1 | -0.718643 | 0.0  |
| 2 | -0.717643 | 0.0  |
| 3 | -0.716643 | 0.0  |
| 4 | -0.715643 | 0.0  |
| 5 | -0.714643 | 0.0  |
| 6 | -0.713643 | 0.0  |
| 7 | -0.712643 | 0.0  |
| 8 | -0.711643 | 0.0  |
| 9 | -0.710643 | 0.0  |

# PYTHON - pandas

```
In [67]: 1 url = 'http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/Results/U_loop.csv'  
2 # Easy way to load the data into pandas DataFrame  
3 U_loop_pd = pd.read_csv(url, names = ['t', 'data'], index_col = 't')
```

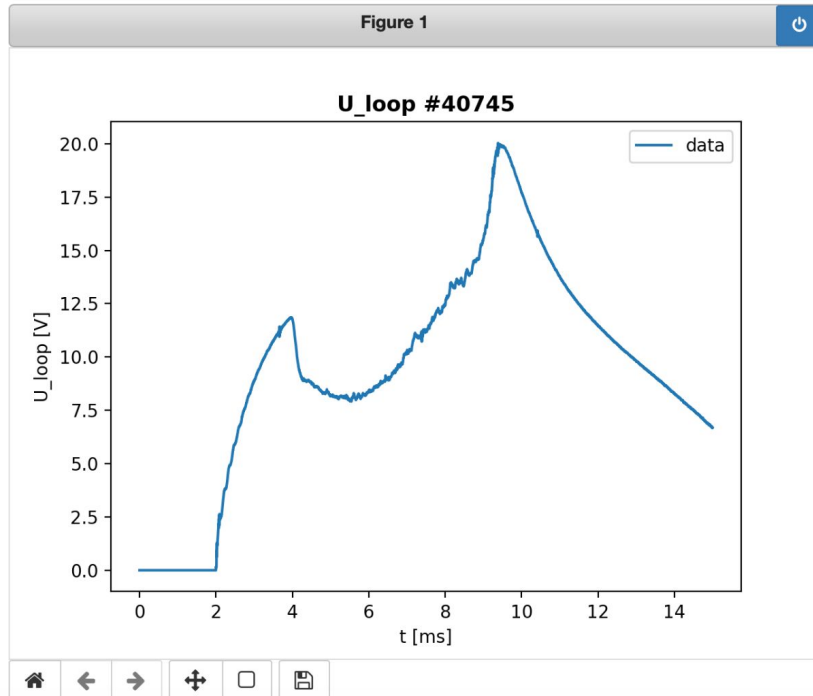
```
In [68]: 1 U_loop_pd.head(10)
```

Out[68]:

|           | data |
|-----------|------|
| t         |      |
| -0.719643 | 0.0  |
| -0.718643 | 0.0  |
| -0.717643 | 0.0  |
| -0.716643 | 0.0  |
| -0.715643 | 0.0  |
| -0.714643 | 0.0  |
| -0.713643 | 0.0  |
| -0.712643 | 0.0  |
| -0.711643 | 0.0  |
| -0.710643 | 0.0  |

# PYTHON - pandas

```
In [75]: 1 U_loop_pd[0: 15].plot() # the way how to plot in pandas, because time is index, [] can be used for slicing data
2 plt.title('U_loop #40745', fontweight = 'bold') # create title
3 plt.xlabel('t [ms]') # create x label
4 plt.ylabel('U_loop [V]') # create y label
5 plt.legend()
```

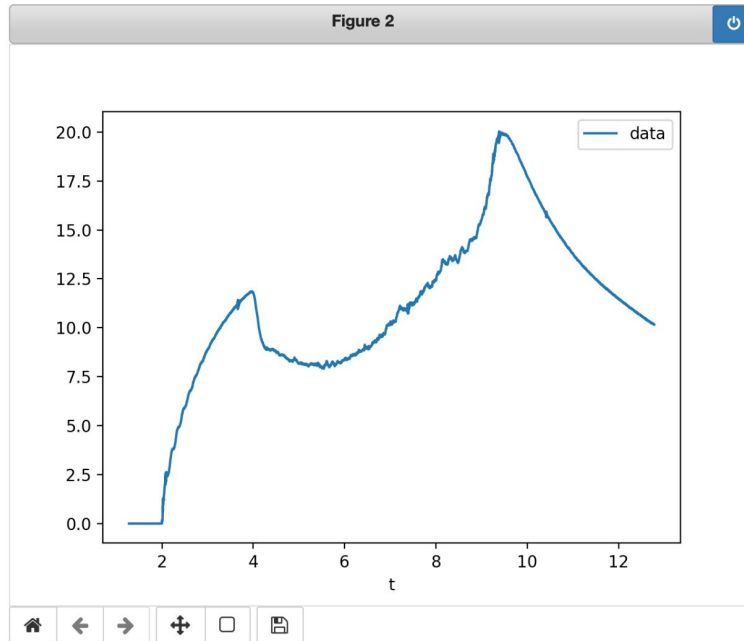




# PYTHON - selecting data in pandas

```
In [80]: 1 # How to select column from a DataFrame (not recommended):  
2 t = U_loop_pd.index  
3 U_loop_pd_data = U_loop_pd.data
```

```
In [85]: 1 # Indexing by index (from index 2000 to index 13500)  
2 U_loop_pd.iloc[2000: 13500].plot()
```

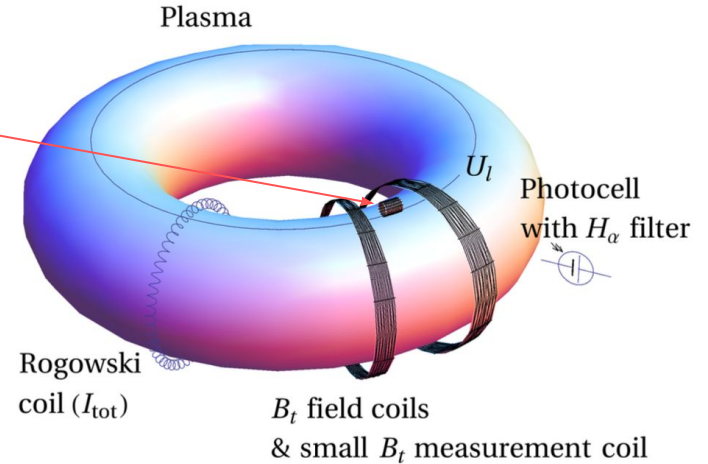


# Simple example of toroidal magnetic field determination

# Toroidal magnetic field determination

- Toroidal magnetic field measured by small coil.
- Its axis parallel to the magnetic field.
- Measures the time derivative of the toroidal magnetic field.
- Toroidal magnetic field determined based on the Faraday's law.

$$B_t(t) = C_{B_t} \int_0^t U_{B_t}(\tau) d\tau$$



# Loading the data

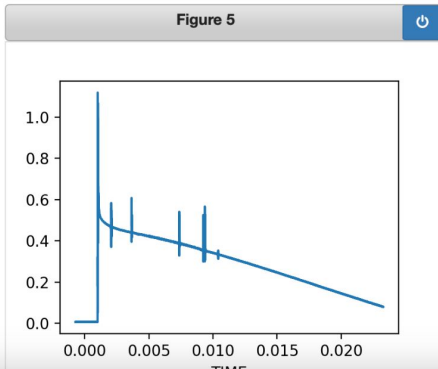
```
In [99]: 1 url = 'http://golem.fjfi.cvut.cz/shots/40745/Diagnostics/BasicDiagnostics/Basic/DAS_raw_data_dir/TektrMS056_ALL.  
2 # Load all basic diagnostics signal into pandas DataFrame  
3 data_basic_diag = pd.read_csv(url, skiprows = 10, index_col = 'TIME')
```

```
In [101]: 1 # Show loaded data. Measured signals corresponds to CH1, .. CH6  
2 data_basic_diag.head(5)
```

Out[101]:

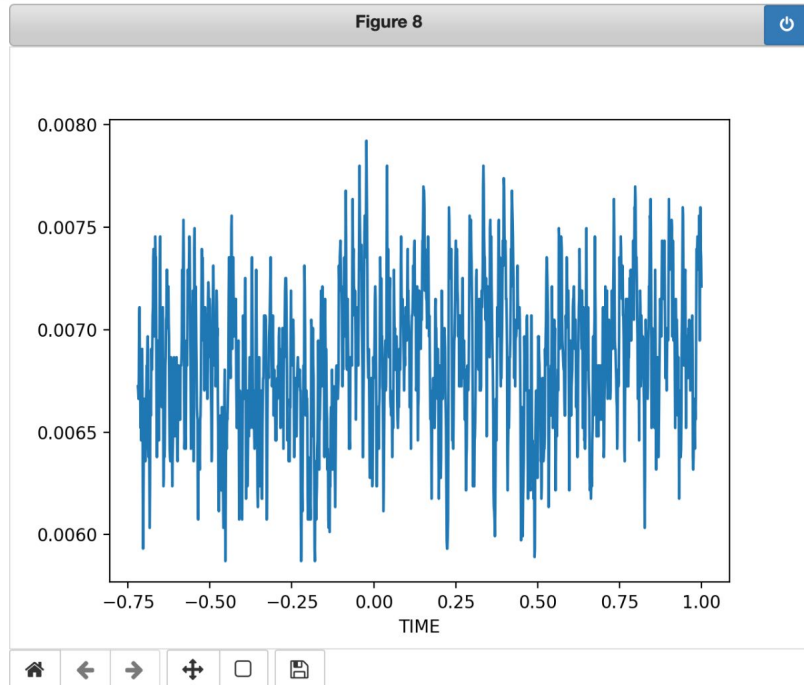
|           | CH1      | CH2      | CH3       | CH4      | CH5      | CH6      | Unnamed: 7 | TIME.1    | MATH1        | Unnamed: 10 | TIME.2    | MATH2      |
|-----------|----------|----------|-----------|----------|----------|----------|------------|-----------|--------------|-------------|-----------|------------|
| TIME      |          |          |           |          |          |          |            |           |              |             |           |            |
| -0.000720 | 0.138281 | 0.006723 | -0.001734 | 0.008125 | 0.043906 | 0.046289 | NaN        | -0.000720 | 4.734645e-07 | NaN         | -0.000720 | -14.246607 |
| -0.000719 | 0.137500 | 0.006703 | -0.000703 | 0.008000 | 0.043906 | 0.043828 | NaN        | -0.000719 | 9.462137e-07 | NaN         | -0.000719 | -14.159606 |
| -0.000718 | 0.128125 | 0.006663 | 0.000484  | 0.008000 | 0.044687 | 0.042656 | NaN        | -0.000718 | 1.416817e-06 | NaN         | -0.000718 | -13.192532 |
| -0.000717 | 0.119531 | 0.006764 | -0.000266 | 0.007938 | 0.045000 | 0.040664 | NaN        | -0.000717 | 1.889567e-06 | NaN         | -0.000717 | -12.307158 |
| -0.000716 | 0.130469 | 0.007048 | -0.002750 | 0.008000 | 0.046875 | 0.039023 | NaN        | -0.000716 | 2.375905e-06 | NaN         | -0.000716 | -13.426744 |

```
In [115]: 1 plt.figure(figsize = (4, 3))  
2 data_basic_diag['CH2'].plot() # plot signal measured by Bt coil
```



# Removing offset I

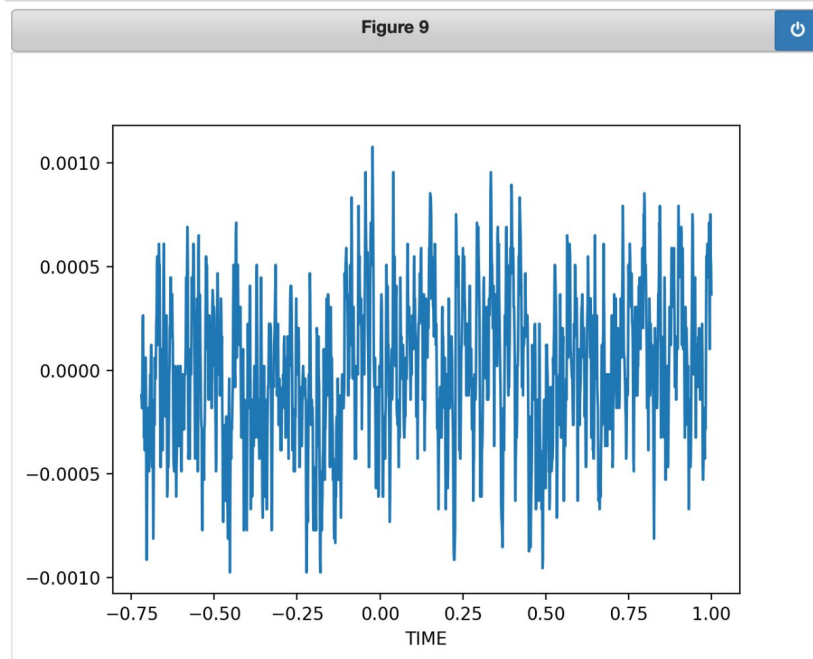
```
In [131]: 1 # Signal is not starting at 0! Must be shifter to zero to calculate integral correctly!  
2 plt.figure()  
3 data_basic_diag.loc[-1:1]['CH2'].plot()
```



# Removing offset II

```
In [132]: 1 data_basic_diag['CH2'] = data_basic_diag['CH2'] - data_basic_diag.loc[-1:1]['CH2'].mean()
```

```
In [133]: 1 # Signal is not starting at 0! Must be shifter to zero to calculate integral corectly!  
2 plt.figure()  
3 data_basic_diag.loc[-1:1]['CH2'].plot()
```

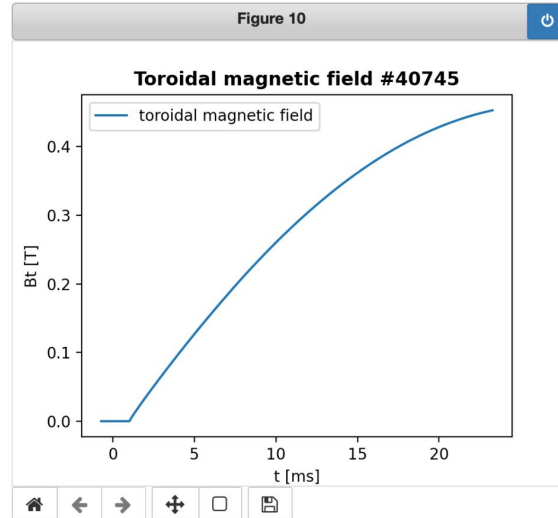


# Integration

```
In [134]: 1 # Import module for numerical integration  
2 from scipy import integrate
```

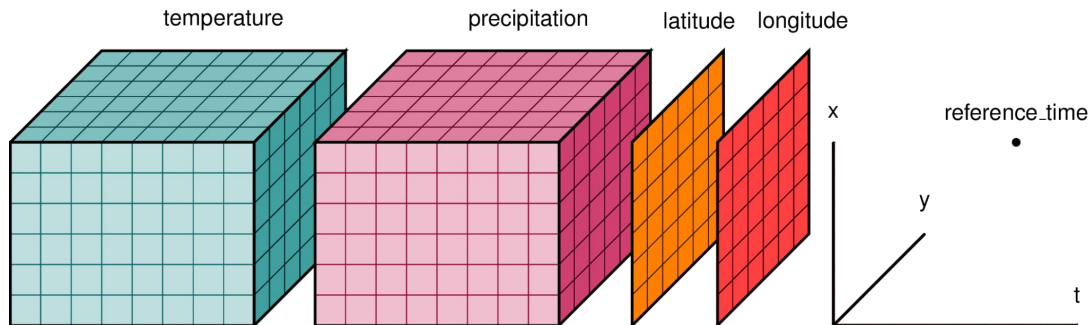
```
In [146]: 1 Bt = pd.Series(integrate.cumtrapz(data_basic_diag['CH2'], x=data_basic_diag.index/1000, initial=0) * 70.42,  
2 name = 'Bt', index = data_basic_diag.index)
```

```
In [150]: 1 plt.figure(figsize = (5, 4))  
2 Bt.plot(label = 'toroidal magnetic field')  
3 plt.xlabel('t [ms]')  
4 plt.ylabel('Bt [T]')  
5 plt.title('Toroidal magnetic field #40745', fontweight = 'bold')  
6 plt.legend()
```



# PYTHON - xarray

- Multidimensional arrays - **DataArray** or **Dataset**.
- More intuitive, more concise, and less error-prone.
- Introduces labels in form of dimensions, coordinates, and attributes.
- Example: **dimensions** (x, y, t), **coordinates** (latitude (x), longitude (y)), **data\_var** (temperature(x, y, t), precipitation(x,y,t))











# PYTHON - xarray example

```
[16]: ds = xr.open_dataset('./tutorial-data/sst/NOAA_NCDC_ERSST_v3b_SST-1960.nc')
      ds
```

```
[16]: xarray.Dataset
```

► Dimensions: (lat: 89, lon: 180, time: 12)

▼ Coordinates:

|             |        |                |                                   |                                                                                                                                                                         |
|-------------|--------|----------------|-----------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>lat</b>  | (lat)  | float32        | -88.0 -86.0 -84.0 ... 86.0 88.0   |   |
| <b>lon</b>  | (lon)  | float32        | 0.0 2.0 4.0 ... 354.0 356.0 358.0 |   |
| <b>time</b> | (time) | datetime64[ns] | 1960-01-15 ... 1960-12-15         |   |

▼ Data variables:

|            |                  |         |     |                                                                                                                                                                         |
|------------|------------------|---------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>sst</b> | (time, lat, lon) | float32 | ... |   |
|------------|------------------|---------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

▼ Attributes:

Conventions : IRIDL

source : <https://iridl.ldeo.columbia.edu/SOURCES/.NOAA/.NCDC/.ERSST/.version3b/.sst/>

history : extracted and cleaned by Ryan Abernathey for Research Computing in Earth Science

# Best Practises

- Document your code - make comments.
- Use one statement per line (do not use `print("A"); print("B"); ...`).
- Code indentation (proper tabs, spaces, etc).
- Do not repeat yourself
- Use a Proper Naming Convention.
- Avoid magic numbers.

# CONCLUSION

- Conversion of the data to useful information.
- Almost all experiments raw data has to be analyzed.
- Data can be processed using several programming languages:
  - **Python**, MATLAB, Octave, GNU PLOT, or Excel.
- Python is currently the best programming language for data analysis.
- Follow best practices.
- The most important - enjoy the process of data analysis :)