JUPYTER NOTEBOOKS FOR COMPUTER-BASED LABORATORIES ON ELECTRICAL ENERGY SYSTEMS

Guðrún Margrét Jónsdóttir and Federico Milano

School of Electrical, Electronic and Communications Engineering
University College Dublin (IRELAND)
gudrun.jonsdottir@ucdconnect.ie, federico.milano@ucd.ie

Abstract

This paper discusses the implementation of Jupyter notebook-based laboratories for the module "Electrical Energy Systems" taught by the authors in the academic year 2018/2019 at University College Dublin. The paper shows that using Jupyter notebooks for teaching can easily lead to a continuous assessment and improve the students learning curve, without being a burden for the instructor and the teaching assistants. An example of the lab activities proposed in the module as well as the students' performance and feedback are discussed in the paper.

Keywords: Electrical machines, electrical energy systems, modelling, inverse problems, Jupyter notebook project, Python, computer-based laboratory.

1 INTRODUCTION

Jupyter notebook is an open-source project that started in 2014 based on IPython which has quickly become popular in the scientific and academic community [1]. A notebook is basically an interactive webpage, that can be easily and efficiently designed for testing live code, embedding narrative text and visualizing results. The term "computational narrative" has been coined for these virtual notebooks and summarizes well their features.

The authors have taught the module "Electrical Energy Systems" for three years now. In the first two years, the students were asked to do standard electrical machine-based labs. These labs posed several logistic problems due to the high number of students (about 160) and the low number of available machines (5 transformers and 5 rotating machines). More importantly, machine-based labs were not particularly suited to 2nd-year students, due to the intrinsic risks associated with high-voltage devices and the need for the students to have properly assimilated the concepts of "modelling" and "measuring". These are indeed subtle engineering concepts that only more mature students, e.g. 3rd or 4th year ones, can appreciate and put into practice in the labs.

In the academic year 2017/18, the authors set up Jupyter-based labs for a small (about 15 students) fourth-year module, namely "Power Systems Dynamics and Control", which is offered exclusively to Electrical Engineering students. The success of this experiment, duly documented in a paper presented in [2], motivated the implementation of the labs for this larger and heterogeneous (students are from mechanical, electrical and electronic engineering programs) module "Electrical Energy Systems."

This paper shows that compared to standard tutorials and mid-term examinations, Jupyter-based labs can improve the students' performance and learning experience. This is done by increasing the variety and number of exercises that they are asked to solve and by gently introducing them to the concept of "modelling" physical devices through equations. Each notebook is designed so that the students have to solve "inverse problems", i.e. finding through a trial-and-error technique the parameters of a given machine or a simple power system to satisfy a given design requirements. Additionally, the students can visualize relevant variables associated to each electrical device or system being studied through plots embedded into the Jupyter notebook sheet.

The contributions of the paper are as follows.

- A description of the key features of Jupyter notebooks that make them adequate for second-year labs on electrical energy systems.
- The feedback of the students on the use of Jupyter notebooks and a comparison of the performance of the undergraduate students that attended the module "Electrical Energy Systems" in the

academic years 2017/18 (when Jupyter notebooks were not implemented) and 2018/19 (with Jupyter notebooks).
- A complete example of a laboratory activity that illustrates modelling issues and an inverse engineering problem.

The paper is organized as follows. Section 2 briefly describes Jupyter notebooks, JupyterHub, which is the server daemon utilized to make Jupyter notebooks available to the students, and Google forms that were used to collect results. Section 3 outlines the contents of the module “Electrical Energy Systems” and the laboratory activities that have been carried out. In Sections 4 the students' feedback on the labs is presented. Finally, Section 5 draws relevant conclusions and suggests future work directions.

2 OUTLINES OF THE SOFTWARE TOOLS

The software tools considered in this section are three: Jupyter notebook, JupyterHub and Google forms. Jupyter notebook is an interactive, web-based interface based on the Python programming language that has become very popular for teaching and laboratory activities [1]. JupyterHub is the server daemon that allows using Jupyter notebooks through a web server interface. Finally, Google forms is a tool that allows collecting information from users via personalized surveys and quizzes. In the labs the students use the notebooks to complete the lab tasks through code and visualize the results. When they have completed the lab tasks they gather their results and submit them through Google forms.

In the remainder of this section, we first briefly describe each tool and then discuss their integration and the main concepts on which the laboratory activities have been designed.

2.1 Jupyter notebook

According to the main webpage [1], “*Jupyter notebook, is an open-source web application that allows you to create and share documents that contain live code, equations, visualization and narrative text.”*

The main idea behind this tool is that one can mix text and equations with live code which can be changed and executed interactively by the user. By default, such a code is Python, but there are many extensions that allow utilizing other popular languages, such as Java and Julia. The native support of Python allows using the Jupyter notebooks with Numpy (for array algebra) and Matplotlib (for displaying results through book-grade figures).

There exists a Jupyter extension that allows to utilize Matlab as the programming language. This solution was considered in the beginning because most students, especially those of the 2nd and 3rd stage, tend to be more familiar with Matlab than Python. However, Python was finally used for the labs of the module Electrical Energy Systems for two reasons:

- Stakeholders have explicitly indicated that they prefer hiring students with experience in Python, which has become a sort of industry standard for mathematical computing (in particular the Numpy library).
- The Matlab extension does not work particularly well with the corporative license provided by UCD. Needless to say, Python does not require any license and, apart from being easy to install does not have any license issue.

For completing the labs, the students are required to implement simple Python scripts. Examples of such scripts are given in the appendix (Section A.1).

2.2 JupyterHub

According to the Jupyter website, *JupyterHub brings the power of notebooks to groups of users. It gives users access to computational environments and resources without burdening the users with installation and maintenance tasks.* [3].

JupyterHub has been installed on a Linux Fedora 28 server maintained by the second author. This has been set up using Apache and SSL to improve security. Any user with an account on the server can access Jupyter notebooks using any web browser, such as Firefox or Google Chrome (see Figure 1).

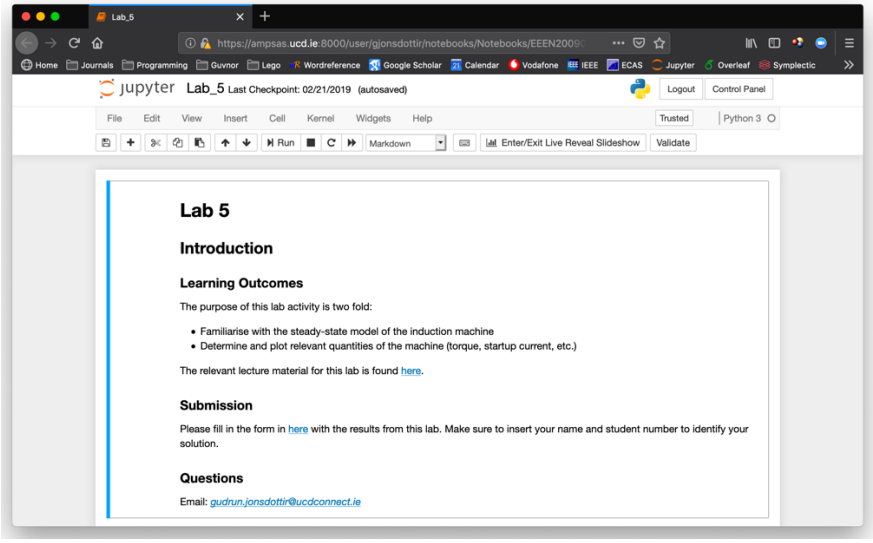

Figure 1. Web-browser-based JupyterHub interface to Jupyter notebooks.

In the first attempt to use JupyterHub for undergraduate modules, students' account were created manually [2]. This solution, however, appears impractical for a large class group such as the one of the module "Electrical Energy Systems," which, depending on the year, range from 150 to 180 students. Thus, the authors have implemented a set of scripts, written in Python, that allows creating automatically the students accounts from a comma-separated value (CSV) files. The first script creates that accounts using student unique IDs and assigning to each account a standard password that includes the student ID as well as a code. All accounts are also provided with the security level required by UCD policies.

It is important to note that individual accounts are also useful to avoid plagiarism. In fact, in other modules, where a single account is used by multiple students' groups, it has happened that some groups have improperly utilized code and material left on the hard disk by other groups. This behaviour is made impossible or, at least, highly discouraged, by the utilization of an individual account for each student.

A second script has been implemented to copy the Jupyter notebooks to the student accounts. Each notebook is uploaded to the students account right after the relevant topic has been covered in the lectures. In this way, the students have access to the notebooks on each of the course topics as the module progresses.

2.3 Google forms

Jupyter notebooks have the feature that both the structure of the lab and the answers and code implemented by the students can be saved in the same file. Such a file can then be exported as a PDF (or other common formats). This is certainly useful when the students are asked to discuss the results such as the activities described in [2]. However, when the number of student is large and the ultimate goal of the activity is to ask the students to solve problems that have a well-defined numerical solution, e.g. the current in a branch of a circuit, it is impractical for teaching assistants to have correct hundreds of PDF files with the whole saved Jupyter notebook. All that is really needed is the synthetic answer to each question.

There exists Jupyter extensions that are aimed at simplifying the collection of answers and the grading process, e.g., nbgrader [4]. However, these extensions do not fit well the purpose of these laboratories. Thus, Google forms were utilized (see Figure 2). These provide the right compromise between simplicity and flexibility and come with a plethora of tools to generate statistics of the answers provided by the students [5].

Section A.2 of the appendix provides an example of the Google form associated with a Jupyter notebook.

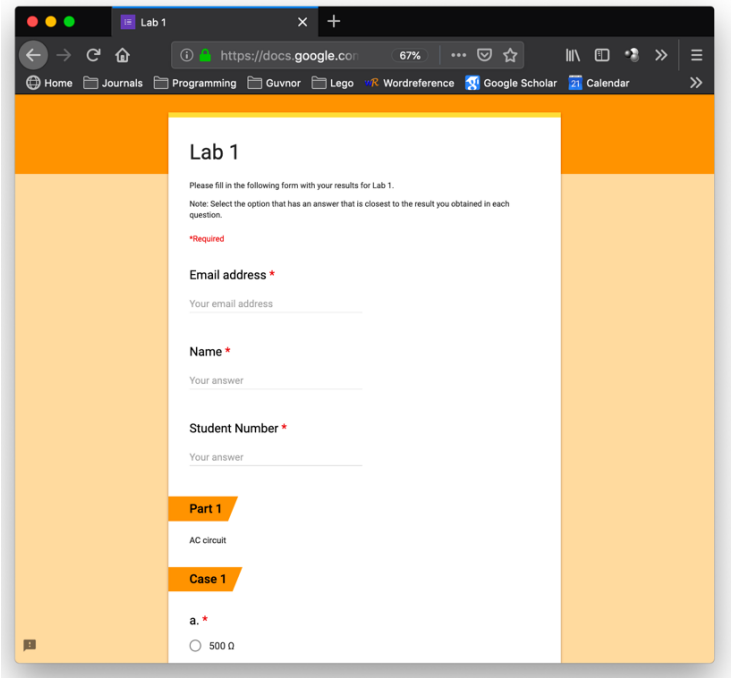

Figure 2. An example of aGoogle form utilized to collect student's answers.

3 MODULE CONTENTS AND LABORATORY ACTIVITIES

This section provides a brief description of the programme of the module "Electrical Energy Systems" (EES) as tought at the 2nd stage of the BSc programmes of Electrical and Electronic Engineering and at the 3rd stage of the BSc programme of Mechanical Engineering of University College Dublin (UCD) by the authors. Reference books of EES are [6-9]. It is highlighted how this particular module would benefit from using Jupyter notebooks for continuous assessment of the students.

3.1 Module Contents

"Electrical Energy Systems" is a core module for three engineering programmes at UCD, namely, Mechanical, Electrical and Electronic Engineering. The three programmes share stage 1, where students are expected to learn a variety of basic subjects, such as calculus, linear algebra, physics and chemistry. During stage 1, introductory modules on mechanics and electrical and electronic engineering are provided to the students. During stage 2, the three programmes include the mandatory module "Electrical and Electronic Circuits", which is offered in semester 1 and provides basic circuit theory concepts, such as Kirchhoff laws and Thevenin theorem. At this point, the path of MEng students and E&E ones start to differ. MEng students focus for the rest of stage 2 and for the first half of stage 3 on matters mostly related to mechanics. It is only in semester 2 of stage 3 that they encounter the module "Electrical Energy Systems".

MEng students might not take any other electrical power modules until the end of their undergraduate studies. Students that choose to pursue the ME in Energy Systems Engineering, which is offered by the School of Mechanical Engineering, can take as options some modules from the ME in Electrical Engineering, such as "Power Systems Dynamics and Control", and in the past also "Power Systems Stability Analysis" (currently discontinued).[1]

"Electrical Energy Systems" is intended to be an introductory module on ac circuits, magnetic circuits and energy conversion, electrical machines and power systems. It includes 30 hours of theoretical lectures, 6 hours of tutorials, and lab activities, for a total of 5 ECTS credits. Due to time limitations, each topic is necessarily just outlined and only very basic definitions and concepts are presented. All topics are explained both theoretically and with problems worked on the blackboard.

---

[1] The interested reader can refer to [10] and [11] for further details on these two modules.

The module is divided into six parts, namely, ac circuit theory including three-phase systems; magnetic circuits; energy conversion; transformers; induction motors; synchronous machines; and power flow analysis. Basic suggested bibliography includes references on electrical machines [6, 7], and power systems [8, 9]. Learning outcomes of the module are: basic concepts of steady-state ac circuits, main electrical machines (transformers, induction motors and synchronous generators) and power system analysis.

In previous years the assessment in the module has been performed through two experimental labs and a midterm test. The authors noticed through examining the students' solutions in the module exams that the midterm exam was not sufficient as the students seemed to not fully grasp each of the module topics. Thus, the authors decided to set up a continuous assessment, through Jupyter notebooks. The lab activities of the EES module in the school year 2018/19 were divided into two parts: (i) open-circuit and short-circuit test of a 3 kVA transformer; and (ii) seven Jupyter notebooks corresponding to the seven parts of the module. Labs on the transformer are organized similarly to what described in [12].

The students have a week to complete each notebook, which is assigned as soon as the respective topic has been covered by the lecturer. The benefits of this approach are twofold. Firstly, the students have the chance to study each topic individually through the notebooks when the topic is fresh in their minds. This approach improves the learning curve of the students. Secondly, the students learn how to solve relevant problems discussed in the module through computer code. At the same time, they improve their programming skills, which is important when pursuing engineering jobs in the future.

4 STUDENTS' FEEDBACK ON LABORATORY ACTIVITIES

This section discusses the feedback provided by the students regarding the utilization of Jupyter notebooks during the laboratory activities of the module EES of the academic year 2018/19. Through the Google forms the students were asked to provide feedback on how easy they found the Jupyter notebooks and the Google forms to use and understand. The feedback results are shown in Figure 3. The students have mixed opinions on using Jupyter notebooks. A little less than 50% of the students found them difficult to use. This is mostly due to the use of Python as is discussed below. On the other hand, Google forms were almost uniformly found to be easy to use and understand.

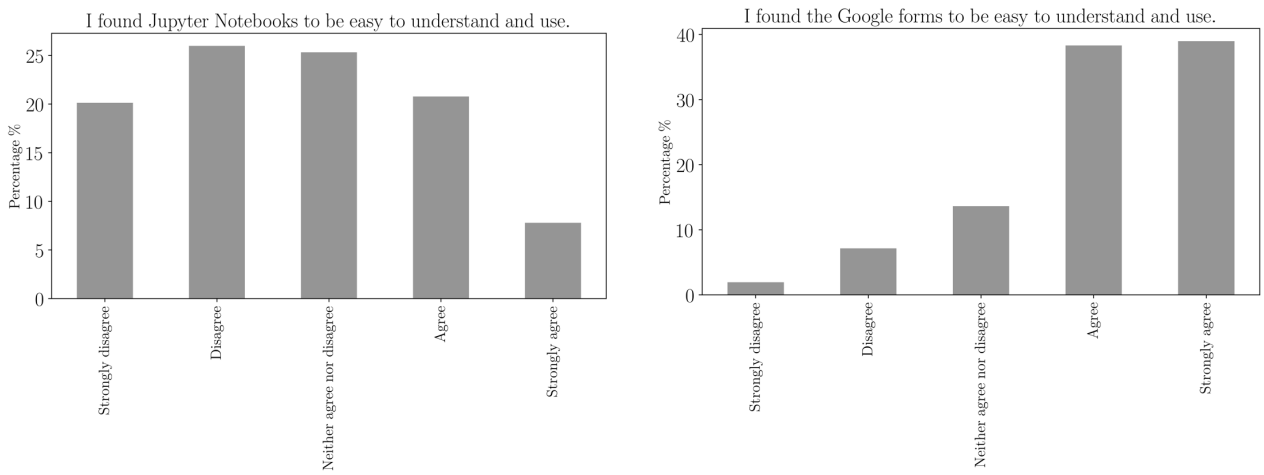

Figure 3. Feedback responses for students on Jupyter notebooks and Google forms.

The students had the option to submit any additional written feedback or concerns they had on the labs through the Google forms. Most often it was the mechanical students who raised concerns. These students find the module in general to be very difficult to grasp and thus struggle to complete these labs. However, the labs do give these students a chance to study each subject independently right after it has been covered in class. This has made it easier for these students to complete the worked problems on the course topics and this ultimately resulted in better grades as shown in Table 1.

Another common concern is on the use of Python from students unfamiliar with this programming language. To help these students, in the coming years, we will include a brief introduction to Python at

the start of the module. This is one of the reasons why so many students found the Jupyter notebooks to be difficult to understand, as shown in Figure 3.

An issue is also raised by some students on the use of multiple platforms, i.e. Jupyter notebooks and Google forms. This is an issue we are aware of. However, Jupyter notebooks at this time do not have the capability to grade the answers as efficiently as Google forms. The plan is to only use Jupyter notebooks in the future when an efficient grading feature, similar to Google forms has been integrated. For the time being the Google forms do not pose much of a problem as less then 10% of the students found them to be difficult to use.

Students also provided some positive feedback. Many liked the use of code to help understand the theory better and specifically liked being able to visualize the course topics through the graphs in the notebooks. These students found Jupyter notebooks to be a good way to learning python and the course material at the same time.

This change in the course format, i.e. having the students study each topic individually through Jupyter notebooks after the lecturer completed it helped prepare the students for the midterm exam. In Table 1, the average midterm grades for the school years 2017/18 (Jupyter notebooks not implemented) and 2018/19 (with Jupyter notebooks) are shown as well as a percentage of the students who passed the exam. Both students' grades and the number of students that have passed the mid-term exam have significantly increased with the introduction of worked problems in Jupyter notebooks.

Table 1. Average midterm grades for the EES course for the school years 2017/18 and 2018/19 and the percentage of student that passed the exam for each year[2].

School year	Average grade	Passed
2017/18	D-	58 %
2018/19	C	80 %

5 CONCLUSIONS

The paper shows that the implementation of a continuous assessment through Jupyter notebooks can improve the students learning experience and their overall performance. The introduction of small bits of Python code in the Jupyter notebooks has also developed students' programming skills. Overall, we believe that Jupyter notebook offers a versatile solution that can seamlessly be adapted to the skills and the interests of the students. Future work will focus on improving the experience of the students by integrating the evaluation scheme in the notebooks (as opposed to use separate Google forms) and providing the students a tutorial on the Python programming language to improve their experience with the Jupyter notebooks.

APPENDIX

This appendix shows the laboratory activity on the induction machine proposed to the students of EES. Section A.1 presents the contents and typical results of a Jupyter notebook while Section A.2 presents the questions asked in the Google form.

A.1 Example of Jupyter notebook

A.1.1 Learning Outcomes

The purpose of this lab activity is twofold:

- Familiarise with the steady-state model of the induction machine
- Determine and plot relevant quantities of the machine (torque, startup current, etc.)

2 To pass the exam the students need to get a D- or higher. The UCD grading scheme is available at: http://www.ucd.ie/registry/assessment/documents/modular%20grades%20explained%20staff.pdf

The relevant lecture material for this lab is found here.

A.1.2 Submission

Please fill in the form in here (link to Google form) with the results from this lab. Make sure to insert your name and student number to identify your solution.

A.1.3 Questions

In this lab a 4-pole, Y-connected, 400V, 53Hz induction machine is studied. The equivalent circuit of the induction machine is shown in Fig. 1.

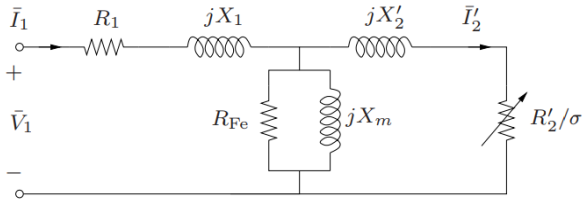

Figure 3: Equivalent circuit of an induction machine.

A.1.3.1 Case 1

Question a: The nominal rotor speed is ω_N = 1500 rpm. Determine the slip factor at full load, σ_N.

```
# Frequency
f = 53 # Hz
# Pole pairs
p = 4/2
# Nominal rotor speed
omega_N = 1500 # rpm
# The primary stator synchronous speed at full load
omega_s1 = 60 * f / p
# Write the formula for the slip factor:
sigma_N = (omega_s1 - omega_N)/omega_s1
print("\u03C3 = {} %".format(sigma_N * 100))
```

Result:

```
σ = 5.660377358490567 %
```

Question b: The following parameters are known: R_1=0.125 Ω, X_1=0.3 Ω, R'_2=0.125 Ω. The torque at full load is T_{mN}=372.7 Nm. Determine the reactance X'_2 through trial and error.

```
from numpy import sqrt, pi
E_1  = 400/sqrt(3)
R_1 = 0.125
R_2i = 0.125
X_1 = 0.3
# Modify X_2i
X_2i = 0.2
X_sci = X_1 + X_2i
# Torque
num1 = 3 * p * E_1**2 * R_2i
den1 = 2 * pi * f * sigma_N * ((R_1 + R_2i/sigma_N)**2 + X_sci**2)
T_mN = num1 / den1
print("X'_2 = {} \u03A9".format(X_2i))
print("T_mN = {} Nm".format(T_mN))
```

Results:

```
X'_2 = 0.2 Ω
T_mN = 372.6554765078525 Nm
```

Question c: At the nominal frequency of the stator f_1=53 Hz (dotted vertical line in Figure 2), what is the resistance R'_2 so that the torque is T_m=440 Nm. Deduce the answer by running the following code.

```
%matplotlib notebook
import matplotlib.pyplot as plt
from numpy import arange
```

```
fx = arange(20, 100 , 0.02)
R_2ix = [0.05, 0.1, 0.15, 0.2, 0.25]
for y in R_2ix:
    num1x = 3 * p * E_1**2 * y
    den1x = 2 * pi * fx * sigma_N * ((R_1 + y/sigma_N)**2 + X_sci**2)
    T_mNx = num1x / den1x
    plt.plot(fx, T_mNx, label='$R{\'}_2 = %3.2f$' % y)
plt.xlabel('$f_1\;\\rm [Hz]$')
plt.ylabel('$T_{m}\;\\rm [Nm]$')
plt.axvline(f, color='k', ls=':')
plt.xlim(left= 20)
plt.xlim(right=100)
plt.legend(loc=1)
```

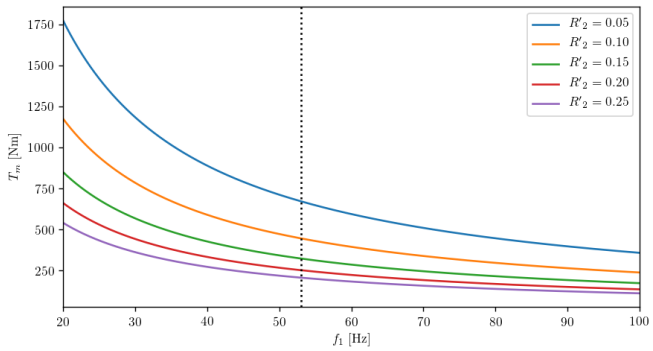

Figure 4. The torque T_m as a function of the frequency of the stator ω_1 for different resistances R'_2.

A.1.3.2 Case 2

Question a: At full load determine the power loss of the machine $P_{m,losses}$ through trial and error so that the mechanical power output is $P_{m,net}$=57,6 kW.

```
# Modify P_losses
P_losses = 900 #W
# Generated mechanical power
P_m = T_mN * omega_N * 2* pi /60
# The mechanical power output
P_net = P_m - P_losses
print("P_losses = {} W".format(P_losses))
print("P_net = {} kW".format(P_net / 1e3))
```

Results:

```
P_losses = 900 W
P_net = 57.63658536585365 kW
```

Question b: Determine the mechanical torque output $T_{m,net}$ from the mechanical power output $P_{m,net}$ found in *Question a*.

```
# Write your code here:
T_net = P_net/(omega_N * 2* pi /60)
print("T_net = {} Nm".format(T_net))
```

Result:

```
T_net = 366.9258985565442 Nm
```

A.1.3.3 Case 3

Question a: At full load determine the rotor current, I'_{2N}.

```
from cmath import polar
# Write your equation here:
I_2Ni = E_1 / ((R_1 + R_2i/sigma_N) + X_sci * 1j)
I_2Ni_z, I_2Ni_ang = polar(I_2Ni)
print("I_2Ni = {} A\u2220{} rad".format(I_2Ni_z, I_2Ni_ang))
```

Result:

```
I_2Ni = 96.77734062546143 A∠-0.21109333322274654 rad
```

Question b: For what iron losses P_{Fe} is the current I_{Fe}=5.8 A?

```
# Modify P_Fe
P_Fe = 1350 #W
I_Fe = P_Fe / E_1
print("P_Fe = {} W".format(P_Fe))
print("I_Fe = {} A".format(I_Fe))
```

Results:

```
P_Fe = 1350 W
I_Fe = 5.84567147554496 A
```

Question c: For what magnetic reactance X_μ is the current through it I_μ=17.8 A $\angle -\pi/2$ rad?

```
# Modify X_mu
X_mu = 13 * 1j # Omega
I_mu = E_1 / X_mu
print("X_mu = {} ".format(X_mu))
print("I_mu = {} A".format(I_mu))
```

Results:

```
X_mu = 13j
I_mu = -17.764623667373105j A
```

Question d: What is the stator current I_{1N}?

```
# Write your equation for I_1N here:
I_1N = I_2Ni + I_Fe + I_mu
I_1N_z, I_1N_ang = polar(I_1N)
print("I_1N = {} A\u2220{} rad".format(I_1N_z, I_1N_ang))
```

Result:

```
I_1N = 107.43555804490664 A∠-0.3619451742768095 rad
```

Plot the machine rotor current as a function of the rotor angular speed for different values of the stator speed by running the code below.

```
%matplotlib notebook
omega_s1x = [500, 900, 1300, 1500]
for y in omega_s1x:
    omega_m = arange(0,y,0.1) # rpm
    sigma_Nx = (y - omega_m)/y
    I_2Nix = E_1**2 / sqrt((R_1 + R_2i/sigma_Nx)**2 + X_sci**2)
    plt.plot(omega_m, I_2Nix, label='$\\omega_1 = %3.2f \;\\rm rpm$' % y)
plt.xlabel('$\\omega_m\;\\rm [rpm]$')
plt.ylabel('$I_{2N}\'\;\\rm [A]$')
plt.xlim(left= 0)
plt.xlim(right=omega_s1)
plt.ylim([0, 100000])
plt.legend(loc=1)
```

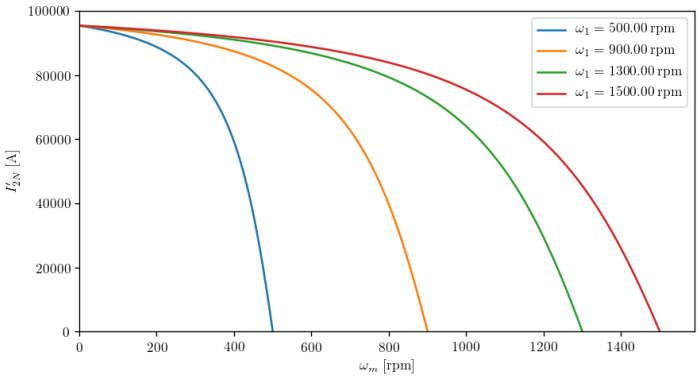

Figure 5. The rotor side current I'_{2N} as a function of the rotor speed ω_m for different stator speeds ω_1.

A.2 Questions of the Google form associated with the Jupyter notebook

Please fill in the following form with your results for Lab 5.

Note: Select the option that has an answer that is closest to the result you obtained in each question.

Case 1

a. Slip factor?	2.1 %	8.6 %	4.3 %	5.7 %	
b. Reactance X'_2?	0.2 Ω	0.1 Ω	0.4 Ω	0.3 Ω	
c. Resistance R'_2?	0.15 Ω	0.05 Ω	0.1 Ω	0.25 Ω	0.2 Ω

Case 2

a. Power loss of the machine?	900 W	1100 W	500 W	700 W
b. Mechanical torque output?	320 Nm	370 Nm	230 Nm	280 Nm

Case 3

a. At full load rotor current I'_{2N}?	107 A∠-0.2 rad 97 A∠-0.2 rad		107 A ∠0.2 rad 97 A ∠0.2 rad	
b. Iron losses?	850 W	1125 W	1350 W	975 W
c. Magnetic reactance?	4 Ω	7 Ω	13 Ω	11 Ω
d. Stator current I_{1N}?	107 A∠-0.37 rad 117 A∠-0.37 rad		107 A∠0.37 rad 117 A∠0.37 rad	

AKNOWLEDGEMENTS

This material is based upon works supported by Science Foundation Ireland, under Investigator Programme, Grant No. SFI/15/IA/3074.

REFERENCES

[1] Jupyter Project, available at http://jupyter.org

[2] F. Milano, G. M. Jónsdóttir, Jupyter notebooks for Computer-based Laboratories on Power System Dynamics and Control, EDULEARN 2018, Palma de Mallorca, Spain, 2-4 July 2018.

[3] JupyterHub Project, available at http://jupyter.org/hub

[4] NBgrader Project, available at https://github.com/jupyter/nbgrader

[5] Google Forms, available at https://www.google.com/forms/about/

[6] G. R. Slemon*, Electric machines* and *drives,* Addison-Wesley, 1992.

[7] J. D. Edwards*, Electrical Machines*: *An Introduction to Principles and Characteristics.* Macmillan, 1986.

[8] O. I. Elgerd*, Electric Energy Systems Theory*: *An Introduction,* McGraw-Hill, 1983.

[9] G. Rizzoni*, Principles and Applications of Electrical Engineering*, Fifth Edition, *McGraw-Hill*, 2010.

[10] F. Milano, Problem Solving through Limit-case Analysis: Experience in Electrical Engineering Programmes, EDULEARN, Barcelona, Spain, 4-6 July 2016.

[11] F. Milano, Thinking Nonlinearly: Experience in Teaching Power System Stability Analysis in Engineering Programmes, EDULEARN, Barcelona, Spain, 4-6 July 2016.

[12] F. Milano*, A Systematic Method for Failure Diagnosis in Electrical Machine Lab Activities*, ICERI 2011, Madrid, Spain, 14-16 November 2011.