APA Thesis LaTeX template

Álvaro Augusto Volpato

São Carlos, Brazil
March 2019

APA Thesis LaTeX template

Template document developed for thesis documents fulfilling the guidelines of the Publication Manual of the American Psychology Association, Sixth Edition

by
Álvaro Augusto Volpato

Advisor: Professor Dr. Luís Fernando Costa Alberto

University of São Paulo
São Carlos School of Engineering
Department of Electrical and Computer Engineering

São Carlos, Brazil
March 2019

Template de tese no estilo APA

Documento padronizado ("template") desenvolvido para uso em teses segundo o Manual de Publicações da Associação Americana de Psicologia, Sexta Edição.

Álvaro Augusto Volpato

Orientador: Professor Luís Fernando Costa Alberto

Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica e de Computação

São Carlos, Brasil
Março de 2019

Volpato, Álvaro Augusto
APA Thesis LaTeX template / Álvaro Augusto Volpato; advisor Luís Fernando Costa Alberto - São Carlos 2019.
51 p.

Dissertation (Master's Degree - Graduate Program in Electrical Engineering) -- São Carlos School of Engineering, University of São Paulo - Brazil, 2019

1. LaTeX. 2. American Psychology Association. I. Alberto, Luís Fernando Costa, advisor. II. Title.

Abstract

This is the abstract in english.
Keywords: keyword 1, keyword 2, keyword 3 ...

Zusammenfassung

Zusammenfassung aus Deutsch.

Stichwörter: Stichwort 1, Stichwort 2,...

Resumo

Resumo em português.
Palavras-chave: palavra-chave 1, palavra-chave 2, ...

List of Figures

1 *Root folder folder tree.* 26
2 `/tex` *folder tree.* 26

3 *Five-centimeter-wide EESC logo.* 32
4 *Five centimeter wide, three centimeter tall EESC logo* 33
5 *EESC logo with a quarter the text column width.* 33
6 *EESC logo rotated 45 degrees.* 34

7 *Panel transconductance (continuous) and MPP (dashed) curves with fixed temperature θ_{SRC} and varying irradiance.* . . 49

List of Tables

1 *Age and height example table.* 35

Abbreviations and Acronyms

OMIB	*One Machine Infinite Bus System*
CG	*Centralized Generation*
DG	*Distributed Generation*
RPGTs	*Renewable Power Generation Technologies*
MASs	*Multi-Agent Systems*
EPSs	*Electric Power Systems*
ESSs	*Energy Storage Systems*
MPPT	*Maximum Power Point Tracking*
PV	*Photovoltaic*
PBD	*Pinning-based Droop*
SM	*Synchronous Machine*
OAM	*One-Axis Model*
ICMPPT	*Incremental Conductance Maximum Power Point Tracking*

List of Symbols

Γ	Gamma greek letter
Λ	Lambda greek letter
ζ	Lowercase Zeta greek letter
$\in$	Set theory belonging/contained in relation
$\lvert \cdot \rvert$	Complex absolute value
$\lVert \cdot \rVert$	Complex vector or matrix euclidian norm

Contents

ABSTRACT 7

ZUSAMMENFASSUNG 9

RESUMO 11

LIST OF FIGURES 13

LIST OF TABLES 15

ABBREVIATIONS AND ACRONYMS 17

LIST OF SYMBOLS 19

1 FIRST STEPS: SETTING UP YOUR THESIS 25
- 1.1 How the template is organized . . . 25
- 1.2 Changing page geometry . . . 27
- 1.3 Changing fonts used . . . 27
- 1.4 How to build the main.tex file . . . 27
  - 1.4.1 *In a command-line* . . . 27
  - 1.4.2 *In a dedicated TeX editor* . . . 28
- 1.5 Setting up university logo, thesis title, front matter and abstracts . . . 28
  - 1.5.1 *Change basic metadata: title, names and university logo* . . . 28
  - 1.5.2 *Second front matter* . . . 28
  - 1.5.3 *Catalographic card* . . . 28
  - 1.5.4 *Abstracts* . . . 28
    - *Modifying abstract properties* 29
  - 1.5.5 *Acronyms and symbols* . . . 29
  - 1.5.6 *Epigraph* . . . 29
  - 1.5.7 *First compile and adding packages* . . . 30
- 1.6 Adding text chapters and appendixes . . . 30

2 FIGURES AND TABLES 31
- 2.1 Adding figures . . . 31
  - 2.1.1 *Figure placement* . . . 32
  - 2.1.2 *Scaling figures* . . . 32
  - 2.1.3 *Rotating figures* . . . 33
- 2.2 Adding tables . . . 33

2.2.1 Coding your table *34*

BIBLIOGRAPHY ***37***

Appendixes

APPENDIX A PV Panel Curves and simulation program **41**

APPENDIX B Second appendix **51**

“An optimist will tell you the glass is half-full.
A pessimist will say it’s half-empty.
An engineer will tell you the glass is twice the size it needs to be.”

— Unknown Author

CHAPTER 1

First steps: setting up your thesis

THIS template is a joint effort between Álvaro Augusto Volpato — graduate Electrical Engineering student from São Carlos School of Engineering (EESC) — and Eduardo Graziozi Silva and Flavia Helena, from EESC's Library staff, to make an APA-style thesis template that abides to the formatting rules of the São Carlos School of Engineering at the University of São Paulo (EESC-USP). While Alvaro is the developer and maintainer, Eduardo and Flavia are the ones responsible for usage and distribution of the class throughout the University of São Paulo and specifically EESC. It is meant as a template for usage in thesis and dissertations written in English and Portuguese, although it can easily be translated to other languages and offers multi-language support.

This template was written using the guidelines of the Publication Manual of the American Association of Psychology, Sixth Edition, which can be obtained at APA's official page. The template example together with its documentation and files can be obtained and downloaded at its repository. There you can clone the repository, fork it, submit pull requests and so on. This documentation is meant to offer guidelines to use this template: read it carefully to understand and use it how the author intended.

To contact Álvaro for suggestions and feature requests, contact him at:

- His e-mail `alvaro.volpato@usp.br`;
- His GitHub page: `http://github.com/Gondolindrim`.

This first chapter will show the first steps in setting yout document in this template: how the template works and how it is organized, how to to change basic features of the template, like page size, margins, fonts, how to set title, author, university name, abstracts and epigraph.

1.1 How the template is organized

The template folder tree is organized in three basic folders, at the root folder, as depicted in figure 1.

- The `/images` folder stores all graphics and image-related files. These are included in the document through a special commend
  `includegraphics`;
- The `/scripts` folder stores the codes, algorithms and listings that will be displayed in the thesis. These codes are displayed through a dedicated command
  `lstinputlisting`, which is used with a customized style defined in the class file;

```
/
├── /images
├── /scripts
└── /tex
```

Figure 1. Root folder folder tree.

```
/tex
├── apaThesis.cls
├── main.tex
├── refs.bib
├── /chapters
│   ├── introduction.tex
│   ├── figs_and_tables.tex
│   ├── equations_and_math.tex
│   └── bibliography.tex
├── /appendixes
│   ├── appendixA.tex
│   └── appendixB.tex
└── /build
    ├── main.pdf
    └── Output files (*.toc, *.bbl, *.out et cetera)
```

Figure 2. `/tex` folder tree.

- The `/tex` folder stores all TeX-related files, including the thesis text, the class file, appendixes.

These settings can be changed anytime and even throughout the document (see sections **??** and **??**); they were adopted as default in order to facilitate organizing data. It is recommended that the user keeps the folder tree intact in order to coherently add graphics and scripts to the document.

The raw script of this template lies in the `tex` folder, as depicted in figure 2.

In the `/tex` folder tree,

- `apaThesis.cls` is the TeX class file that stores the APA style guidelines and formats the document. It is invoked in the vey beggining of `main.tex`. It is recommended that this file is not edited nor moved/deleted unless needed;
- `main.tex` is the main LaTeX file and describes the organization and contents of the document. This is where the pre-textual elements (title, front matter, catalographic card, abstracts, lists of figures, tables, acronyms and symbols, table of contents) are set-up, invoked and built;
- `refs.bib` is the bibliography file, containing the references in BibTeX format;
- `/tex/chapters` folder contains the text contents of the chapters, each chapter separated into a single *.tex file. This is done so as to make editing easier; otherwise, all text content is added to a single main.tex file which becomes too big and harder to maintain, specially when using versioning softwares like Git;
- `/tex/appendixes` folder contains the appendixes text contents;

- `/tex/build` folder contains the final build files, including the *.pdf file generated and other build files. It is recommended that the build folder be separated because those output files can flood the file tree and difficult maintenance.

1.2 Changing page geometry

The default implementation of the example uses the US Letter Paper size with 1 inch margins, as recommended by APA in their Publication Manual. This is done in the first lines of the `main.tex` file, by calling the class file with no options, as in listing 1.1.

Listing 1.1: Using the class with US Letter paper size

```
\documentclass{apaThesis}
```

There is also an option for A4 paper, by invoking the `a4paper` option as in listing 1.2.

Listing 1.2: Using the class with A4 paper size

```
\documentclass[a4paper]{apaThesis}
```

These two sizes should cover most universities requirements. If however a custom size needs to be used, you can declare a custom size in the first section of the class file, you can comment line 67 of the class file, uncomment line 70 and use the `paperwidth` and `paperheight` options, editing the height and width options as desired. You can also alter margin values if needed.

1.3 Changing fonts used

The fonts definitions of the class lie in lines 81-90 of the class file, in the `(2) MAIN FONTS`. Any of the used fonts can be changed by the user; for a list of LaTeX supported fonts, see the LaTeX font Catalogue.

- Font Rosario is used as the sans serif for titles and some highlights. This font is called by using the command `\usepackage{Rosario}`;
- Font Times New Roman is used as serif font for most of the text body. This font is called by the `\usepackage{times}`. Times font is also used as the main math font by using `\usepackage{newtxmath}`
- Font Inconsolata is used as the monospace font. This font is called by the `\usepackage{inconsolata}`.
- `\usepackage{lettrine}` is used to display a big letter at the beggining of every chapter.

1.4 How to build the main.tex file

Two things must be noted when building the files: the first, that the build folder must be specified (otherwise the output files will be generated at the `/tex` folder, flooding it) and that the PDFLaTeX engine must be used (neither XeLaTex nor LuaLaTeX will work).

1.4.1 In a command-line

In Linux and Windows' command prompt, when using the shell script, building the main file is done by using the following command in the `/tex` folder:

```
pdflatex --output-directory=build main.tex
```

If you need to invoke PDFLaTeX from the root folder, use:

```
pdflatex --output-directory=./tex/build ./tex/main.tex
```

1.4.2 In a dedicated TeX editor

If an editor like TeXMaker, Lyx or TeXStudio is used, these parameters (using PDFLaTeX and the build directory) must be set expressly in the editor configurations. If Overleaf/ShareLaTeX is used, then this configuration is not needed as the platform generates output in real-time and the PDF file can be downloaded at any time.

1.5 Setting up university logo, thesis title, front matter and abstracts

After setting up the basic geometry of the document, and before inputting the text body, you should input your name, thesis title, advisor name, university logo and name, and so on. This section will go through a step-by-step way to do this.

1.5.1 Change basic metadata: title, names and university logo

First, go to the first section of `main.tex` called `''(1) DOCUMENT DATA''` and edit the data accordingly. Beware that the preamble text must generally follow a convention by your university or institute, so be sure to check past thesis and use the same format. Also be sure to correctly write the affiliation out; generally, the first line is the university name, second line is the institute name and third line is the department or research group name.

The university logo can be changed by simply replacing the `./images/uniLogo.pdf` file with your university's or institute's logo. You might need to adjust this logo size by adjusting the `\uniLogoWidth` command in line 38 of `main.pdf`. The default value is `0.2\textwidth`.

1.5.2 Second front matter

Many universities will require that after the english front matter, a second front matter is added in the native language of that institute or university. This is done by editing the parameters in lines 55-64 of `main.pdf`. If you don't need a second frontmatter, commend out these lines.

1.5.3 Catalographic card

The catalographic card is a piece of meta data used by libraries to classify and organized their stored works. Generally they follow a very similar structure throughout the world.

As I was not able to automatize this card, it is needed that you edit it accordingly to the structure your university requires. The example pattern is used throughout the world and should be very common; all you need is to change the names accordingly.

1.5.4 Abstracts

Adding an abstract is easily done through the `newabstract` environment. This environment is defined in the class file so that you can add as many abstracts as you wish, in as many languages.

To add a new abstract in english, use the following code:

Listing 1.3: Adding an abstract in english

```
\begin{newabstract}{Abstract}
  This is the abstract in english. \\
\noindent
\textbf{Keywords}: keyword 1, keyword 2, keyword 3 ...
\end{newabstract}
```

To add an abstract in another language, use the other language names and abstract body; for example, to add an abstract in german, use:

Listing 1.4: Adding an abstract in german

```
\begin{newabstract}{Zusammenfassung}
Zusammenfassung aus Deutsch. \\
\noindent
\textbf{Stichw\"{o}rter}: Stichwort 1, Stichwort 2,...
\end{newabstract}
```

Beware that in most universities the english abstract will be needed, and an abstract in another language is optional. Either case, at least an abstract should be devised.

Modifying abstract properties

The `newabstract` environment is defined in section ``(13) NEWABSTRACT ENVIRONMENT'' of the class file. The default implementation is accepted widely and will most probably fit your university's requirements, but can be changed as desired.

1.5.5 Acronyms and symbols

Next are the lists of acronyms and symbols. The example shows how to add acronyms:

Listing 1.5: Adding an acronym

```
%\acronym{<acronym>}{<What the acroym means>}
\acronym{SG}{Synchronous Generator}
```

The procedure is the same for symbols:

Listing 1.6: Adding an acronym

```
%\item{<symbol>}{<What the symbol means>}
\item{$k_B$}{Boltzmann's constant}
```

Also note that the `<symbol>` key can be a math expression, in between `$` characters (the example document contains examples like this).

If acronyms and/or symbols are not needed, you can comment or delete the `acronyms` (lines 170-184 in the `main.tex` document) and `listofsymbols` (lines 186-193 in the `main.tex` document) environments.

1.5.6 Epigraph

The epigraph is that little sentence or thought you add in italics, generally in latin, with the sole purpose of sounding smart.

To add an epigraph, use the `newepigraph` environment as used in line 211 of `main.tex`. If you don't want to sound smart, you can comment the environment (lines 211-219).

1.5.7 First compile and adding packages

This is the end of the initial setting up steps. You should be able to compile your document now – albeit with no text – and see if it meets your requirements. The next chapter will focus on how to add text body.

In order to add your packages to the document, use the `\usepackage[...]{...}` commands before the `\begin{document}` command, ideally right below `\documentclass{apaThesis}`.

1.6 Adding text chapters and appendixes

The text body can be added as a normal LaTeX text.

The main example uses the `\input` command, which allows you to input text from another file into your `main.pdf` file. This allows you to write each piece of text in a dedicated file, making maintenance easier. If a single `main.pdf` is used for the whole document, it can grow too big and make version management difficult, specially with Git, by generating conflicts between commits.

The figure, table and listing labels are kept throughout the whole document even if they are defined in different files; for example, say that in `chapter3.tex` you defined a figure with label `fig:graph`. You can call this same label in `chapter2.tex` just as if the chapters were written in the same file and not in separate files. For example, figure 7 is defined in the `/tex/appendixes/appendixA.tex` while this introduction chapter file is `/tex/chapters/introduction.tex` and the figure label can be used by typing `\ref{fig:ivCurve}`, which is how its label is defined in the appendix file.

To add chapters and text body, use the syntax in listing 1.7. Add that command after the `/begintextbody` command and between `\bibliography`.

Listing 1.7: Adding a chapter to text body

```
%\input{<Chapter *.tex file>}
\input{./chapters/chapter1.tex}
```

Note that the chapter files do not need headings or packages, as all configuration is inherited by them from main.pdf. By default in the template, chapter files are added to the `/tex/chapters` folder, but you can create further folders and add them to your `input` command.

This process is the same for adding appendixes; however, those should be added after the `\appendix` command, since this command makes appendixes be numbered in progressive letters (Appendix A, Appendix B *et cetera*). See listing 1.8 for details.

Listing 1.8: Adding an appendix to text body

```
\part*{Appendixes}
\appendix

%\input{<Appendix *.tex file>}
\input{./appendixes/appendixA.tex}
\input{./appendixes/appendixB.tex}
```

CHAPTER 2

Figures and tables

This chapter verses on how to add figures, tables and scripts to your document.

Adding those features is pretty straightforward as they are already formatted to the APA style. The only thing you should take care when adding these elements is the element size in relation to the page, that is, if the figure or table does not extend past the margins.

For figures, this is generally a tradeoff between having legible text in the figure and having enough page space, meaning that you should adjust the image until it is readable and legible and its size is correct.

2.1 Adding figures

To add a figure to the document, use syntax in listing 2.1.

Listing 2.1: Basic code for adding a figure.

```
\begin{figure}[h]
  \centering
  \includegraphics[<Scaling and angle options>]{<Figure file location>}
  \caption{<Figure caption>}
  \label{<Human-readable label>}
\end{figure}
```

`\centering` forces figure centering; `\caption{}` is the figure caption that explains or defines it; `\label{}` is the human-readable label that you use to reference the image in the text.

The command `\includegraphics[]{}` is the command to insert the figure. This command defines the size of the inserted image; for example, using `\width = 5cm` will add a five-centimeter-wide figure that is also vertically scaled. See listing 2.2; figure 3 shows the output of that code.

Listing 2.2: 5cm-wide EESC logo (figure 3).

```
\begin{figure}[h]
  \centering
  \includegraphics[width = 5cm]{../images/uniLogo.pdf}
  \caption{Centimeter-wide EESC logo}
  \label{fig:5cmEESC}
\end{figure}
```

This procedure is highly customizable, according to the parameters of the various environments and commands used.

Figure 3. Five-centimeter-wide EESC logo.

2.1.1 Figure placement

The `figure` environment makes a floating figure that is adjusted by the LaTeX engine . The `[h]` option defines that the figure should be placed preferably where it is placed in the source text, but that does not mean it will necessarily be placed there. For more placing options, see this StackExchange page. There are also other options, like:

- The `[t]` option will position the figure at the top of the page;
- The `[b]` option will position the figure at the bottom of the page;
- The `[p]` option will position the figure at a special page reserved for floating environments only;
- The `[H]` option forces the engine to place the figure at *exactly* the location it was defined in the source file.

2.1.2 Scaling figures

The `\includegraphics` command has a variety of options, like `width` and `height`, used too adjust the width and height of the image; if only one is used, the image is scaled to that dimension. For example, in listing 2.2 only the width option was used, so the image height was scaled so that the width would be the specified value. If, however, both height and width values are specified, the image will be adjusted for the given values. Listing 2.3 shows an example of both values given and image 4 shows the result. Note that the image looks flattened because the height and width values given are not proportional to the original proportions of the image.

Listing 2.3: 5cm-wide EESC logo (figure 4)

```
\begin{figure}[h]
  \centering
  \includegraphics[width = 5cm, height = 3cm]{../images/uniLogo.pdf}
  \caption{Five centimeter wide, three centimeter tall EESC logo}
  \label{fig:53cmEESC}
\end{figure}
```

To make it easier for you to scale graphics in relation to the text column width, this length is given in LaTeX by the command `\textwidth`; for instance, to insert an image that has width of a quarter the text width, use `width = 0.25\textwidth`, as in listing 2.4. Figure 5 shows the output of such code. Giving height and

Figure 4. Five centimeter wide, three centimeter tall EESC logo

width values in fractions of the text column width guarantees that the image will look more organic and better placed.

Listing 2.4: EESC logo with quarter the text column width (figure 5)

```
\begin{figure}[h]
  \centering
  \includegraphics[width = 0.25\textwidth]{../images/uniLogo.pdf}
  \caption{EESC logo with a quarter the text column width}
  \label{fig:quarterTextEESC}
\end{figure}
```

Figure 5. EESC logo with a quarter the text column width.

2.1.3 Rotating figures

The `\includegraphics` command also has an `angle` option which rotates the image by a given angle, measured in degrees. Listing 2.5 shows a 45-degree rotated logo, and figure 6 shows the results.

Listing 2.5: EESC logo rotated 45 degrees (figure 6)

```
\begin{figure}[h]
  \centering
  \includegraphics[width = 3cm, angle = 45]{../images/uniLogo.pdf}
  \caption{EESC logo rotated by 45 degrees.}
  \label{fig:45degreeEESC}
\end{figure}
```

2.2 Adding tables

Tables are a straightforward way to display data. The APA Publication Manual does recommend a particular table format, which is achieved by the code in listing 2.6. The `table` environment is pretty much alike the

Figure 6. EESC logo rotated 45 degrees.

figure one in that it takes the same alignment options, caption and label convention. So all alignment options for the figures apply here, as well as caption and label.

- The `table` environment sets a floating type for the table which positioning parameters are the same as figures;
- The `tabular` environment effectively code the table;
- The `<Column alignment options>` are passed to the tabular environment to define how many columns a table has and each columns alignment option;
- The `hline` command prints a horizontal line in the table spanning all of its width;
- The `\caption` command defines the table caption;
- The `\label` command defines a human-readable label by which the table can be referenced in the text.

Listing 2.6: Basic table code

```
\begin{table}[h]
  \begin{center}
    \begin{tabular}{<Column alignment options>}
      \hline
      <Header code> \\
      \hline
      <Rows code>\\
      \hline
    \end{tabular}
    \caption{<Table caption>}
    \label{<Human-readable label>}
  \end{center}
\end{table}
```

2.2.1 Coding your table

Table coding is fairly simple, with a basic rule: cells are separeted by a & character and rows are broken using a double backslash sequence \\. Table 1 shows an example table used to exemplify table coding; this table has three columns with the first one left-aligned and the other ones center-aligned.

Before inserting table data, is it important to input first the number of columns the table will have and the alignment of each column. This is done by passing the alignment options to the `tabular` environment.

Each column can be centered (`c`), left aligned (`l`) or right aligned (`r`). In the example table of listing 1, this is done by passing the `{l c c}` option, which defines the table will have three columns and each columns alignment.

According to the APA Publication Manual, tables should have boldface headers. The header row should be separated by a horizontal line, at its top and bottom. Boldface is achieved by the `\textbf{}` command and the horizontal line is achieved by the `hline` command.

Listing 2.7: Example table (table 1)

```
\begin{table}[h]
  \begin{center}
    \begin{tabular}{l c c}
      \textbf{Name} & \textbf{Age} & \textbf{Height} (m) \\
      \hline
      John & 25 & 1.80\\
      Mary & 34 & 1.72\\
      Janett & 14 & 1.54\\
      \hline
    \end{tabular}
    \caption{Age and height example table.}
    \label{tab:exampleTable}
  \end{center}
\end{table}
```

Name	**Age**	**Height** (m)
John	25	1.80
Mary	34	1.72
Janett	14	1.54

Table 1
Age and height example table.

First of all

Bibliography

Appendixes

APPENDIX A

PV Panel Curves and simulation program

This appendix shows the simulation curves of the photovoltaic panel static equation (**??**). It also shows listing A.1, the Python program used to obtain those curves. The parameter values of the photovoltaic panels are denoted in table **??**.

Listing A.1: Python PV panel simulation program developed to generate figures 7 to **??**

```
#-----------------------------------------------------------------
# PV PANEL MAXIMUM POWER POINT SOLVER
#-----------------------------------------------------------------
# PV Panel MPP Solver
# Author: Alvaro Augusto Volpato
# Description: this script intends to find a PV panel's MPP by solving its nonlinear
    equations.

# Header: importing libraries ----------------------------------------
import math
import numpy as np
from scipy.optimize import fsolve
linspace = np.linspace
logspace = np.logspace
import matplotlib.pyplot as pyplot
import matplotlib as mplot

mplot.rcParams['text.usetex'] = True
mplot.rcParams['text.latex.unicode'] = True


exp = np.exp
sqrt = np.sqrt

# Definig the parameters ---------------------------------------------
IrrSrc = 8.487;     # Irradiation curent at SRC
I0Src = 6.33*10**-9;    # Diode reverse current at SRC
Rs = 5.125*10**-3;    # Cell series resistance
Rp = 5.837;     # Cell parallel resistance
n = 1.149;      # Diode ideality factor
Ns = 50;      # Number of cells in series
Np = 1;       # Number of cells in parallel
GSrc = 1000;      # Irradiance at SRC
Eg0 = 1.17*1.6*10**-19;   # Silicon bandgap energy
alpha = 7.01*10**-4*1.6*10**-19;# Bandgap energy factor alpha
beta = 1108;      # Bandgap energy factor beta
```

```
k = 1.38*10**-23;   # Boltzmanns constant
ThetaSrc = 25 + 273.15;   # Temperature at SRC
mu = 0.05/100*IrrSrc;     # Short Circuit current temperature factor
q = 1.6*10**-19;    # Electron fundamental charge

Rs *= Ns/Np;      # Making the resistance transformation
Rp *= Ns/Np;      # Idem


# Defining the system nonlinear function ----------------------------

def FSim(V,I,G,Theta):    # Defining simulation nonlinear function
  return -I + Np*G/GSrc*(IrrSrc + mu*(Theta - ThetaSrc)) - Np*I0Src*(ThetaSrc/Theta)
      **(3/n)*exp(q/(n*k)*(Eg0 - alpha*Theta**2/(Theta + beta))*(1/ThetaSrc - 1/Theta))
      *( exp(q*(V + Rs*I)/(Ns*n*k*Theta)) - 1 ) - (V + Rs*I)/Rp;

def FMpp(z,G,Theta):    # Defining nonlinear function to find MPP
  V = z[0]
  I = z[1]

  F0 = -I + Np*G/GSrc*(IrrSrc + mu*(Theta - ThetaSrc)) - Np*I0Src*(ThetaSrc/Theta)
      **(3/n)*exp(q/(n*k)*(Eg0 - alpha*Theta**2/(Theta + beta))*(1/ThetaSrc - 1/Theta))
      *( exp(q*(V + Rs*I)/(Ns*n*k*Theta)) - 1 ) - (V + Rs*I)/Rp;
  F1 = I + (Rs*I - V)/Rp + Np*I0Src*(ThetaSrc/Theta)**(3/n)*exp(q/(n*k)*(Eg0 - alpha*
      Theta**2/(Theta + beta))*(1/ThetaSrc - 1/Theta))*exp(q*(V + Rs*I)/(Ns*n*k*Theta))
      *(q*(Rs*I - V)/(Ns*n*k*Theta));
  return [F0,F1]

# Function to set pretty minor and major ticks ----------------------

phi = (sqrt(5) + 1)/2

def prettyAxis(ax):
  ax.get_xaxis().set_minor_locator(mplot.ticker.AutoMinorLocator())
  ax.get_yaxis().set_minor_locator(mplot.ticker.AutoMinorLocator())
  # Seting axis lines x = 0 and y = 0 ---------------------------------
  ax.axhline(0,color = 'black', linewidth = 0.5)
  ax.axvline(0,color = 'black', linewidth = 0.5)
  # Setting major and minor grids -------------------------------------
  ax.grid(which = 'major', color = 'k', linestyle = '-', alpha = 0.5, linewidth = 0.5)
  ax.grid(which = 'minor', color = 'k', linestyle = ':', alpha = 0.5, linewidth = 0.5)
  # Setting axis and tick labels --------------------------------------
  ax.tick_params(labelsize = 16)
  # Setting axis aspect ratio -----------------------------------------
  xleft, xright = ax.get_xlim();
  ybottom, ytop = ax.get_ylim();

  ax.set_aspect(abs(1/phi*(xright - xleft)/(ytop - ybottom)))
  return ax

# Initializing figures ----------------------------------------------
# I-V curve figure varying irradiance
fig1 = pyplot.figure();
ax1 = fig1.add_axes([0.1,0.1,0.8,0.8]);

# P-V curve figure varying irradiance
fig2 = pyplot.figure();
ax2 = fig2.add_axes([0.1,0.1,0.8,0.8]);
```

```
# I-V curve figure varying temperature
fig3 = pyplot.figure();
ax3 = fig3.add_axes([0.1,0.1,0.8,0.8]);

# P-V curve figure varying temperature
fig4 = pyplot.figure();
ax4 = fig4.add_axes([0.1,0.1,0.8,0.8]);

# I-V curve figure varying temperature and irradiance
fig5 = pyplot.figure();
ax5 = fig5.add_axes([0.1,0.1,0.8,0.8]);

# P-V curve figure varying temperature and irradiance
fig6 = pyplot.figure();
ax6 = fig6.add_axes([0.1,0.1,0.8,0.8]);

# Open-circuit voltage curve
fig7 = pyplot.figure();
ax7 = fig7.add_axes([0.1,0.1,0.8,0.8]);

# Efficiency curve
fig8 = pyplot.figure();
ax8 = fig8.add_axes([0.1,0.1,0.8,0.8]);

# ------------------------------------------------------------------
# MPP TRACKING VARYING IRRADIANCE
# ------------------------------------------------------------------
# Defining plot points and irradiance values ------------------------
size = 10**3;     # Number of plot points
g = linspace(10,1000,size,1); # Array of irradiance values

# Preallocating solution vector
v = [0]*size;
i = [0]*size;
p = [0]*size;
guess = [25,4]    # MPP initial guess
for m in range(len(g)):

  # Defining funtion to be solved -----------------------------
  def f(z):
    G = g[m];
    Theta = ThetaSrc;
    return FMpp(z,G,Theta)

  sol = fsolve(f,guess) # Solving
  v[m] = sol[0];
  i[m] = sol[1];
  p[m] = sol[0]*sol[1];
  guess = sol;  # Updating next current guess

ax1.plot(v,i, '--', linewidth = 1.5, color = 'black');
ax2.plot(v,p, '--', linewidth = 1.5, color = 'black');

# ------------------------------------------------------------------
# MPP TRACKING VARYING TEMPERATURE
# ------------------------------------------------------------------
# Defining plot points and temperature values -----------------------
size = 10**3 + 1;       # Number of plot points
t = linspace(-100 + 273.15,100 + 273.15,size,1);  # Array of irradiance values
```

```
# Preallocating solution vector
v = [0]*size;
i = [0]*size;
p = [0]*size;
guess = [25,4]    # MPP initial guess
for m in range(len(t)):

  def f(z): return FMpp(z,GSrc,t[m])  # Defining funtion to be solved

  sol = fsolve(f,guess) # Solving
  v[m] = sol[0];
  i[m] = sol[1];
  p[m] = sol[0]*sol[1];
  guess = sol;  # Updating next current guess

ax3.plot(v,i, '--', linewidth = 1.5, color = 'black');
ax4.plot(v,p, '--', linewidth = 1.5, color = 'black');


# Defining plot points and irradiance values ----------------------
size = 100;     # Number of plot points
g = linspace(100,1000,10,1);  # Array of irradiance values

# ----------------------------------------------------------------
# IV-PV CURVE VARYING IRRADIATION
# ----------------------------------------------------------------
# Defining color map for plots ------------------------------------
cmap = mplot.cm.get_cmap('Spectral');
cindex = linspace(0,1,len(g),1);
colors = [0]*len(g);
for i in range(len(g)):
  colors[i] = cmap(cindex[i])

for m in range(len(g)):

  # Calculating open-circuit voltage for a given irradiance ---
  def f(V):
    I = 0;
    return FSim(V,I,g[m],ThetaSrc)

  Voc = fsolve(f,30);

  # Defining and preallocating vectors ------------------------
  v = linspace(0,Voc,size,1); # Defining the voltage array for points
  i = [0]*size;     # Preallocating current solution vector
  p = [0]*size;     # Preallocating power solution vector
  guess = IrrSrc*g[m]/GSrc; # Current initial guess

  # Solving equation through voltage list ---------------------
  for j in range(len(v)):
    def f(I): return FSim(v[j],I,g[m],ThetaSrc) # Implicit function to solve

    i[j] = fsolve(f,guess)  # Solving
    p[j] = v[j]*i[j]; # Obtainig corresponding power
    guess = i[j];   # Updating next current guess

  ax1.plot(v,i, linewidth = 1.5, color = colors[m], label = format(g[m],'.0f'));
  ax2.plot(v,p, linewidth = 1.5, color = colors[m], label = format(g[m],'.0f'));
```

```
# ------------------------------------------------------------------
# IV-PV CURVE VARYING TEMPERATURE
# ------------------------------------------------------------------
# Defining plot points and temperature values ----------------------
size = 100;     # Number of plot points
t = linspace(-100 + 273.15,100 + 273.15,11,1);  # Array of irradiance values

# Defining color map for plots -------------------------------------
cmap = mplot.cm.get_cmap('Spectral');
cindex = linspace(0,1,len(t),1);
colors = [0]*len(t);

for i in range(len(t)):
  colors[i] = cmap(cindex[i])

Voc = 30;   # Open-circuit voltage initial guess
for m in range(len(t)):
  # Calculating open-circuit voltage for a given irradiance ----------
  def f(V): return FSim(V,0,GSrc,t[m])
  Voc = fsolve(f,Voc);

  # Defining and preallocating vectors -------------------------------
  v = linspace(0,Voc,size,1); # Defining the voltage array for points
  i = [0]*size;     # Preallocating current solution vector
  p = [0]*size;     # Preallocating power solution vector
  guess = IrrSrc;     # Current initial guess

  for j in range(len(v)):
    def f(I): return FSim(v[j],I,GSrc,t[m]) # Implicit function to solve

    i[j] = fsolve(f,guess)  # Solving
    p[j] = v[j]*i[j]; # Obtainig corresponding power
    guess = i[j];   # Updating next current guess

  ax3.plot(v,i, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0f'
     ));
  ax4.plot(v,p, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0f'
     ));

# ------------------------------------------------------------------
# IV-PV CURVE VARYING IRRADIATION AND TEMPERATURE
# ------------------------------------------------------------------
t = linspace(-40 + 273.15,80 + 273.15,13,1);  # Array of temperature values
size = 10**3;
g = linspace(10,1000,size,1);     # Array of irradiance values

# Defining color map for plots -------------------------------------
cmap = mplot.cm.get_cmap('Spectral');
cindex = linspace(0,1,len(t),1);
colors = [0]*len(t);
for i in range(len(t)):
  colors[i] = cmap(cindex[i])

guess = [25,0];
for m in range(len(t)):

  # Defining and preallocating vectors -------------------------------
  v = [0]*size;     # Defining the voltage array for points
```

```
  i = [0]*size;     # Preallocating current solution vector
  p = [0]*size;     # Preallocating power solution vectora

  for a in range(len(g)):
    def f(z): return FMpp(z,g[a],t[m]) # Defining function to be solved

    sol = fsolve(f,guess) # Solving
    v[a] = sol[0];
    i[a] = sol[1];
    p[a] = sol[0]*sol[1];
    guess = sol;  # Updating next current guess

  ax5.plot(v,i, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0f'
      ));
  ax6.plot(v,p, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0f'
      ));

# ------------------------------------------------------------------
# Open-circuit voltage
# ------------------------------------------------------------------
# Defining plot points and temperature values ----------------------
size = 1000;      # Number of plot points
g = linspace(1,GSrc,size,1);      # Array of irradiance values
t = linspace(-40 + 273.15,80 + 273.15,13,1);  # Array of temperature values

# Defining color map for plots -------------------------------------
cmap = mplot.cm.get_cmap('Spectral');
cindex = linspace(0,1,len(t),1);
colors = [0]*len(t);

for i in range(len(t)):
  colors[i] = cmap(cindex[i])

guess = 30;   # Open-circuit voltage initial guess

for m in range(len(t)):
  Voc = [0]*size;
  for a in range(len(g)):
    # Calculating open-circuit voltage for a given irradiance
    def f(V): return FSim(V,0,g[a],t[m])
    Voc[a] = fsolve(f,guess);
    guess = Voc[a];

  ax7.plot(g,Voc, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0
      f'));

# ------------------------------------------------------------------
# EFFICIENCY
# ------------------------------------------------------------------
t = linspace(-40 + 273.15,80 + 273.15,13,1);  # Array of temperature values
size = 10**3;
g = linspace(1,1000,size,1);      # Array of irradiance values

# Defining color map for plots -------------------------------------
cmap = mplot.cm.get_cmap('Spectral');
cindex = linspace(0,1,len(t),1);
colors = [0]*len(t);
for i in range(len(t)):
  colors[i] = cmap(cindex[i])
```

```
320
guess = [25,0];
for m in range(len(t)):

  # Defining and preallocating vectors ------------------------
  v = [0]*size;     # Defining the voltage array for points
325  i = [0]*size;     # Preallocating current solution vector
  p = [0]*size;     # Preallocating power solution vectora
  eta = [0]*size;
  for a in range(len(g)):
330    def f(z): return FMpp(z,g[a],t[m]) # Defining function to be solved

    sol = fsolve(f,guess) # Solving
    v[a] = sol[0];
    i[a] = sol[1];
335    p[a] = sol[0]*sol[1];

    IL = g[a]/GSrc*( IrrSrc + mu*(t[m] - ThetaSrc))
    eta[a] = p[a]/(Np*IL*(v[a] + Rs*i[a]))
    guess = sol;  # Updating next current guess
340
  ax8.plot(g,eta, linewidth = 1.5, color = colors[m], label = format(t[m] - 273.15,'.0
      f'));
# ------------------------------------------------------------------
# PROCESSING FIGURES
# ------------------------------------------------------------------
345
pyplot.rc('text', usetex=True)
pyplot.rc('font', family='serif')

# Processing figure 1 ------------------------------------------------
350 ax1.set_xlabel('Voltage (V)', fontsize = 16)
ax1.set_ylabel('Current (A)', fontsize = 16)
ax1.set_title(r'Panel current versus voltage at $\theta = 25^{\circ}$C', fontsize =
    16)
legend = ax1.legend(loc = 'upper right', shadow = True, title = 'Irradiation (W/m$
    ^{-2}$)', fontsize = 12)
ax1 = prettyAxis(ax1)
355
# Processing figure 2 ------------------------------------------------
ax2.set_xlabel('Voltage (V)', fontsize = 16)
ax2.set_ylabel('Power (W)', fontsize = 16)
ax2.set_title(r'Panel power versus voltage at $\theta = 25^{\circ}$C', fontsize = 16)
360 legend2 = ax2.legend(loc = 'upper left', shadow = True, title = 'Irradiation (W/m$
    ^{-2}$)', fontsize = 12)
ax2 = prettyAxis(ax2)

# Processing figure 3 ------------------------------------------------
ax3.set_xlabel('Voltage (V)', fontsize = 16)
365 ax3.set_ylabel('Current (A)', fontsize = 16)
ax3.set_title(r'Panel current versus voltage at $\phi = 1000$ W.m$^{-2}$', fontsize =
    16)
legend3 = ax3.legend(loc = 'upper left', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax3 = prettyAxis(ax3)

370 # Processing figure 4 ------------------------------------------------
ax4.set_xlabel('Voltage (V)', fontsize = 16)
ax4.set_ylabel('Power (W)', fontsize = 16)
```

```
ax4.set_title('Panel power versus voltage at $\phi = 1000$ W.m$^{-2}$', fontsize = 16)
legend4 = ax4.legend(loc = 'upper left', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax4 = prettyAxis(ax4)

# Processing figure 5 ------------------------------------------------
ax5.set_xlabel('Voltage (V)', fontsize = 16)
ax5.set_ylabel('Current (A)', fontsize = 16)
ax5.set_title('Panel MPP curves', fontsize = 16)
legend5 = ax5.legend(loc = 'upper left', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax5 = prettyAxis(ax5)

# Processing figure 6 ------------------------------------------------
ax6.set_xlabel('Voltage (V)', fontsize = 16)
ax6.set_ylabel('Power (W)', fontsize = 16)
ax6.set_title('Panel MPP curves', fontsize = 16)
legend6 = ax6.legend(loc = 'upper left', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax6 = prettyAxis(ax6)

# Processing figure 7 ------------------------------------------------
ax7.set_xlabel(r'Irradiance $\phi$ (W.m$^{-2}$)', fontsize = 16)
ax7.set_ylabel(r'Open-circuit voltage $V_{OC}$ (V)', fontsize = 16)
ax7.set_title('Panel open-circuit voltage versus irradiance, parametrized by
    temperature', fontsize = 16)
legend7 = ax7.legend(loc = 'lower right', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax7 = prettyAxis(ax7)

# Processing figure 8 ------------------------------------------------
ax8.set_xlabel(r'Irradiance $\phi$ (W.m$^{-2}$)', fontsize = 16)
ax8.set_ylabel(r'Panel efficiency $\eta$', fontsize = 16)
ax8.set_title('Panel efficiency versus irradiance, parametrized by temperature',
    fontsize = 16)
legend8 = ax8.legend(loc = 'lower right', shadow = True, title = r'Temperature ($^{\
    circ}$C)', fontsize = 12)
ax8 = prettyAxis(ax8)

# --------------------------------------------------------------------
# SHOW FIGURES
# --------------------------------------------------------------------
pyplot.show();
```

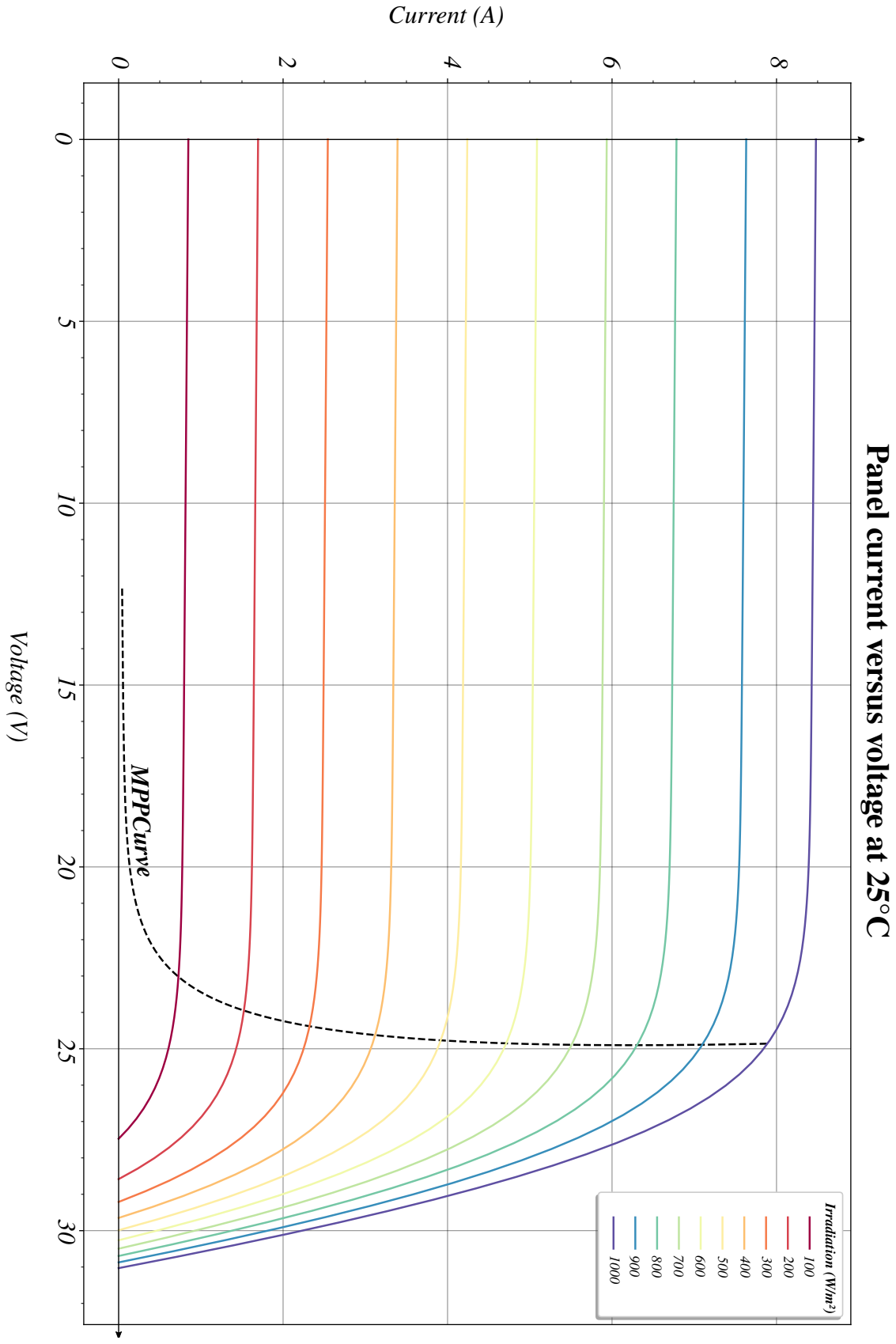

Figure 7. Panel transconductance (continuous) and MPP (dashed) curves with fixed temperature θ_{SRC} and varying irradiance.

APPENDIX B

Second appendix

This is the second appendix.