Preparing Figures in Matlab and LaTeX for Quality Publications

Azad Ghaffari

Cymer Center for Control Systems and Dynamics - UC San Diego

Second Edition, January 2014

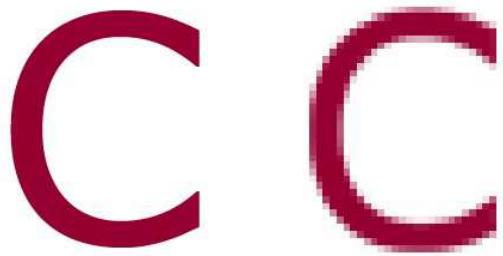
Image formats: Vector vs. Raster

Raster graphics or bitmap

- Made up of individual pixels, resolution dependent
- Resizing reduces quality
- Minimal support for transparency
- Conversion to vector is difficult
- File types: .jpg, .gif, .tif, and .bmp

Vector graphics or line art

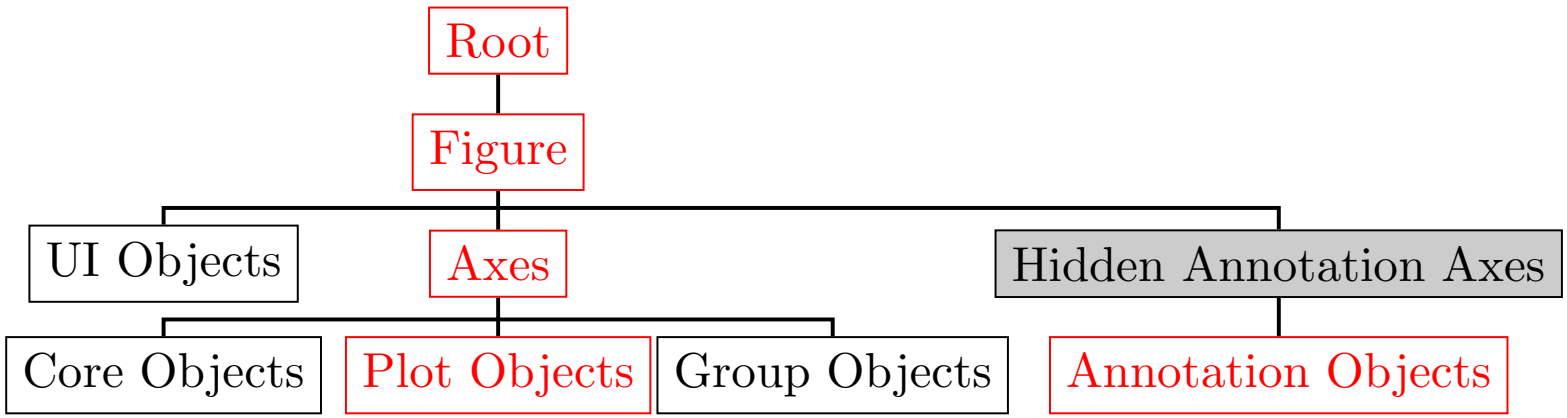
- Created mathematically w/o the use of pixels
- High resolution
- Scalable to any size w/o pixelation or quality loss
- Conversion to raster is easy
- File types: .eps, .pdf, .ai, and .dxf



Vector Raster

Figures in Matlab

- Handle Graphics is an object-oriented structure for creating, manipulating and displaying graphics
- Graphics objects: basic drawing elements used in Matlab to display graphs and GUI components
- Every graphics object:
  - Unique identifier, called a handle
  - Set of characteristics, called properties
- Possible to modify every single property using the command-line
- Objects organized into a hierarchy



Avoid common mistakes

Don't

- Use graphical commands with their default setting
- Export figures using the "export" menu function
- Modify figure properties using the mouse
- Use third party graphics editors where possible

Do

- Use functions and scripts to generate plots: **Reuseability**
- Specify figure properties: **Modifability**
- Generate your figures using `print` command: **Controllability**

`plot` function

Calling the plot function creates graphics objects:

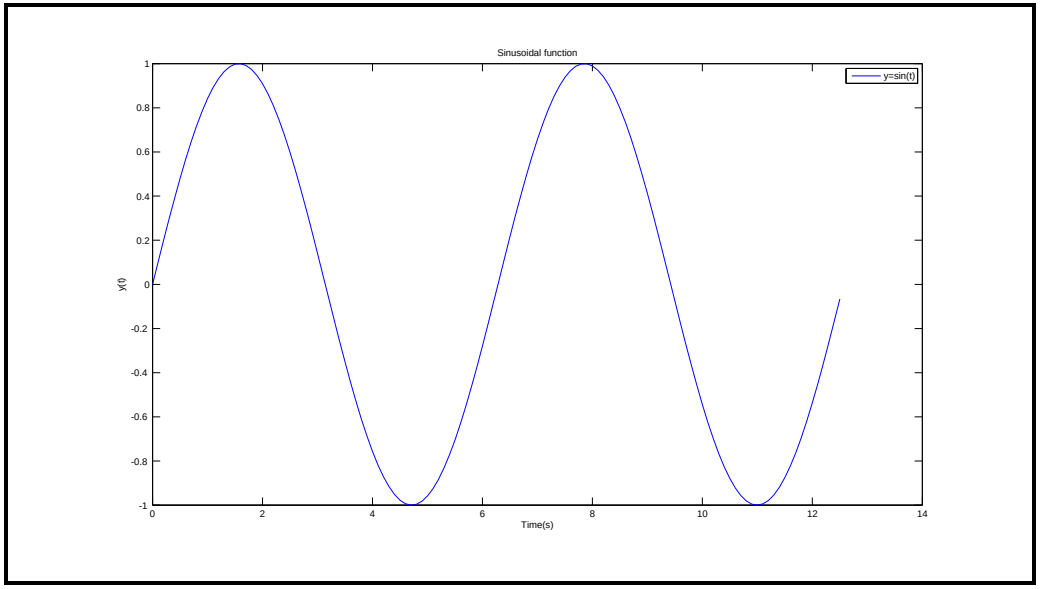
Figures: Windows that contain axes toolbars, menus, etc.
Axes: Frames that contain graphs
Lineseries plot objects: Representations of data passed to the plot function
Text: Labels for axes tick marks, optional titles and annotations

Main functions for working with objects

`gcf` Handle of the current figure
`gca` Handle of the current axis in the current figure
`get` Query the values of an object's properties
`set` Set the values of an object's properties
`delet` Delete an object
`copyobj` Copy graphics object

Example

```
t = 0:.1:4*pi;
y = sin(t);
plot(t,y)
xlabel('Time(s)')
ylabel('y(t)')
title('Sin function')
legend('y=sin(t)')
```



- Save the plot as .eps
- Use LaTeX command
  `\includegraphics[width=2.5in]{sin1}`

Problems:

- Huge difference between font size of the text and figure
- Axes are not proportional
- Figure is not informative to the audience!

Figure size

What is the size of your presentation?

For a beamer slide: width=5.04 in, length=3.78 in

What is the desired figure size?

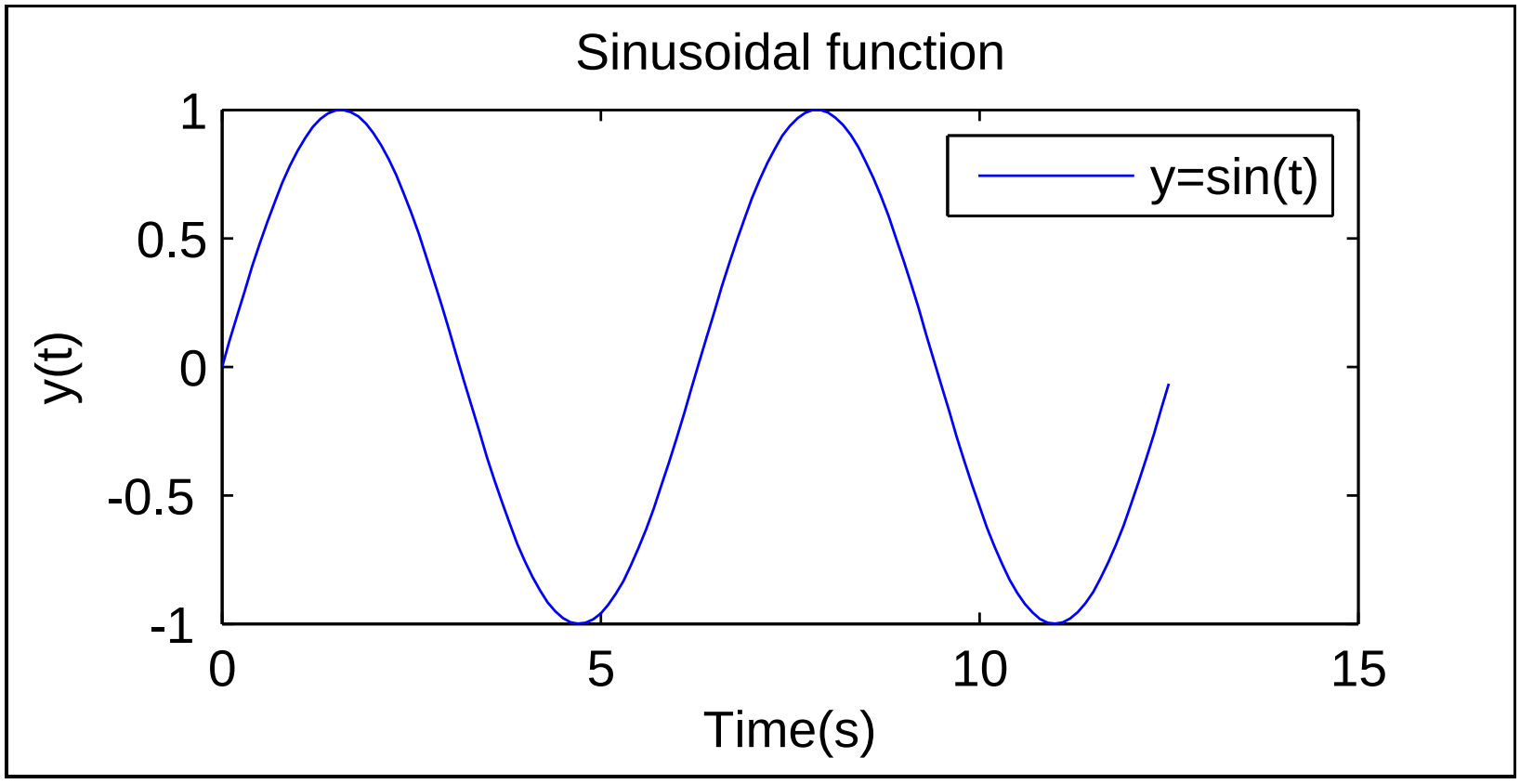
Figure `width=4in`, figure `height=2in`

Run `figure` command before drawing the plot

```
figure('Units','inches',...
       'Position',[x0 y0 width height],...
       'PaperPositionMode','auto');
```

`(x0,y0)` = position of the lower left side of the figure

Figure size



- Dimensions are corrected
- Correction needed:
  - Font size and type
  - Axes limits
  - Legend and labels to appear in LaTeX format

Axes settings

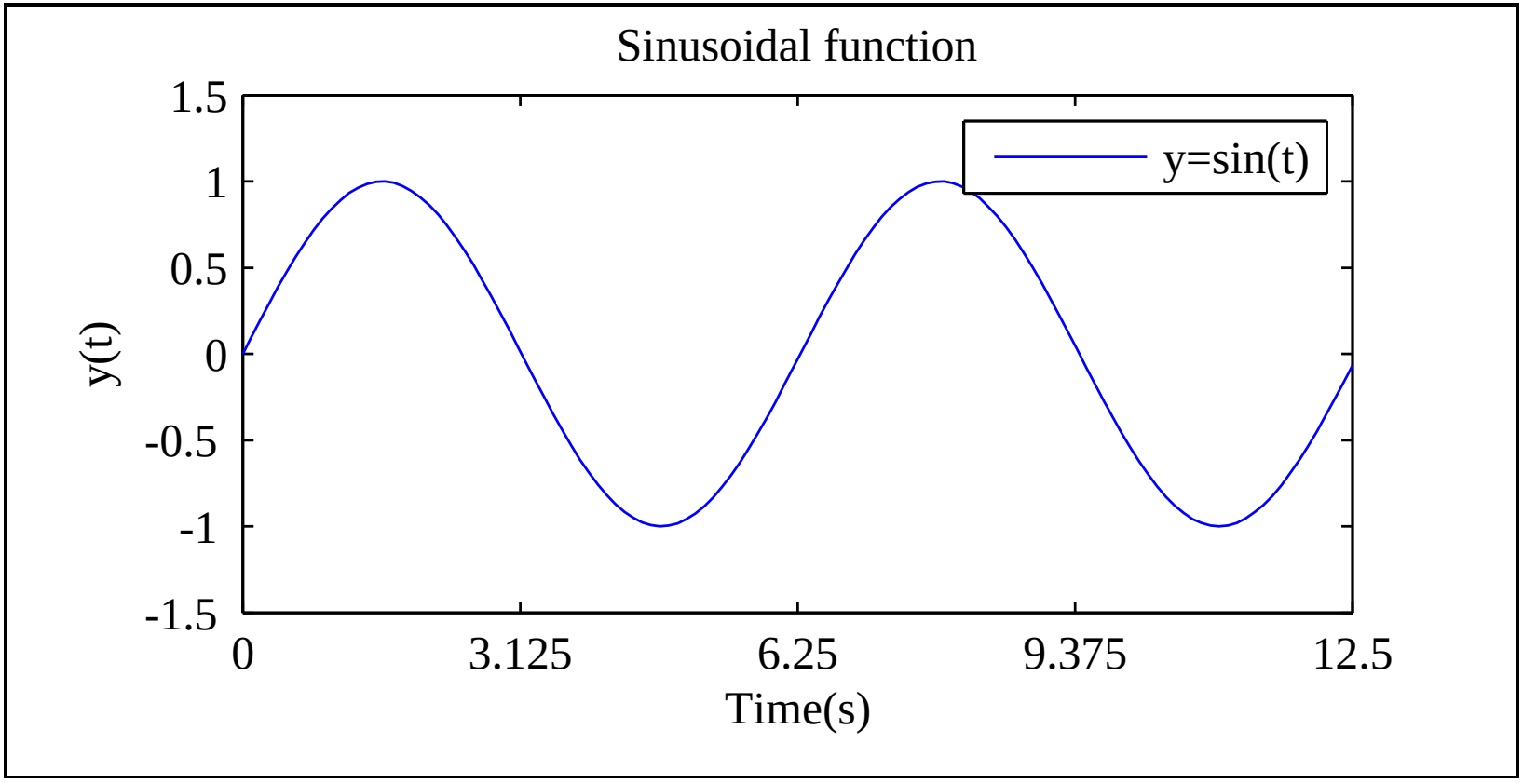
Commands right after running `plot`

```
axis([0 t(end) -1.5 1.5])
set(gca,...
    'Units','normalized',...
    'YTick',-1.5:.5:1.5,...
    'XTick',0:t(end)/4:t(end),...
    'Position',[.15 .2 .75 .7],...
    'FontUnits','points',...
    'FontWeight','normal',...
    'FontSize',9,...
    'FontName','Times')
```

Figure is exported to .eps

Axes setting

Axes position, limits, font, and ticks locations are corrected



Labels and legend

LaTeX typesetting by setting `interpreter` to `latex`

Labels can have different font sizes

```
ylabel({'$y(t)$'},...
    'FontUnits','points',...
    'interpreter','latex',...
    'FontSize',9,...
    'FontName','Times')
xlabel('Time(s)',...
    'FontUnits','points',...
    'FontWeight','normal',...
    'FontSize',7,...
    'FontName','Times')
```

Labels, legend, and LaTeX commands

```
legend({'$y=\sin(t)$'},...
    'FontUnits','points',...
    'interpreter','latex',...
    'FontSize',7,...
    'FontName','Times',...
    'Location','NorthEast')
title('Sinusoidal function',...
    'FontUnits','points',...
    'FontWeight','normal',...
    'FontSize',7,...
    'FontName','Times')
```

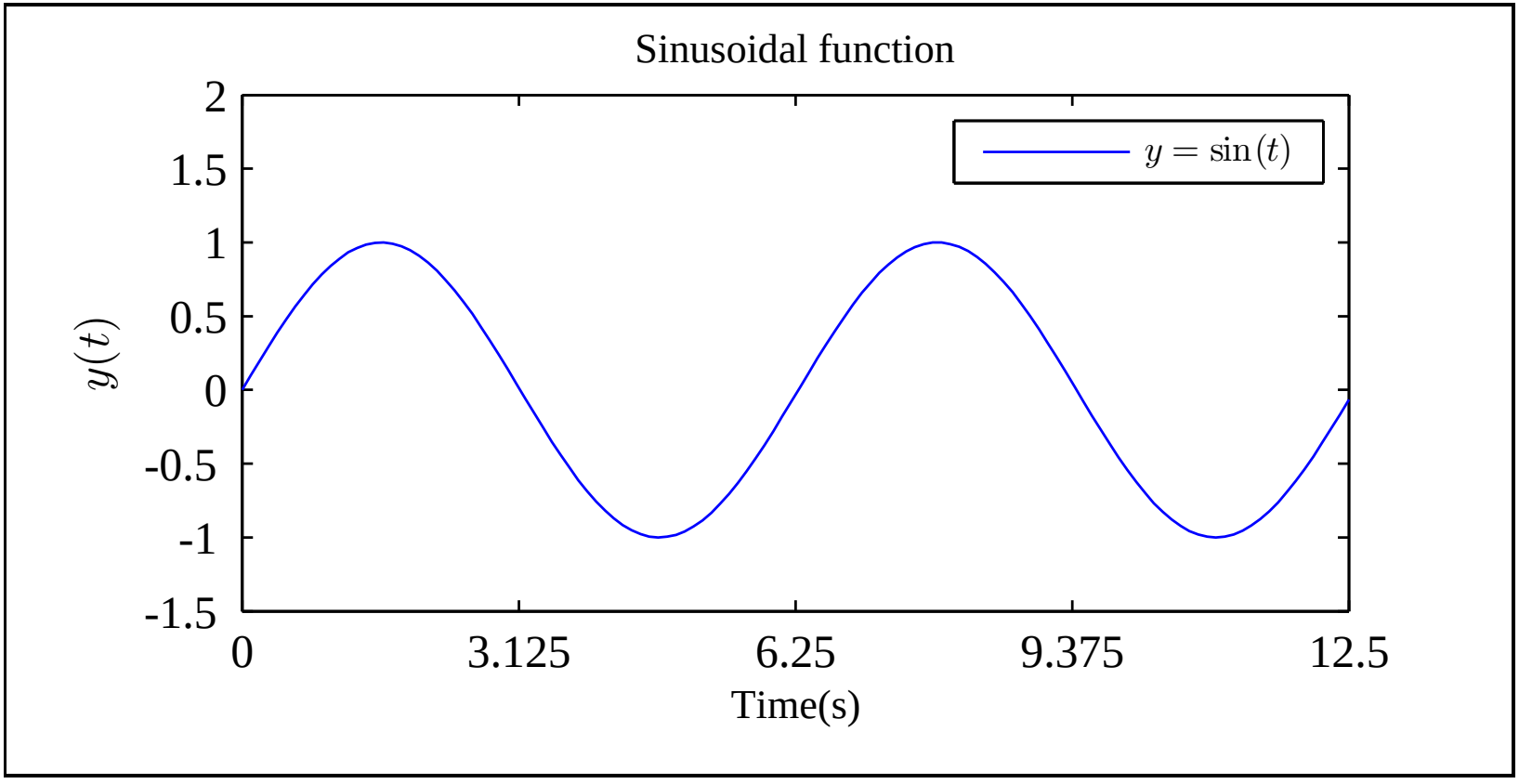
The figure is exported to .eps

Labels and legend

Mathematical writing is corrected

Figure has large white boundaries

Fonts are not proportional to the values we want



How to save the plot

Don't export the plot to .eps

Use `print` command to generate .eps files

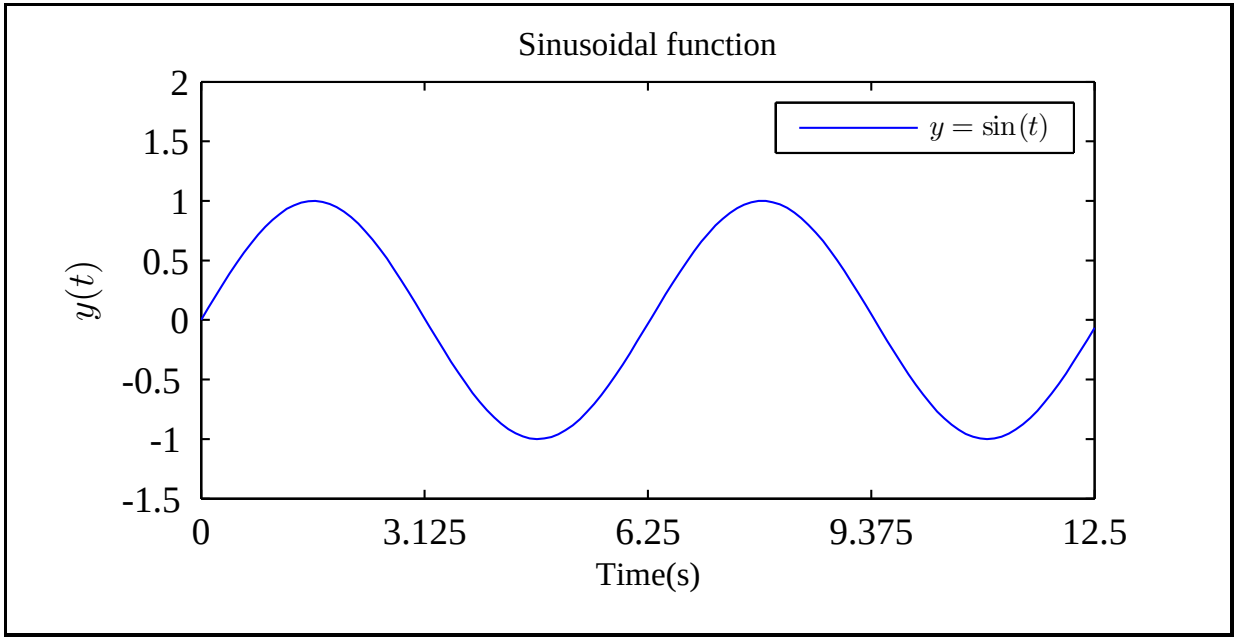
```
print -depsc2 myplot.eps
```

Main vector formats

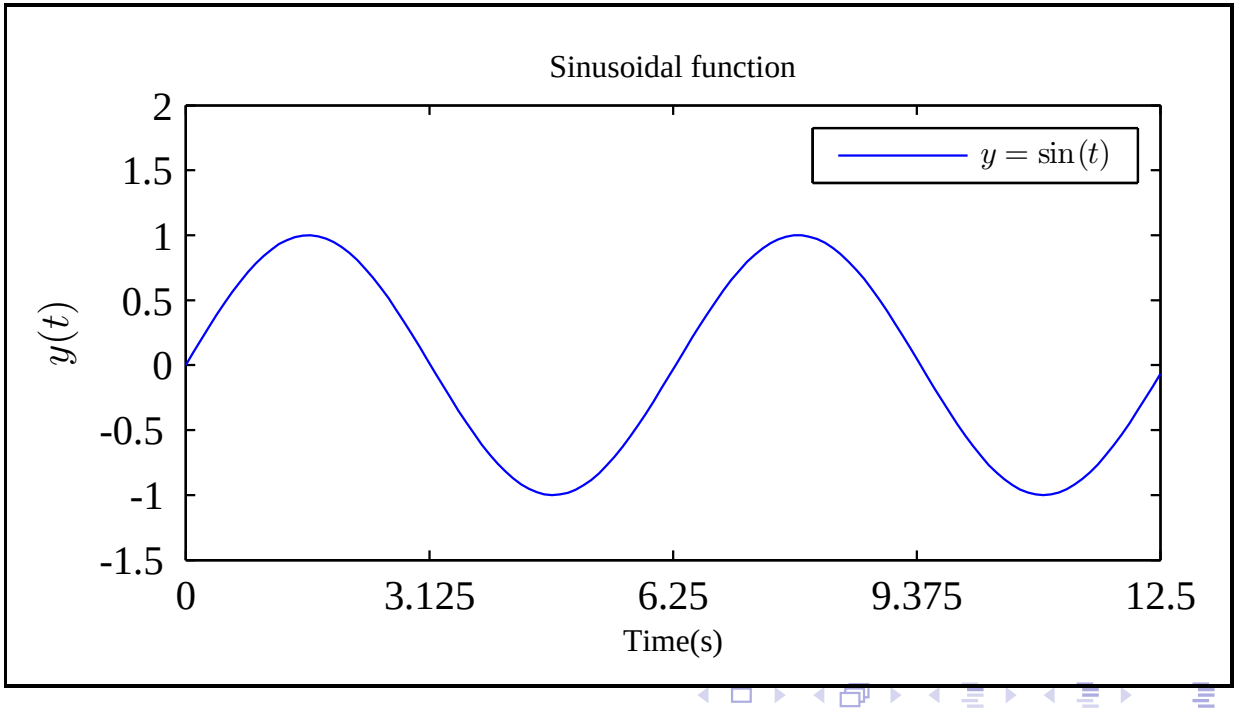
`-deps`	.eps black and white
`-depsc`	.eps color
`-deps2`	.eps level 2 black and white
`-depsc2`	.eps level 2 color
`-dpdf`	.pdf color file format

Exported .eps vs. printed .eps

Exported .eps

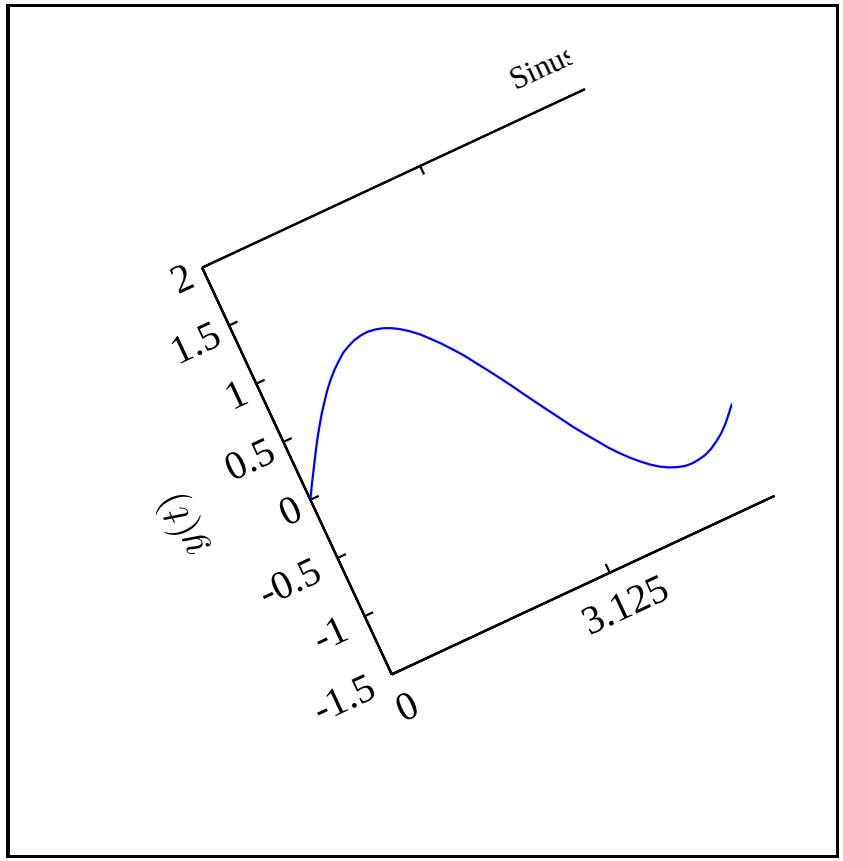


Printed .eps

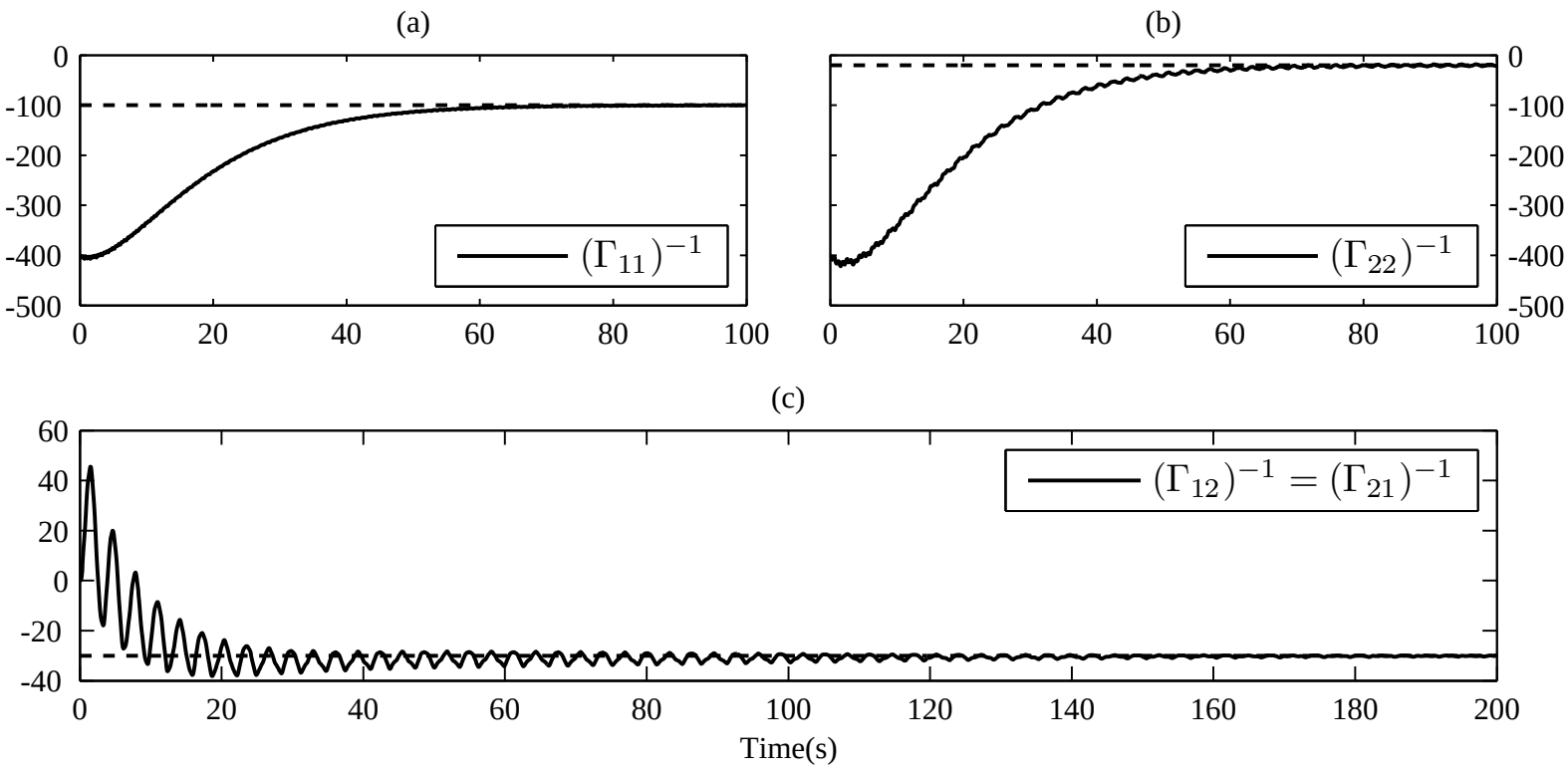


Inserting .eps in LaTeX

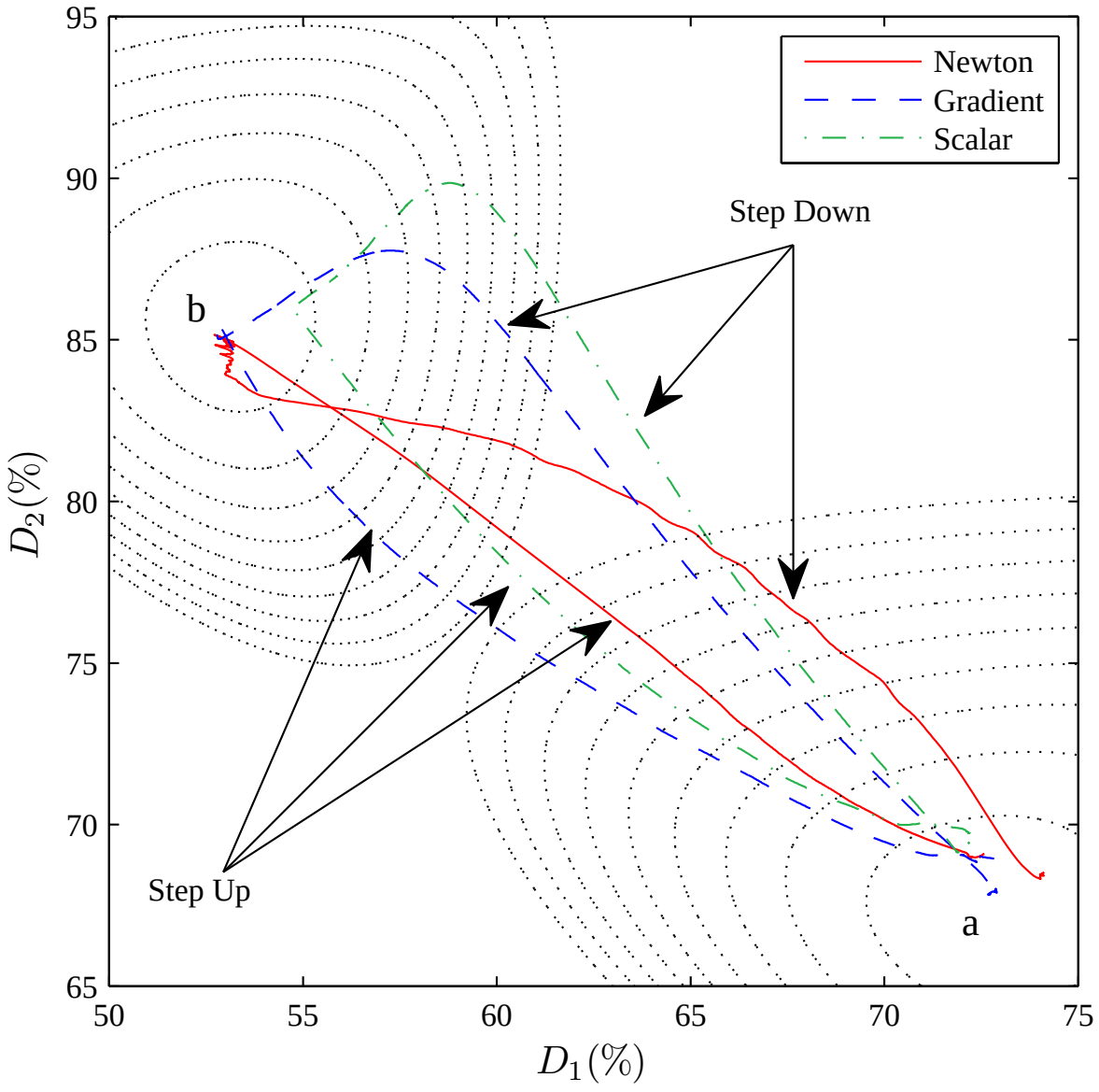
`\includegraphics[options]{myplot}` is useful to change the look of the .eps file



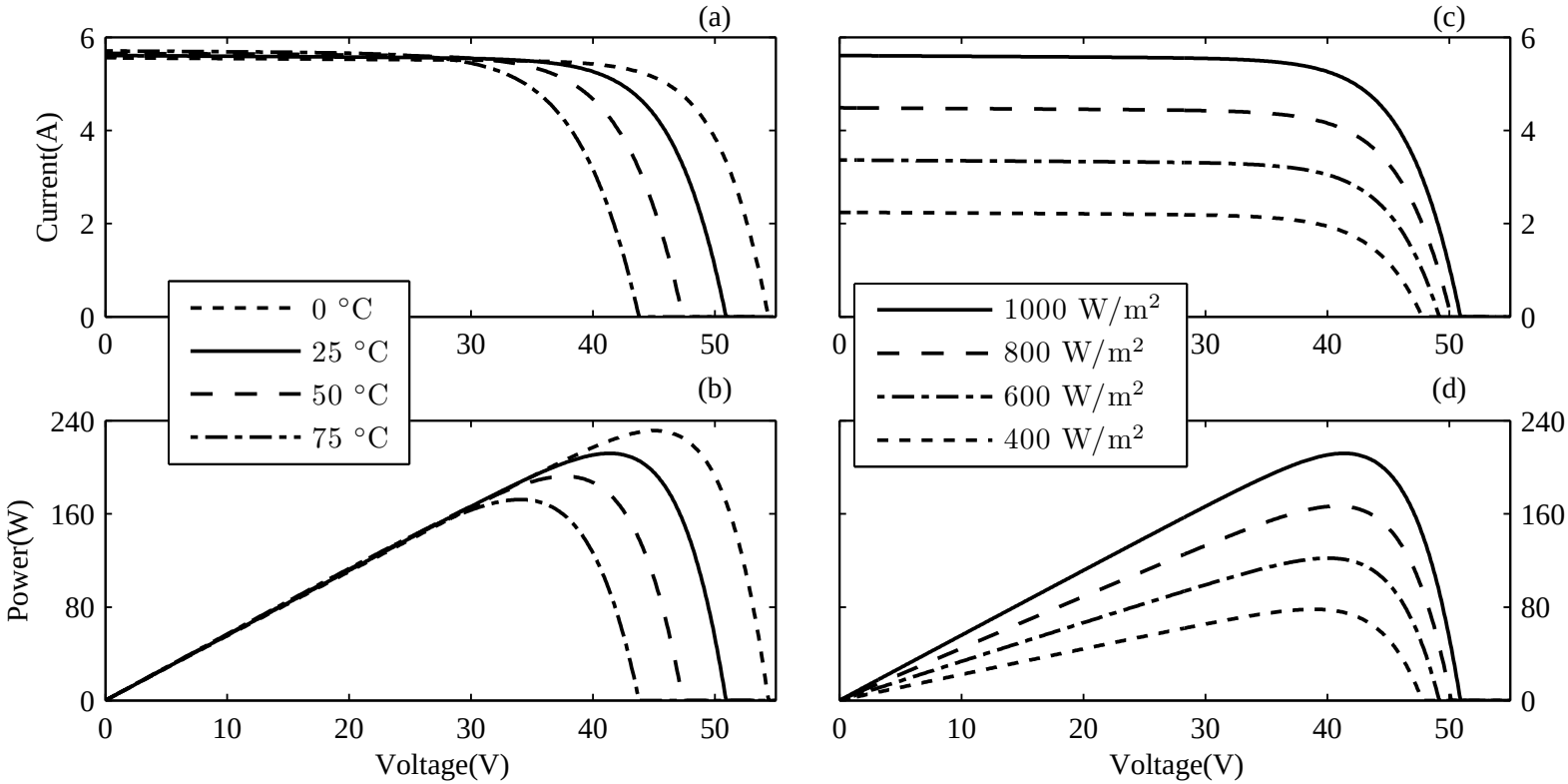
Ex. 1



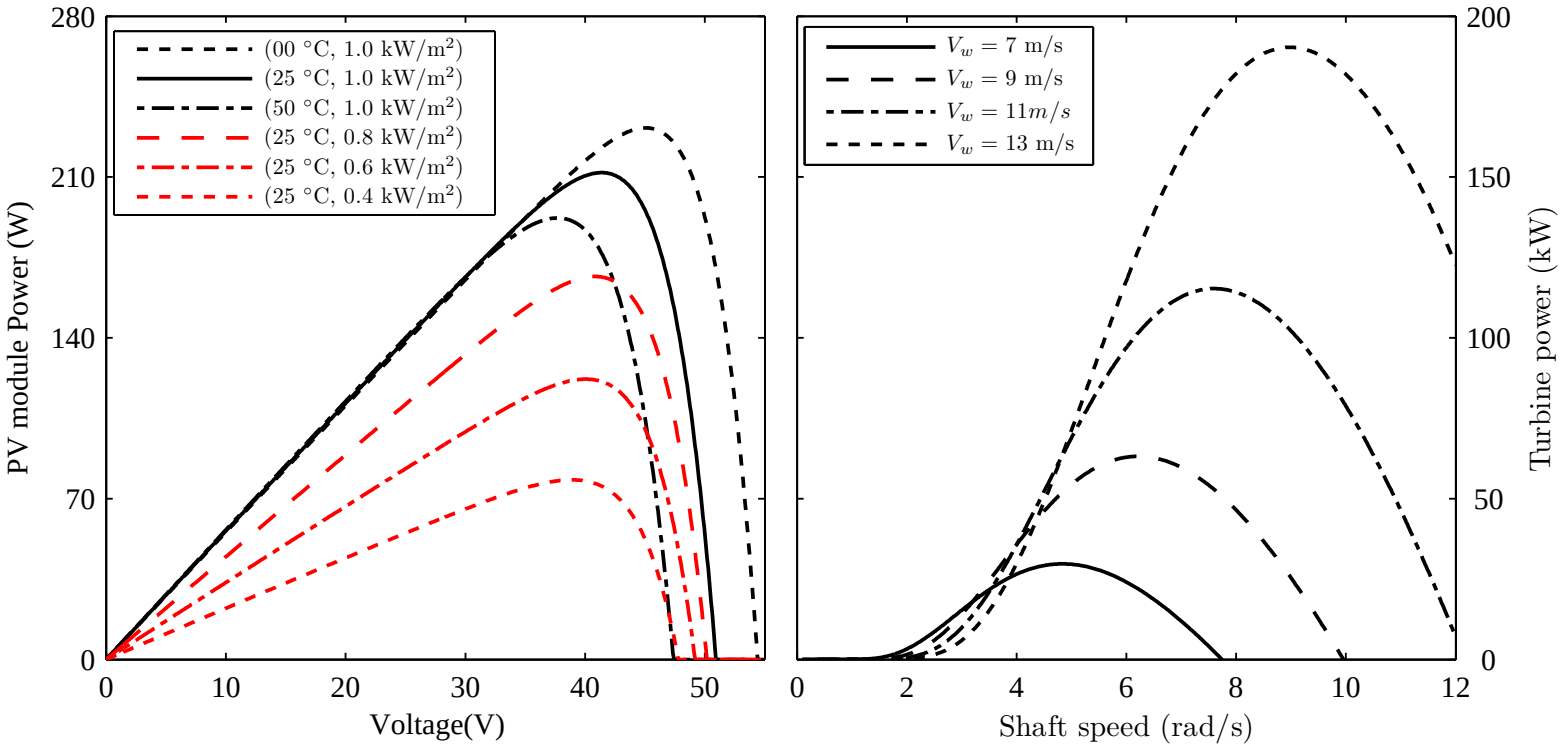
Ex. 2


Newton
Gradient
Scalar
Step Down
Step Up
b
a
95
90
85
80
75
70
65
50
55
60
65
70
75
$D_2(\%)$
$D_1(\%)$

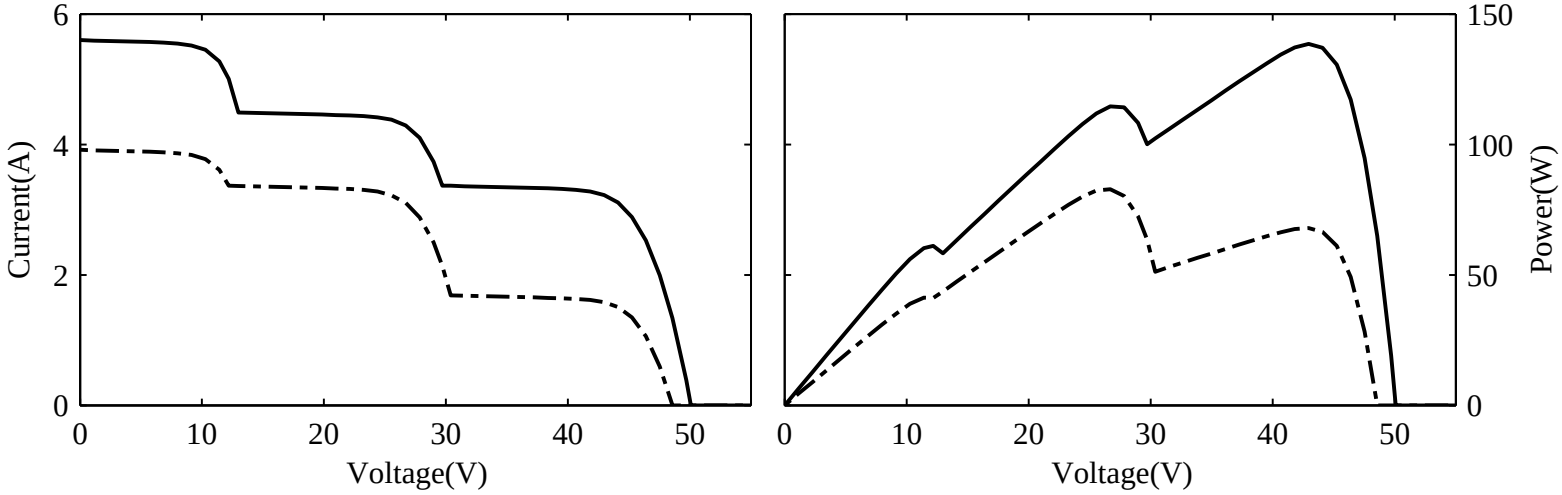
Ex. 3


(a)
(b)
(c)
(d)
Current(A)
Power(W)
Voltage(V)
Voltage(V)
0 °C
25 °C
50 °C
75 °C
1000 W/m²
800 W/m²
600 W/m²
400 W/m²

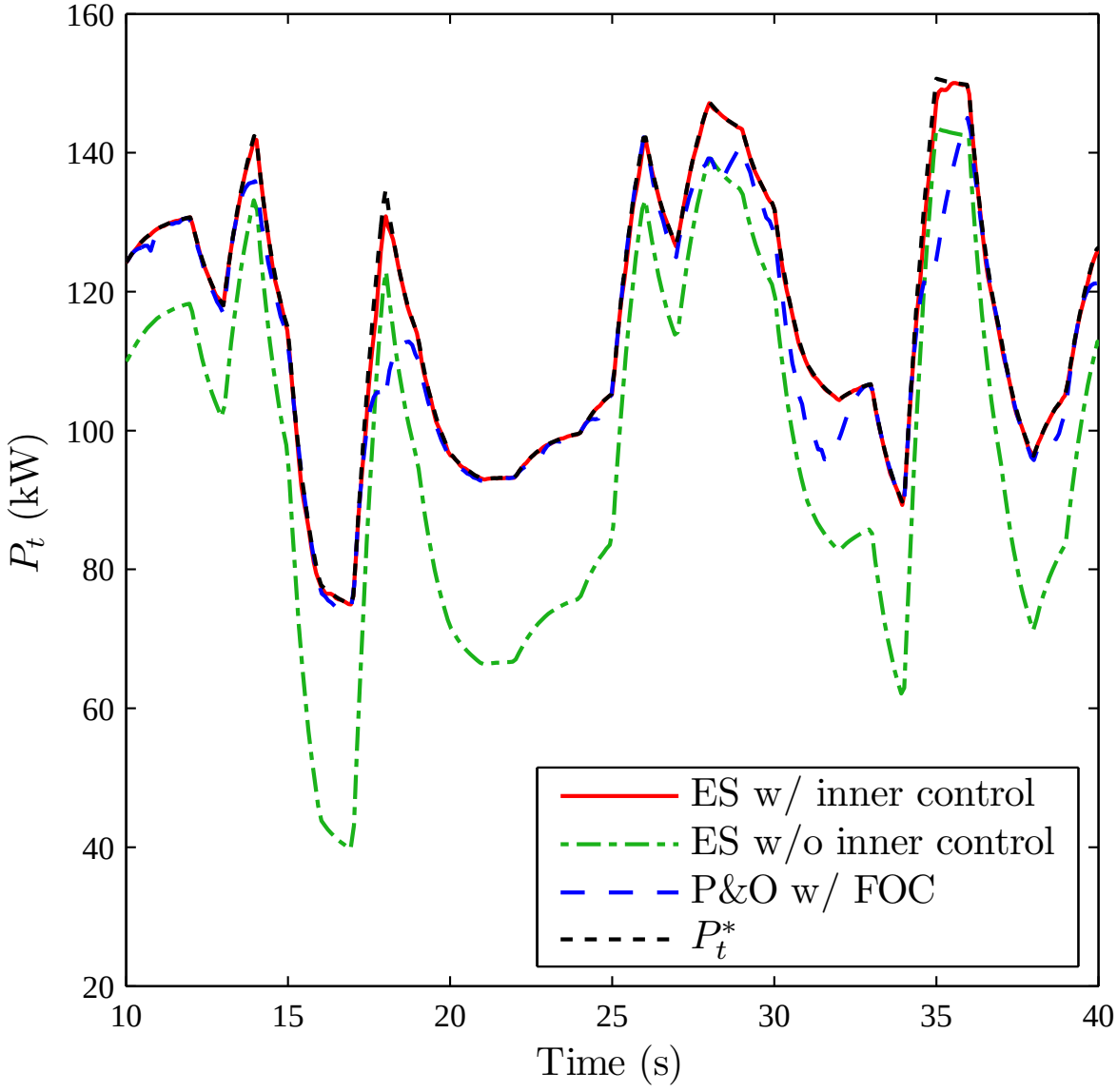
Ex. 4


(00 °C, 1.0 kW/m²)
(25 °C, 1.0 kW/m²)
(50 °C, 1.0 kW/m²)
(25 °C, 0.8 kW/m²)
(25 °C, 0.6 kW/m²)
(25 °C, 0.4 kW/m²)
PV module Power (W)
Voltage(V)
0
70
140
210
280
0
10
20
30
40
50
V_w = 7 m/s
V_w = 9 m/s
V_w = 11m/s
V_w = 13 m/s
Turbine power (kW)
Shaft speed (rad/s)
0
50
100
150
200
0
2
4
6
8
10
12

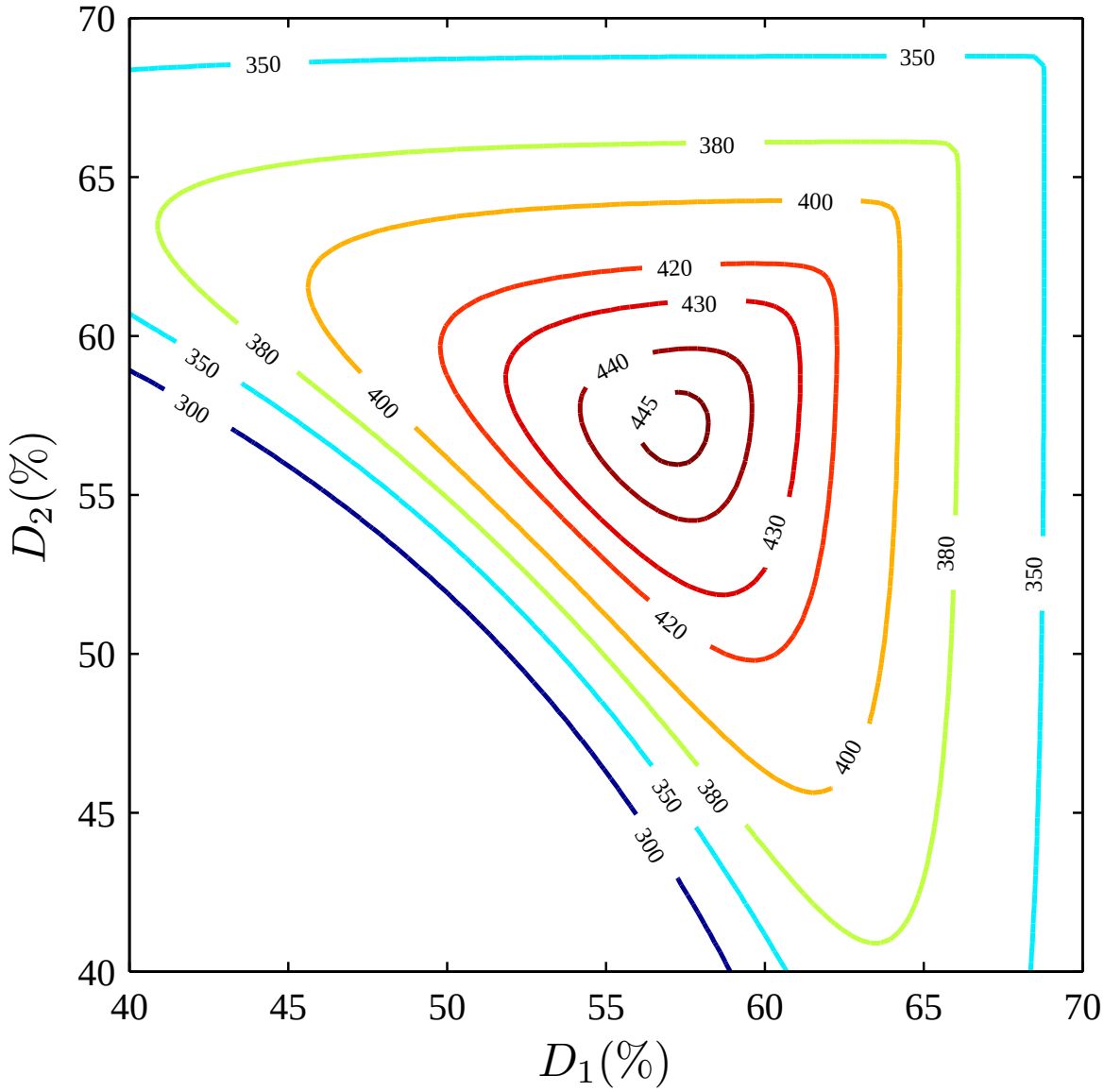
Ex. 5


Current(A)
Voltage(V)
Power(W)
Voltage(V)
0
2
4
6
0
10
20
30
40
50
0
50
100
150

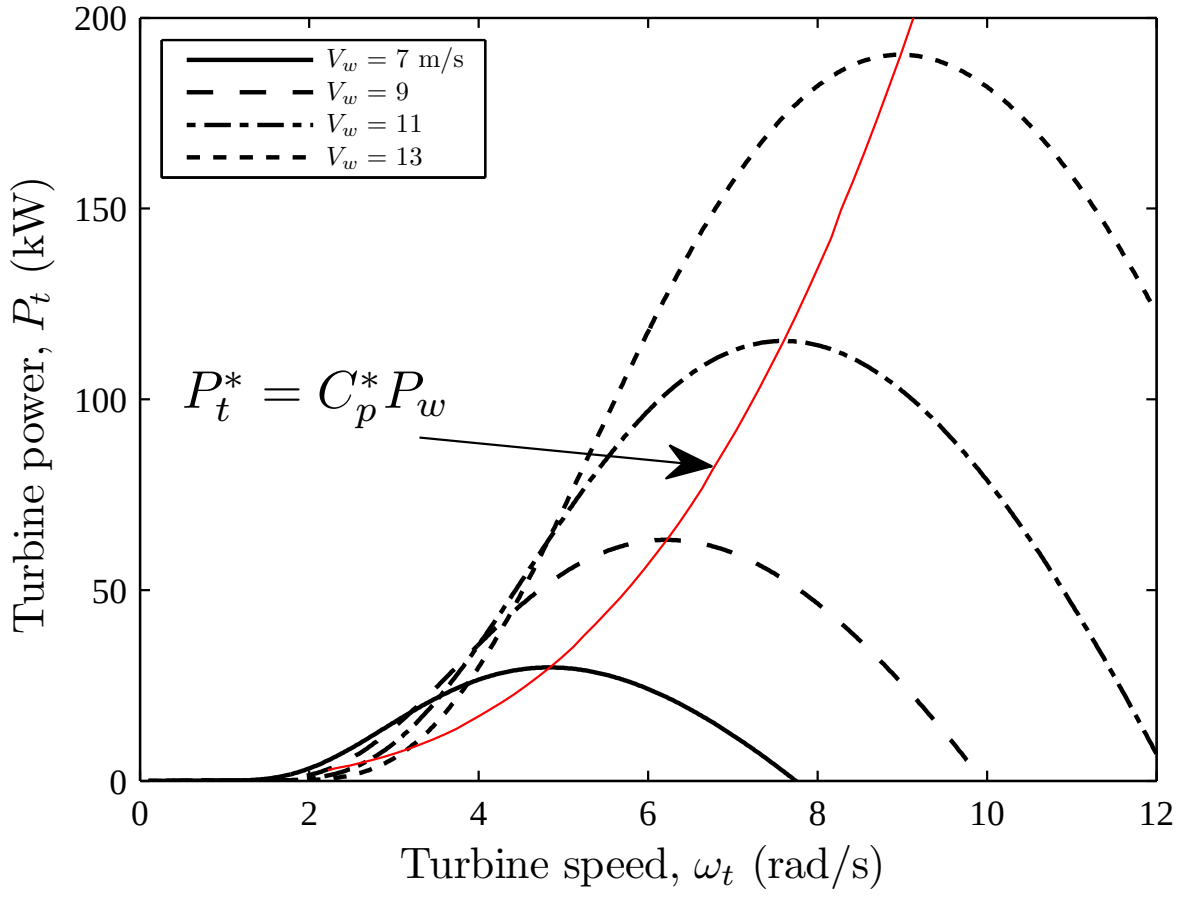
Ex. 6


160
140
120
100
80
60
40
20
10
15
20
25
30
35
40
P_t (kW)
Time (s)
ES w/ inner control
ES w/o inner control
P&O w/ FOC
P_t^*

Ex. 7


70
65
60
55
50
45
40
40
45
50
55
60
65
70
$D_2(\%)$
$D_1(\%)$
350
350
380
400
420
430
440
445
350
380
400
420
430
380
350
300
350
380
400
350
300

Ex. 8


$V_w = 7$ m/s
$V_w = 9$
$V_w = 11$
$V_w = 13$
$P_t^* = C_p^* P_w$
Turbine power, P_t (kW)
Turbine speed, ω_t (rad/s)
0
50
100
150
200
0
2
4
6
8
10
12

Export Simulink models (Not for publication)

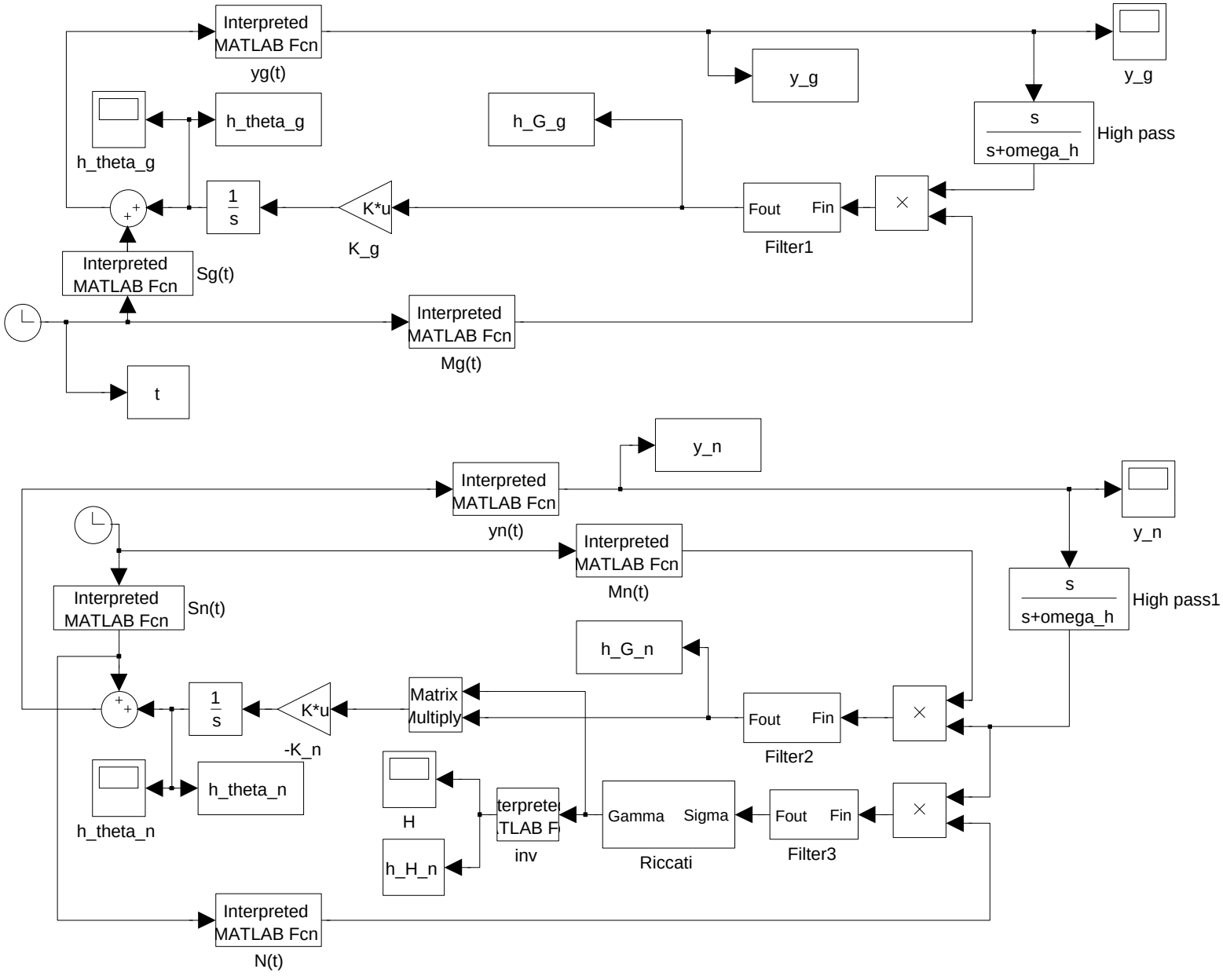
- Change the orientation to portrait, landscape, or tall
- Open the model

```
orient(gcs, 'portrait')
```

- print the model to an .eps file
- specify the name of your Simulink model using the switch -s

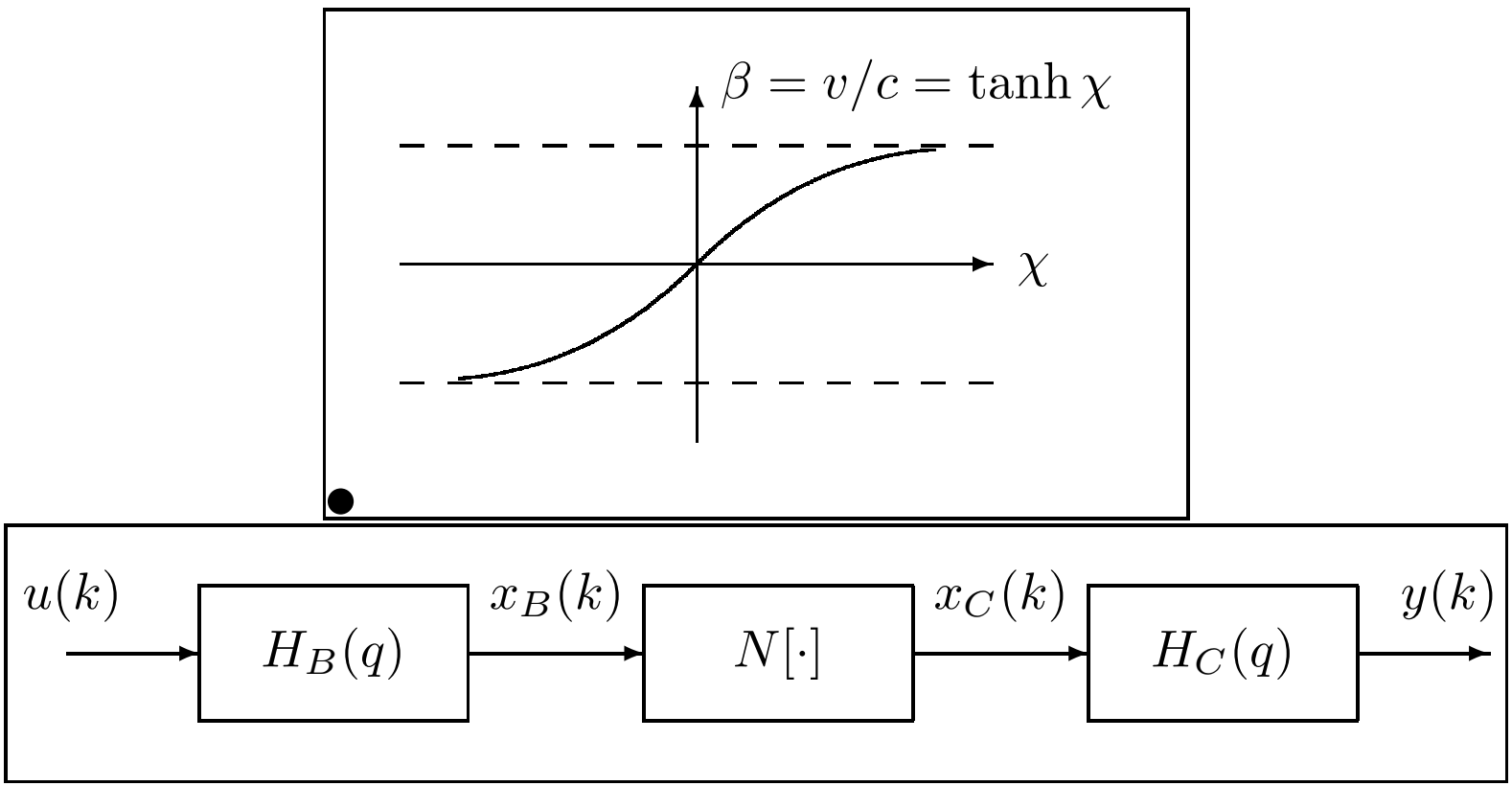
```
print -deps -r300 -s myfig.eps
```

Export Simulink models (Not for publication)



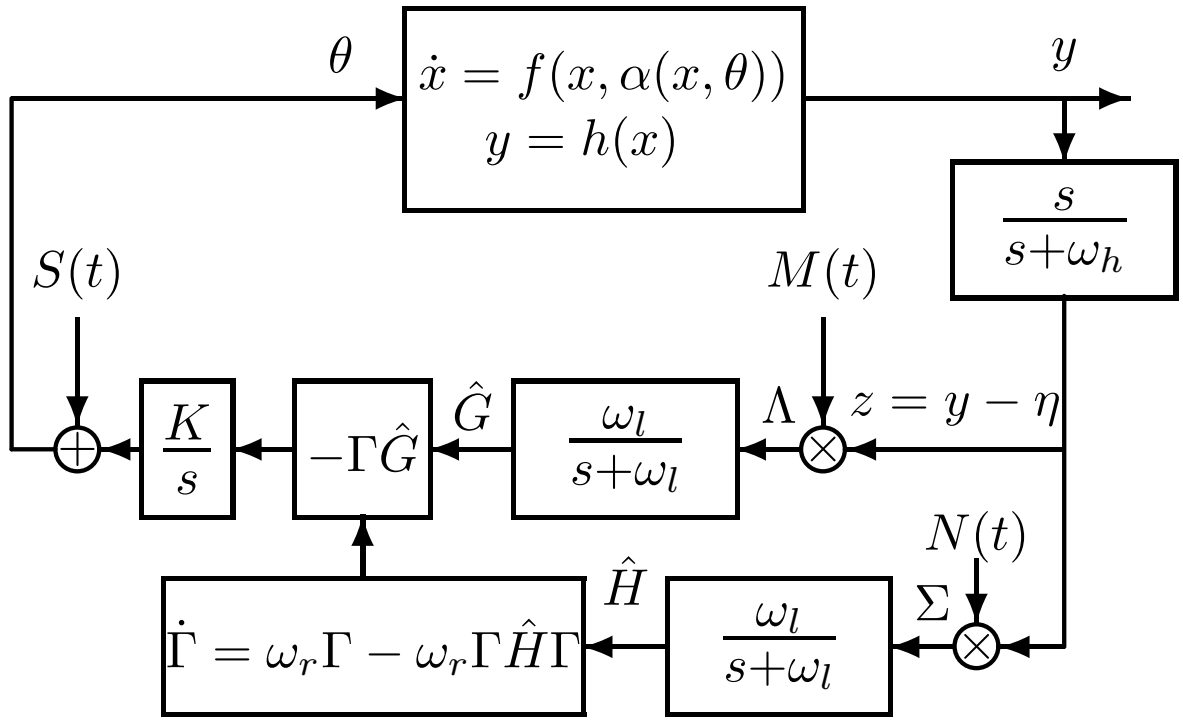
Diagrams in LaTeX– Picture environment

- For mathematical drawings
- Very limited options
- Time consuming

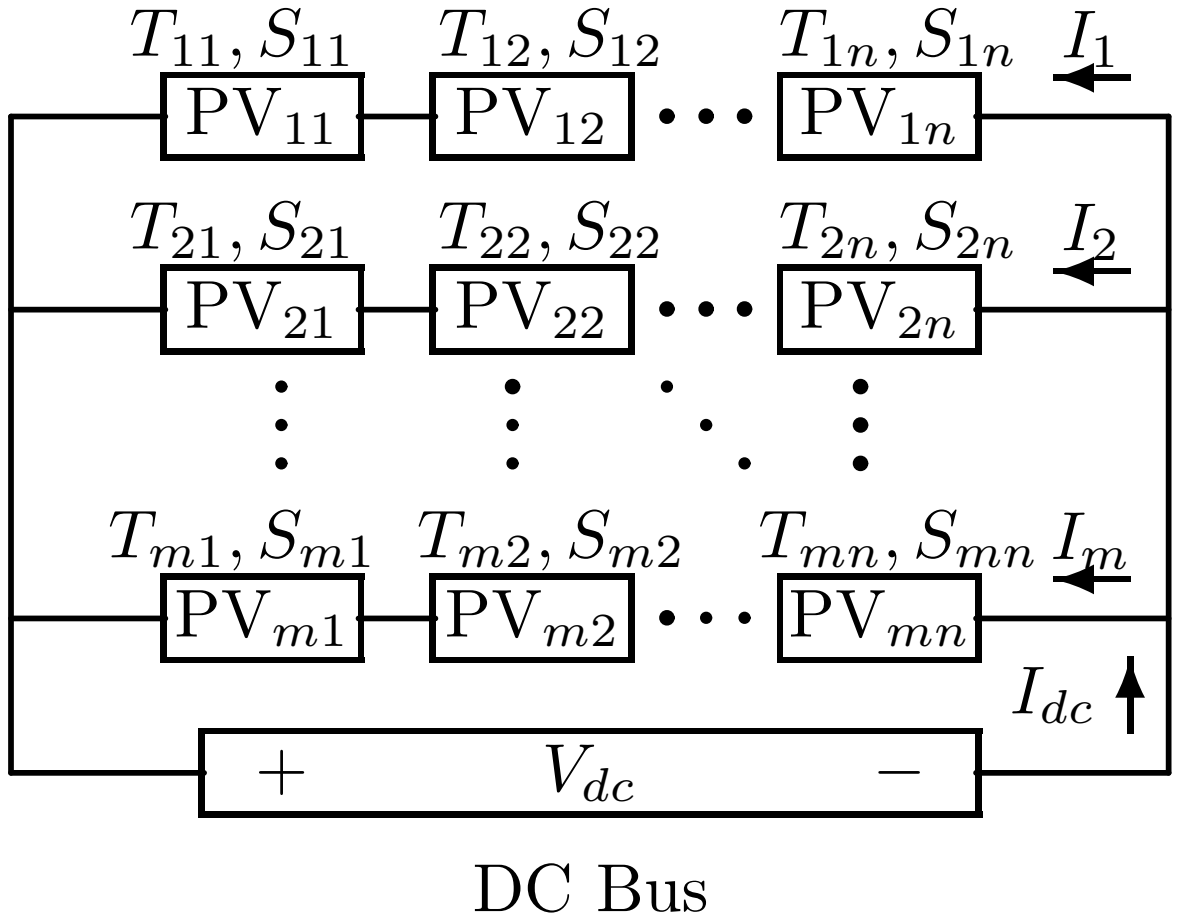


Diagrams in LaTeX– LaTeXCAD package

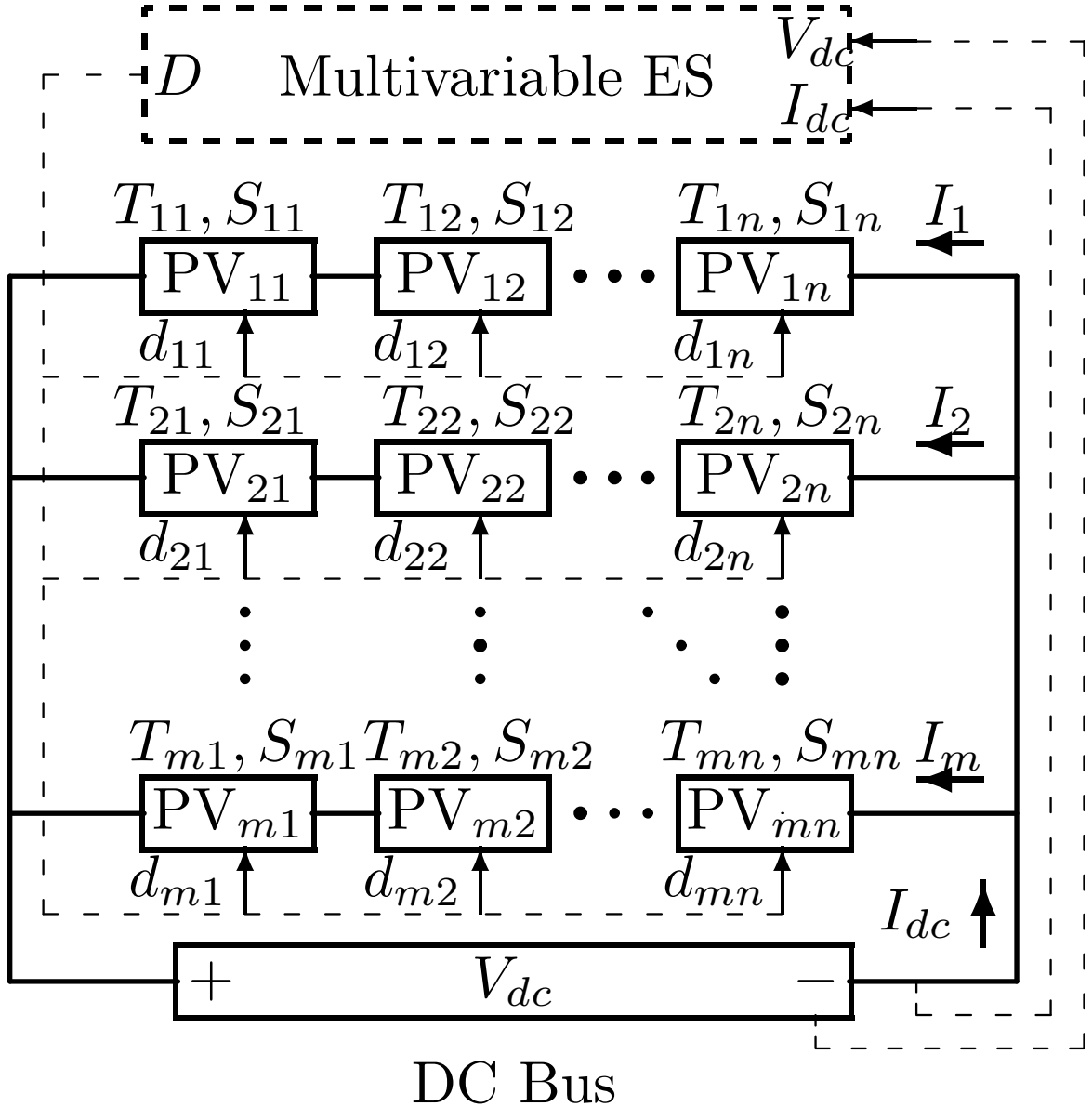
- Has a basic GUI
- Easy to use and very time saving
- Not precise, basic graphical elements with 3 different pen sizes
- Generates a LaTeX compatible .tex output



Diagrams in LaTeX– LaTeXCAD package

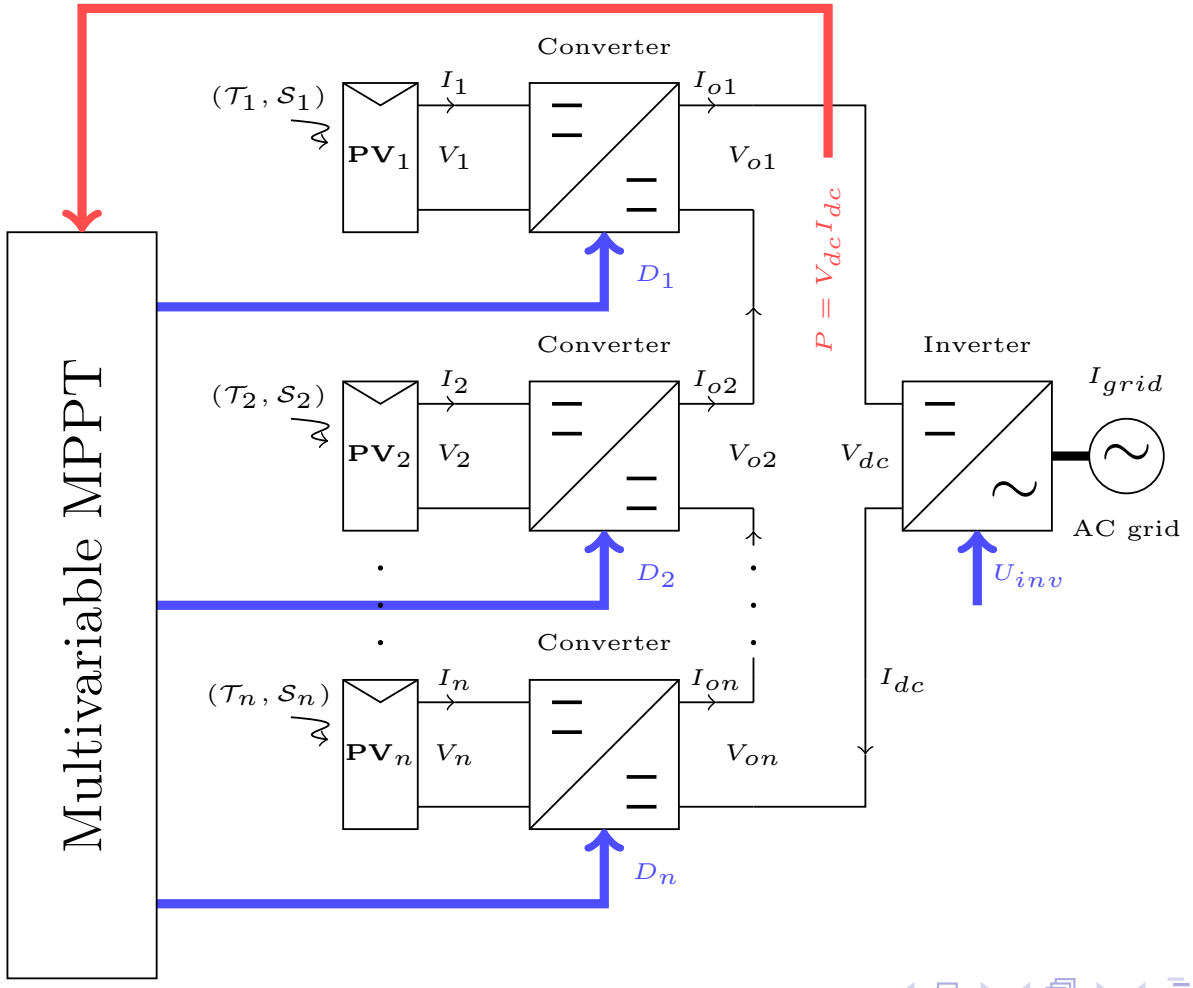


Diagrams in LaTeX– LaTeXCAD package

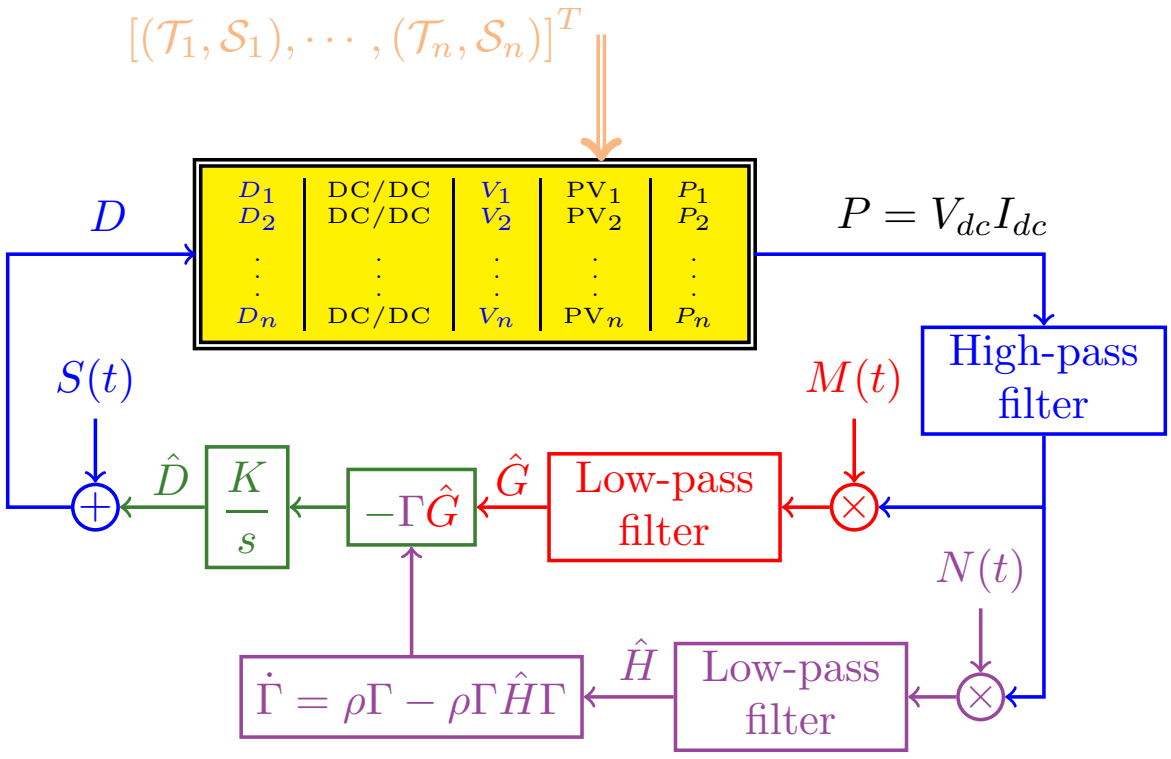


Diagrams in LaTeX– TikZ and PGF packages

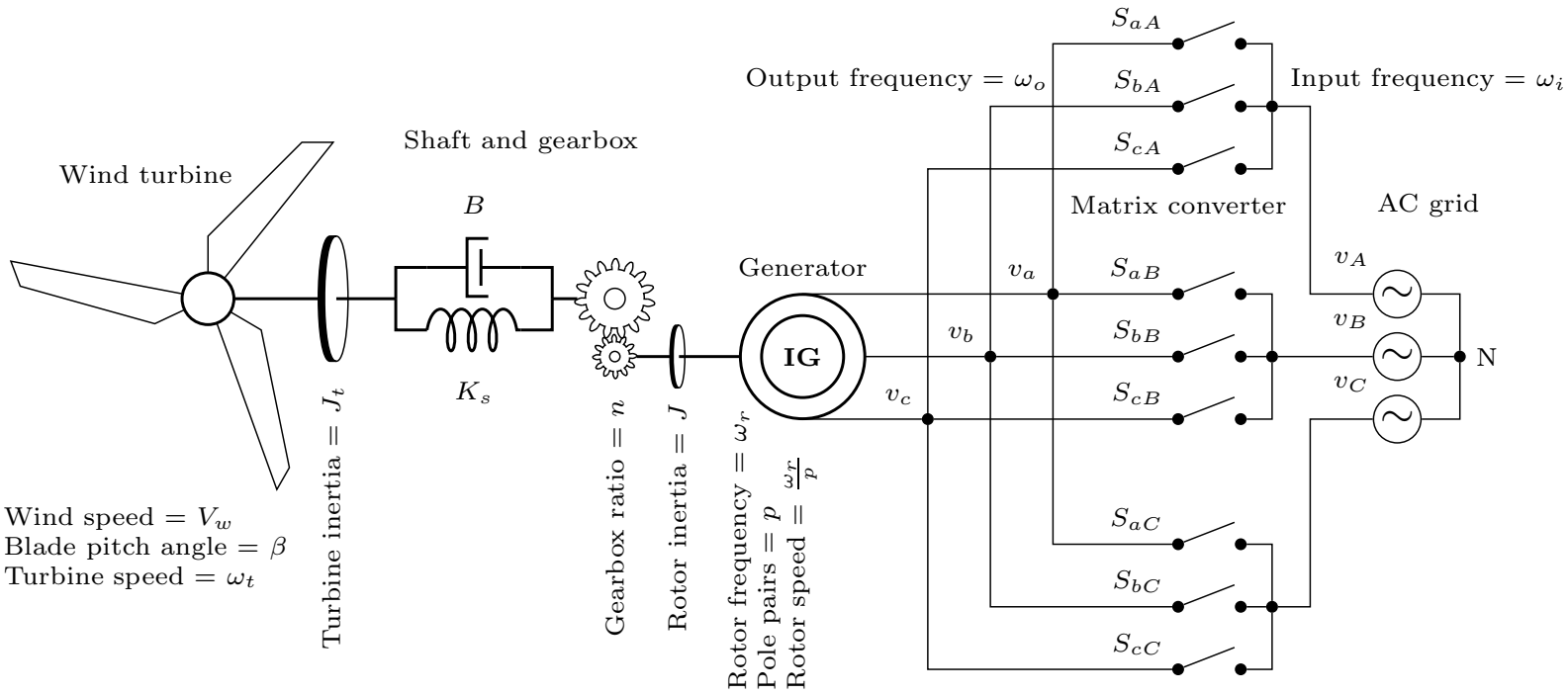
- Many options and tools
- Very sophisticated
- Cover many types of diagrams
- Other useful extensions based on Tikz and PGF



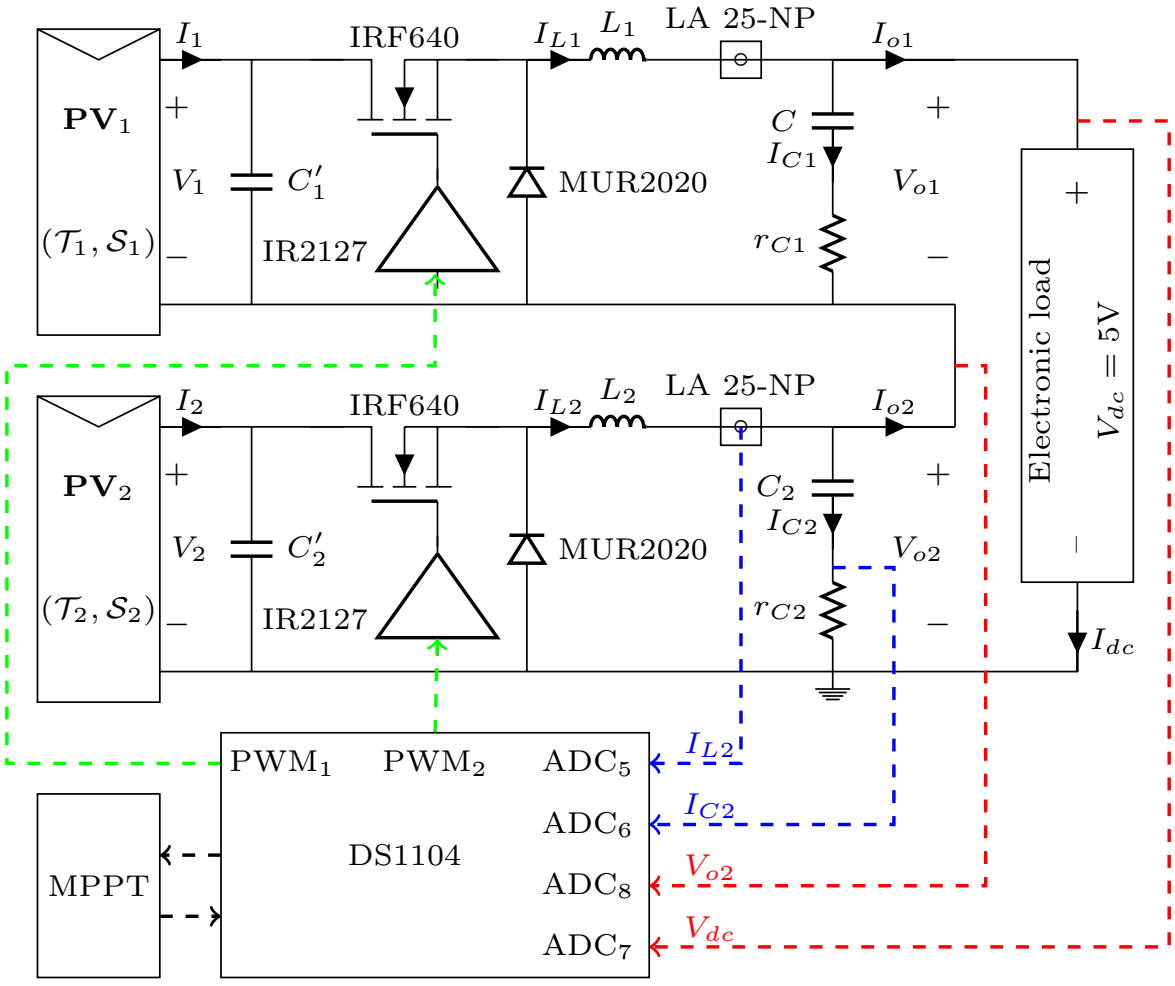
Diagrams in LaTeX– TikZ and PGF packages



Diagrams in LaTeX– TikZ and PGF packages



Electrical circuits in LaTeX– CrcuitTikZ package



How to convert LaTeX-produced figures into .eps

- Put figure in a separate LaTeX file
- Generate .dvi output using `latex` command
- Make sure figure fits in one page
- Convert .dvi to .eps using command line
  `dvips -E figure.dvi -o figure.eps`
- Open .eps file using ghostview and measure lower-left (Ax, Ay) and upper-right (Bx, By) coordinates
- Open .eps file using a text editor and look up
  `%%BoundingBox: X1 Y1 X2 Y2`
- Replace X1 Y1 X2 Y2 with Ax Ay Bx By
- Command `includegraphics` with option `clip` prints only BoundingBox area on the output

References

Damiano Varagnolo,
How To Make Pretty Figures With Matlab,
http://www.ee.kth.se/~damiano/Matlab/HowToMakePrettyFiguresWithMatlab.pdf
Version 8, 2010.

Yair Moshe,
Advanced Matlab Graphics and GUI,
http://sipl.technion.ac.il/new/Download/Matlab_Support/Matlab_Guides/Graphics%20and%20GUI%20using%20Matlab.pdf
November, 2010.

Richard E. Turner, Umesh Rajashekar
Publication quality figures using Matlab,
http://www.gatsby.ucl.ac.uk/~turner/TeaTalks/matlabFigs/matlabFig.pdf
June, 2011.

Jeffrey D. Hein,
Creating .eps figures using TikZ,
http://heinjd.wordpress.com/2010/04/28/creating-eps-figures-using-tikz/
April, 2010.