Windsim manual

Contents:

1 General

When MATLAB is running, the *MATLAB command window*, will appear. In this window it is possible to give commands.
First check if you are working in the right directory in MATLAB. Use `cd` to see which directory is active. If you use the command `dir` all the files in the directory will be shown. The *.m* files you see are those which will be used for the assignments. They can be opened in the c*ommand window*.

1.1 Call *.m* files

The given *.m* files don't need to be changed. In the *command window* you call the functions and you insert the parameters.
All *.m* files start with:

```
function [outputs] = function name(inputs)
```

When you call an *.m* file, you have to insert values for the `inputs.` This value can be a number as well as a vector or matrix. Pay attention to the order of the parameters of the function. An example:

```
function  [Cdax,Cp, a] = cplambda(windturbine, lambda, theta)
```

The input `windturbine` requires the name of the file with the parameters of a wind turbine, e.g. the Lagerwey 50/750. The parameters from this wind turbine are stated in the *.m* file *LW50.m*. `Lambda` is a vector of different tip speed ratios, `theta` is the pitch angle of the blade. Both are free to choose. The file *cplambda.m* is called from the *command window* as follows.

```
>> [Cdax1, Cp1, a1] = cplambda('LW50', 0.5 : 0.1 :1.5, 1)
```

LW50 is put between quotation marks because it is a string. A vector `lambda` is created as a second parameter that runs from *0.5* to *1.5* in steps of *0.1*. (It is possible to see that on line 27 in *cplambda.m* the length of `lambda` is taken. Thus for `lambda` a vector of a random length can be chosen.)
For `theta` the value 1 is used in this case.

The outputs (results) of *cplambda.m* are saved in the variables left from the = sign: after the call, *Cdax1* equals the thrust coefficient C_{dax} (for theta is 1 and lambda runs from 0.5 to 1.5 on steps of 0.1). The values of `Cdax1` appears on the screen through:

```
>> Cdax1
```

If you call the function again with different values for the inputs, it is possible to save the outputs of the function in new variables:

```
>> [Cdax2, Cp2, a2] = Cplambda('LW50', .5:.1:1.5, .2)
```

In this way you can calculate `Cdax`, for different values of `theta`, and compare them.

As mentioned before, there is no need to change the given *.m* files, but in some assignments you may need parameters of a different wind turbine then the given ones. Just save for example *LW50.m* as a new *.m* file (e.g. LW70). In this file you can put the changes. With

```
>> [Cdax71, Cp71, a71] = cplambda('LW70', .5:.1:1.5, .1)
```

the same calculations are carried out, but for the new wind turbine.

Now you can put all the gathered data in a graph. For the layout of the graphs use the general help of Matlab.

1.2 Trics

By using the command *help*, the inputs and outputs of an *.m* file are shown immediately. For example for *cplambda.m*:

```
>> help cplambda
```

The program gives you the syntax and comments of *cplambda.m*. By using copy (*ctrl+c*) from the syntax and paste (*ctrl+v*), minimizes the chance of mistakes in the order of the parameters if you call a certain file. Furthermore, there is an explanation of all the inputs and outputs. In this way you can easily see which to use.

In case you deal with an unknown command in the *.m* files, *help* can provide information about the command. For example the command *plot* to make graphs:

```
>> help plot
```

The way to insert the parameters is obvious now as well as the possibilities to show graphs. An example of a plot will be given at the end of this manual together with an example of a script file. A script file is created by making a new .m file. Instead of typing several commands in the command window, you can place them together in a script file and call this file. On the other hand you can also call a function more then once with a script file by placing the function in a *for loop*.

Table 1 presents some useful information concerning the *command window.* In table 2, there are examples of the syntax of MATLAB.

Key/ command	***Action***
Arrow up	Shows former command
`close all`	Closes the figure windows.
`Help <subject>`	In case you want to know more about a *MATLAB* subject.
`Who`	Gives all the variables defined in MATLAB at that instant
`Clear`	Erases the memory

Table 1: Special keys and commands in command window.

Syntaxes	***Shows***
`a./b`	First element of *a* divided by first element of b etc. Also .* .^
`% in front of a line`	This line will be ignored by *MATLAB*: comments.
`;` behind a line	*MATLAB* won't show the result from the line on the screen.
`x(2)`	2nd element of vector x.
`M(2,:)`	2nd row of matrix M
`M(:,2)`	2nd column of matrix M
`for i=1:6`	Repeats for the values i=1,2,3,4,5,6
`length(vector)`	Gives the length of the vector.
`max(vector)`	Gives the largest element of a vector.

Table 2: Some syntax examples from *MATLAB*

2 Survey *MATLAB* programs

Paragraph 2.1 presents a graphical overview of the main files and the files called in these main files. The main files you need to call in *MATLAB* will be given in paragraph 2.2. Finally, paragraph 2.3 discusses the subroutines called by the main files.

The following goes with the *.m* files:

Run files	*Subroutines*
Syntax	
What does it	What does it
In which command is it used	From *.m* file is it called
Which functions are called	Which functions are called

2.1 Run files

Powercurve1

```
[Dax,Mbeta,Mr,P,Cdax,Cp,a,theta,omr]=Powercurve1(windturbine,V)
```

Given the wind speed and a wind turbine with variable rpm and blade pitch angle, the forces, torque and power is calculated. Firstly, the rpm of the rotor and the blade angle are determined. Two cases are distinguished: partial load $V<=V_n$ and full load $V>V_n$. If $V>V_n$, the power stays constant at rated value. *Bem.m* is used to calculate the induction factor that is used to calculate the forces.
It is assumed the wind turbine has an optimal λ control.

Assignment: Rotor, tower.

Called subroutines:

- LW50
- bem
- fun_power
- fzero

Powercurve2

```
[Dax, Mbeta, Mr, P, Cdax, Cp, a]=Powercurve1(windturbine, V, omr)
```

Similar to powercurve1, but here you deal with a wind turbine with stall control and constant rpm. Furthermore the blade angle is constant.
Assignment: Rotor

Called subroutines:

- LW50
- bem

Cplambda

```
[Cdax, Cp, a]=Cplambda(windturbine, lambda, theta)
```

Similar to powercurve1, but here you can insert a vector with different tip speed ratios λ.
Graphs can be created with Cd_{ax}, *Cp* and the induction factor *a* against λ.
Assignment: Rotor

Called subroutines:

- LW50
- bem

Transfer

```
[sys_tf,sys_ss]=Transfer(windturbine,V0)
```

Determination of the transfer functions of the wind turbine.

Called subroutines:

- LW50
- bem
- equi
- gener
- dynmod

2.2 Subroutines

LW50

Gives all the necessary parameters from the Lagerwey 50/750.
Called in powercurve1, powercurve2, Cplambda, transfer.

Dowec

Gives all the necessary parameters from the Dowec turbine.
Called in powercurve1, powercurve2, Cplambda, transfer.

V66

Gives all the necessary parameters from the Vestas V66.
Called in powercurve1, powercurve2, Cplambda, transfer.

Bem

Bem stands for blade element momentum theorem.
In *fun_bem* is for a given induction factor *a* the difference calculated between D_{ax} found using the blade element method and D_{ax} found with the momentum theorem $D_{ax}=4a(1-a))$.
In *bem.m* the *a* is determined with *fzero* for which the function *fun_bem* reaches a zero. The output from *bem.m* are the forces and torque following from this value of *a*.

Called subroutines:

- fun_bem
- Aero
- Aero2

Fun_bem

In fun_bem the difference is calculated between D_{ax} found using the blade element method and D_{ax} found using the momentum theorem; $D_{ax}=4a(1-a)$

Called subroutines:

- Aero

Aero

Given the parameters (induction factor, wind speed, rpm, flap velocity, tower top velocity and turbine characteristics), the forces per annulus are calculated and the sum is taken.
Called in Bem.
Called subroutines:

- Lift — Table Cl
- Drag — Table Cd

Aero2

Similar to aero, but a faster version because vectors are used in stead of for-loops.

Equi

Determination of the steady state; the operating point is the rpm where equilibrium between rotor torque and generator torque is established.
Called in Transfer
Called subroutines:

- bem
- fun_equi
- fun_power

Fun_power

Determines the difference between the nominal power and power based on given input parameters.
Called in powercurve1, equi.

Fun_equi

Determines the difference between the moment according to bem and the generator characteristic.
Called in equi.

Dynmod

Determination of the differential equations forming the equations of motion of the system.
Called in transfer.
Called subroutines:

- aero2
- gener

Gener

Torque rpm characteristic of generator.
Called in dynmod.

Gust1

Smooth wind gust.
Called in dynmod.

Gust2

A sinusoidal gust with a frequency equal to the rpm of the rotor (1P); this represents the variations in the wind speed felt by a blade element as a result of wind shear, yawed flow, tower shadow and rotational sampling of turbulence. The output of this routine can be used as input for the simulation via *step* and *lsim*.

Lift

Table with data of C_L for different angles of attack α ; the first column is the angle of attack and the second the lift coefficient C_L. When called with a certain α, the corresponding value for C_L is given as output.
Called in Aero.

Drag

Table with C_d against the angle of attack; the first column is the angle of attack and the second the C_d. When called with a certain α, the corresponding value for C_D is calculated as output.

Turbulence.mat

In this file there are sequential values of turbulent wind; the variables u (turbulence in m/s) and t (time in s) become available via 'load turbulence'. The turbulent wind can be used as an input for the simulation of the wind turbine via *step* en *lsim*.

2.3 Applied *MATLAB* routines

In this section some standard *MATLAB* routines used in the given *.m* files are discussed.
For more information (about inputs and outputs for example) use *help*, as mentioned before.

Lsim

```
[y,t,x]=lsim(sys,U,t1)
```

This *MATLAB* routine simulates the dynamical system specified in `sys`. In `sys` the transfer functions of a system are given. With a given time vector t1 and matrix U, *lsim* calculates the response of the system in the outputs y and states x.
The simulation of the wind turbine in case of a gust and an average wind speed of 14m/s is done as follows with *lsim*:

```
[systf,sys]=transfer('LW50',14);
t1=0:0.01:30;
U=gust1(t1);
[y,t,x]=lsim(sys, [zeros(size(t1));U],t1);
```

step

```
step(sys,Tend)
```

This a standard routine in *MATLAB*, similar to *lsim*, for the simulation of a dynamical system specified in `sys`. The routine calculates and plots the response of the system of a step shaped variation of all the inputs (in our case the pitch angle of the blade and the undisturbed wind speed).

```
[systf,sys]=transfer('LW50',14);
step(sys, 20);
```

fzero

Searches the zero of a function. Different options can be given like an initial value or an interval in which the zero has to be searched.

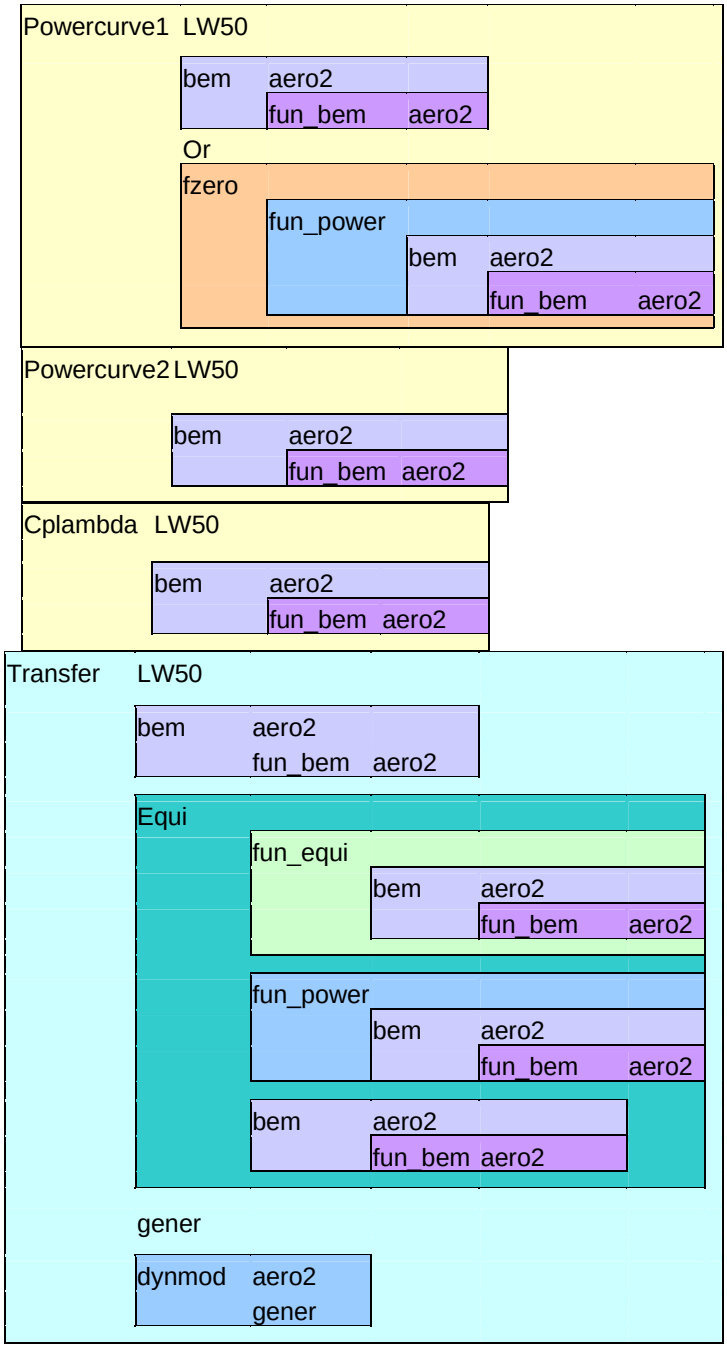
2.4 Graphical overview

Powercurve1.m

Powercurve2.m

Cplambda.m

Transfer.m



3 Some examples

Example of a script file

```
% In a for-loop, we call for different thetas cplambda and
% plot Cpmax against the corresponding theta.
clear;

thet=-1.5:.1:.5
n=length(thet);
lambda=.5:.1:1.5;

for i=1:n
      theta=thet(i);
      [Cdax,Cp,a]=Cplambda('LW50', lambda, theta);
      Cpmax(i)=max(Cp);
end
plot(thet, Cpmax);
```

Example of graphical output

```
% Graph of y=x^2. For 11 points, the value of y is calculated.
% These points are plotted in red (r) with an *
% (*)for the calculated points and a line (-) between the
% points.

x=0:1:10;
y=x.^2
plot(x,y, 'r*-');
shg;                                  % show graph on screen
title('The graph of y against x');
xlabel('x');
ylabel('y');

pause

% Another graph: In blue the lift curve is drawn.

alpha=-90:90
Cl=lift(alpha);
plot(alpha,Cl, 'b');
title('The lift curve for the given blade profile.');
xlabel('alpha');
ylabel('Cl');
```