C:\Users\jstho\Desktop\VestasPic.png

wtLin Guide

A wind turbine linear analysis tool

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Who** | **Description** |
| 00 | 22-Dec-2017 | JSTHO | Initial version |
| 01 |  | ANDBN |  |
|  |  |  |  |
|  |  |  |  |

# Table of Contents

[0 Table of Contents 3](#_Toc502738242)

[1 wtLin overview 5](#_Toc502738243)

[1.1 How-to generate mat-file 6](#_Toc502738244)

[1.2 How-to perform analysis 6](#_Toc502738245)

[1.3 How-to perform parameter sweeps 7](#_Toc502738246)

[2 Components overview 8](#_Toc502738247)

[2.1.1 Inputs and outputs 9](#_Toc502738248)

[2.2 Matlab connect examples 9](#_Toc502738249)

[2.2.1 Example 1 10](#_Toc502738250)

[2.2.2 Example 2 10](#_Toc502738251)

[2.2.3 Example 3 11](#_Toc502738252)

[3 Turbine components 12](#_Toc502738253)

[3.1 Converter 12](#_Toc502738254)

[3.1.1 Converter with time constant 12](#_Toc502738255)

[3.1.2 Converter unity model 12](#_Toc502738256)

[3.1.3 Converter unity model with DTD 12](#_Toc502738257)

[3.2 Generator 12](#_Toc502738258)

[3.2.1 Generator model 12](#_Toc502738259)

[3.3 Drivetrain 13](#_Toc502738260)

[3.3.1 Flexible drivetrain model 13](#_Toc502738261)

[3.3.2 Stiff drivetrain model 14](#_Toc502738262)

[3.3.3 Stiff drivetrain model with nacelle rotation 14](#_Toc502738263)

[3.3.4 Stiff drivetrain model with external torque 14](#_Toc502738264)

[3.3.5 Stiff drivetrain model with external torque and nacelle rotation 14](#_Toc502738265)

[3.4 Aerodynamic torque 15](#_Toc502738266)

[3.4.1 Aerodynamic torque in full load 15](#_Toc502738267)

[3.4.2 Aerodynamic torque in partial load 15](#_Toc502738268)

[3.5 Aerodynamic thrust 15](#_Toc502738269)

[3.6 Rotor wind 16](#_Toc502738270)

[3.7 Fore-aft tower model 16](#_Toc502738271)

[3.8 Side-side tower model 16](#_Toc502738272)

[3.9 Pitch system dynamics 17](#_Toc502738273)

[3.9.1 Pitch model 17](#_Toc502738274)

[3.9.2 Pitch model without dynamics 17](#_Toc502738275)

[4 Controller components 18](#_Toc502738276)

[4.1 Full load controller (FLC) 18](#_Toc502738277)

[4.2 Partial load controller (PLC) 18](#_Toc502738278)

[4.3 Fore-aft tower damper (FATD) 18](#_Toc502738279)

[4.4 Band-stop filters 18](#_Toc502738280)

[4.5 Drive train damper (DTD) 19](#_Toc502738281)

[5 Appendix: Rotor sensitivities 21](#_Toc502738282)

[6 Appendix: Future development and notes 23](#_Toc502738283)

# wtLin overview

wtLin is an internal tool for linear analysis of wind turbines. For example, the full load controller closed loop dynamics can be analyzed.

The tool can be found in the “wtLinPath” below, where “C:\repo\tsw” is you own TSW-repository, and thus it might have another name.

C:\repo\tsw\application\phTurbineCommon\Simulink\ControllerConfiguration

A VTS setup can be used for generating a mat-file which acts as input to wtLin. The mat-file contains all extracted source parameters, but when imported by wtLin the parameters change into wtLin parameters with new naming. Afterwards linear parameters and systems can be derived. See the figure below where shaded block are data and white blocks are functions.



* **Get params:** A script is used for generating mat-file. The script uses parameter from VTS master-file, parts files including controller, and an aero-file for the turbine. In addition, other parameters need to be found, for example DTD parameters must be found and entered manually.
* **Import mat:** A function within wtLin which maps parameters from the mat-file to wtLin “source” parameters, denoted gross parameters (gp).
* **Gross parameters:** Structure with all source parameters (gp).
* **Generate operating point:** Function for generating operating point values (op).
* **Operating point:** Structure with operating point values (op).
* **Calc lin params:** Takes the operating point and the gross parameter and generating new parameters valid for that specific operating point. These parameters are called linear parameters (lp). A linear parameter could for example be the effective FLC gain, or it could be a sensitivity used in the aerodynamic models.
* **Lin params:** Structure with all linear parameters (lp).
* **Calc comp:** This function takes all linear parameters and calculates all components. Components are linear dynamic systems like the generator, converter, pitch system, full load controller, and so on.
* **Components:** Structure (comp) with dynamic systems. Each system is a transfer function or state space model, and each system has defined input and output names.
* **Calc standard loops:** Some default systems are defined within wtLin. This function calculates these default systems.
* **Standard dyn systems:** Structure (loop) with dynamic systems composed by connection of components.
* **Calc own loops:** It is possible to connect components (in comp-structure) by using the matlab function “connect” as each component has defined input and output names. In this way, open-loop and closed-loop system can be calculated. New system can even be made and included in the analysis.
* **Own dyn systems:** Own dynamic system produced using the connect command.

## How-to generate mat-file

In the following it is described, step-by-step, how-to generate a mat-file.

* Goto folder “wtLinPath”
* Copy template file “wtLin\_example\_getParam\_matVer0.m” to own work folder.
* Change name of file to fit turbine, for example to “wtLin\_getParam\_V126\_Mk2A.m”.
* Find a VTS setup that you want to use. In theory, the wtLin version should be the same as the software version used in the VTS setup. So, the best approach would be to generate a new controller code from Simulink using the same repo as the wtLin version used. If this is not possible, or too cumbersome, try to see if it works anyway. Problems usually only arise if names of core controller parameters have changed.
* Modify each line in the script. Make sure parameters and paths are correct. DTD parameters might be difficult to find, but you might trust settings provided in:
  + Y:\\_Data\LAC\Control\ControlTools\Shared\wtLinExtra\DTD\_Relations.xlsx
* Run the script. A mat-file should be generated (the name of the file is specified in the script).

## How-to perform analysis

In the following it is described, step-by-step, how-to generate and plot dynamic systems.

* Goto folder “wtLinPath”
* Copy template file “wtLin\_example\_use.m” to own work folder.
* Change name of file to fit analysis, for example to “wtLin\_SoftTowers\_V126\_Mk2A.m”.
* Modify script to your needs.
  + You need to select an operating point for each analysis (\*).
  + Before each analysis any number of gross parameters (gp) can be changed (just remember to change back again after analysis).
  + Before calculating components linear parameters (lp) can be changed. This might be preferred to speed up execution time for wtLin, but usually gp’s are changed directly.
  + You need to decide whether to use standard loops and own customized systems. Both methods are included in the template. It is advised to use the own customized systems due to flexibility, though, the risk of errors is higher.
  + Several variants of dynamic systems can be calculated, simply by repeating the analysis.
  + Different turbines can also be compared by loading different mat-files before each analysis.
  + Define plot functions for desired dynamic systems. For example, bode plots of closed loops systems.
* Run the script.

(\*) Consisting of

1. Select operating point (op)
2. Generate linear parameters (lp)
3. Generate components (comp)
4. Generate system, either standard systems (loop) and own systems.

## How-to perform parameter sweeps

In the following it is described, step-by-step, how-to perform a parameter sweep. Notice, this way of handling wtLin is less flexible as the default method described above.

* Goto folder “wtLinPath”
* Copy template file “wtLin\_example\_paramStudy.m” to own work folder.
* Change name of file to fit analysis, for example to “wtLin\_sweepKpFLC\_V126\_Mk2A.m”.
* Read help text for function.
* Modify script to your needs.
* Run the script.

# Components overview

The system to be analyzed is composed by components which are connected by Matlab’s “connect”-function. A quick overview of components is given in the table below. Some components are different versions of the same system, and thus, only one of those can be selected, i.e. each output name must be unique.

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Name** | **Inputs** | **Outputs** |
| Band stop filters in SP on measured generator speed. | FBfilt | w | wFilt |
| Band stop filters in SP on measured generator speed. Discrete 10Hz version. | FBfiltd10Hz | w | wFilt |
| Full Load Controller | FLC | e | thRef |
| Fore Aft Tower Damp | FATD | py  vy | thFatd |
| Partial Load Controller | PLC | e | Pref |
| Pitch controller | pit | thRef  thFatd | th |
| Pitch direct feed-through | pitUn | thRef  thFatd | th |
| Converter | cnv | Pref | Pconv |
| Converter direct feed-through | cnvUn | Pref | Pconv |
| Converter direct feed-through with DTD input | cnvDtdUn | Pref  Pdtd | Pconv |
| Rotor wind speed taking tower movement into account. Include this if tower fore aft is included, otherwise not needed. | rotWind | vfree  vy | v |
| Fore aft tower model | towSprMassFa | Trot | py  vy  ay |
| Side-side tower model | towSprMassSs | Mrot  Mgen | pWy  vWy |
| Aerodynamic thrust model | aeroThr | th  v  W | Trot |
| Aerodynamic torque (moment) model (presently, divided into FLC and PLC due to model problems) | aeroFLC | th  v  W | Mrot |
| Aerodynamic torque (moment) model (presently, divided into FLC and PLC due to model problems) | aeroPLC | th  v  W | Mrot |
| Generator | gen | Pconv  W | Mgen |
| Drive train model | drt | Mrot  Mgen | W  w |
| Drive train model including side-side tower movements | drtSs | Mrot  Mgen  vWy | W  w |
| Drive train model including damping by generator torque (testing only) | drtDmp | Mrot  Mgen  dmpMgen | W  w |
| Drive train model including side-side tower movements, and including damping by generator torque (testing only) | drtSsDmp | Mrot  Mgen  dmpMgen  vWy | W  w |
| Controller for rotor damping by power (testing only) | rotDmpFLC | W | dmpMgen |
| Drive train damper, DTD | DTD | w | Pdtd |

### Inputs and outputs

|  |  |  |  |
| --- | --- | --- | --- |
| **Description** | **Name** | **Symbol** | **Unit** |
| Generator speed | w |  | rpm |
| Rotor speed | W |  | rpm |
| Generator speed error | e |  | rpm |
| Pitch angle | th |  | deg |
| Reference pitch angle | thRef |  | deg |
| Pitch offset from FATD | thFatd |  | deg |
| Power reference to converter | Pref |  | W |
| Power from converter | Pconv |  | W |
| Power from DTD | Pdtd |  | W |
| Free wind speed | vfree |  | m/s |
| Free wind speed modified by tower mevements | v |  | m/s |
| Rotor torque | Mrot |  | W/(rad/s) = Nm |
| Generator torque (mapped to low speed side) | Mgen |  | W/(rad/s) = Nm |
| Rotor thrust (force) | Trot |  | W/(rad/s) = Nm |
| Tower position, velocity and acceleration in y-direction (i.e. fore aft) | py  vy  ay |  | m  m/s  m/s^2 |
| Tower position, velocity and acceleration in x-direction (i.e. side-side) | px  vx  ax |  | m  m/s  m/s^2 |
| Nacelle rotation around y-axis | pWy  vWy  aWy |  | rad  rad/s  rad/s^2 |
|  |  |  |  |

## Matlab connect examples

Notice, systems (components) are connect by names so order does not matter.

### Example 1

Create a system from reference generator speed (wRef) to generator speed (w) for full load operation, including FATD, fore aft tower, no converter dynamics and no pitch dynamics. For this to work ALL needed components MUST be included. For example, the pitch system cannot be left out as this would break the closed loop system. In addition, the ‘SumRef’ system must be included to close the loop.

|  |
| --- |
| SumRef = sumblk('e','wRef','w','+-');  sysCL\_wRef2w\_FLC = connect(c.FLC,c.FATD,c.pitUn,c.cnvUn,c.gen,c.drt,c.aeroFLC,… c.aeroThr,c.rotWind,c.towSprMassFa,SumRef,'wRef','w'); |

The resulting system has been sketched below. Notice, the arrows starting with a square are connected to other blocks.



### Example 2

Create a system from free wind speed (vfree) to fore-aft tower velocity (vy) for full load operation, including FATD, fore aft tower, no converter dynamics and no pitch dynamics. For this to work ALL needed components MUST be included, and this includes the speed feedback given by ‘SumRef’.

|  |
| --- |
| SumRef = sumblk('e','wRef','w','+-');  sysCL\_vfree2vy\_FLC = connect(c.FLC,c.FATD,c.pitUn,c.cnvUn,c.gen,c.drt,c.aeroFLC,… c.aeroThr,c.rotWind,c.towSprMassFa,SumRef,'vfree','vy'); |

### Example 3

Create a system from free wind speed (vfree) to side-side tower angular velocity (vWy) for full load operation, including DTD, side-side tower, no converter dynamics and no pitch dynamics. For this to work ALL needed components MUST be included, and this includes the speed feedback given by ‘SumRef’.

|  |
| --- |
| SumRef = sumblk('e','wRef','w','+-');  sysCL\_vfree2vy\_FLC = connect(c.FLC,c.pitUn,c.DTD,c.cnvDtdUn,c.gen,c.drtSs,c.aeroFLC,… c.towSprMassSs,SumRef,'vfree','vWy'); |

# Turbine components

## Converter

The converter is modelled in different ways. At least for newer converter types, the converter dynamics is very fast compared to rotor and tower dynamics and therefore it makes sense to simply model it as a direct feed-through. Different linear models of the converter is given below.

Relevant linear parameters can be found under “lp.s.mp.cnv” and “lp.s.mp.dtd”.

### Converter with time constant

Name: *cnv*. Inputs . Outputs .

The converter is modelled as a first order system. Presently, the time constant has no associated source-parameter, i.e. gp-parameter. The model is mostly relevant for old turbine variants as new variants have a very high bandwidth.

### Converter unity model

Name: *cnvUn*. Inputs . Outputs .

The converter is modelled as direct feed-though.

### Converter unity model with DTD

Name: *cnvDtdUn*. Inputs . Outputs .

This model is needed if DTD is to be included.

## Generator

### Generator model

Name: *gen*. Inputs . Outputs .

In VTS the power conversion efficiency is given by tables taking output grid power and generator speed as input and outputting efficiencies (electric, mechanic and auxiliary), i.e.

The power is the generator power or the “No Loss” power.

Based on this the power loss is given by

The generator torque (referred to low speed side) can be approximated as shown below. The low speed velocity is given by where is the fixed gear ratio (notice, is only identical to for a stiff drive train).

Around an operating point this can be written as below, where in this equation the variables , and are small-signals. This is the generator model.

The power to generator torque sensitivity can be written as (not small-signals)

The rotational speed to generator torque sensitivity can be written as (derivative of multiplication)

Notice, in wtLin the present generator model has input and . Thus, this assumes a stiff drivetrain, i.e. .

Notice, the terms , , and are calculated within wtLin based on the operating point are other relevant parameters.

Notice, in full load operation the small signal equals zero, so the first term in the generator model will also be zero. The second part consists of the term . This is a negative damping term caused by keeping a constant power level.

## Converter and Generator combined

**Pending**

## Drivetrain

Several drivetrain models exist in wtLin.

Torques are referred to the low speed side (Lss) and inertia’s must be mapped accordingly.

### Flexible drivetrain model

**Review**

Name: *drtFree2*. Inputs . Outputs .

Model with two free inertia coupled with a spring (with stiffness ) and a damper (with damping ).

### Stiff drivetrain model

Name: *drt*. Inputs . Outputs .

Model is a direct map to angular acceleration. Angular velocity is given by integration.

So, the angular velocities are given by

### Stiff drivetrain model with nacelle rotation

Name: *drtSs*. Inputs . Outputs .

This is a stiff drivetrain model as before, except nacelle rotation due to tower bending is included. This is included by an input describing the relative angular velocity.

So, the angular velocities are given by

### Stiff drivetrain model with external torque

Name: *drtDmp*. Inputs . Outputs .

This model is similar to “drt”, except that an additional torque input is applied. This system is only for testing.

### Stiff drivetrain model with external torque and nacelle rotation

Name: *drtSsDmp*. Inputs . Outputs .

This model is similar to “drtSs”, except that an additional torque input is applied. This system is only for testing.

### Flexible drivetrain model with nacelle rotation

**Pending**

## Aerodynamic torque

**Review**

Aerodynamic rotor torque can be written as below.

With fixed sensitivities around an operating point the torque can be approximated by (now all variables are small-signals)

### Aerodynamic torque in full load

Name: *aeroFLC*. Inputs . Outputs .

The model is simply given by the model above.

### Aerodynamic torque in partial load

Name: *aeroPLC*. Inputs . Outputs .

The model includes induction lag by inserting a low pass filter on the generator speed. The low pass filter is parameterized by the induction lag time constant .

Basically, the induction lag time constant should describe how the flow through the rotor is preserved when for example pitching. However, currently wtLin does not include an induction state variable, and therefore, the model above is used in an attempt to mitigate this. Though, it would be better to update wtLin.

## Aerodynamic thrust

Name: *aeroThr*. Inputs . Outputs .

Aerodynamic rotor thrust can be written as below.

With fixed sensitivities around an operating point the thrust can be approximated by (now all variables are small-signals)

## Rotor wind

Name: *rotWind*. Inputs . Outputs .

This model is included to describe the interaction from tower fore-aft movement and wind speed seen by the turbine. Notice, and tables take the distant (free) wind speed as inputs, and therefore, the wind speed is not the rotor wind speed. Instead is the free wind speed modified by the tower fore-aft velocity .

## Fore-aft tower model

Name: *towSprMassFa*. Inputs . Outputs .

The fore-aft tower dynamics is modelled by a mass, spring and damper system with no rotation.

Stiffness is calculated based on tower fore-aft mass (sum of hub mass, nacelle mass and rotor mass) and specified tower 1st eigenfrequency.

The damping coefficient is calculated based on a fixed damping ratio of . Using such fixed number should not pose a problem for the model as damping primarily comes from the aerodynamics; i.e. via .

## Side-side tower model

Name: *towSprMassSs*. Inputs . Outputs .

The side-side tower dynamics is modelled by a mass, spring and damper system with no rotation.

The effective tower height that describes the relation between torque and effective sideways force is given by a crude approximation. is the hub height.

Stiffness and damping coefficient are calculated as for the fore-aft tower model. Thus, in this case the fixed damping ratio poses a problem for the model.

Output from the model is the angular rotational position and velocity, and . These variables are given by the effective hub height.

## Pitch system dynamics

### Pitch model

Name: *pit*. Inputs . Outputs .

The pitch system includes the pitch controller and in wtLin it is modelled by a simple first order system.

### Pitch model without dynamics

Name: *pitUn*. Inputs . Outputs .

This version of the pitch system is a direct feed-through.

### Pitch model extended

**Pending**

# Controller components

## Full load controller (FLC)

### PI Controller

Name: *FLC*. Inputs . Outputs .

The full load controller is a PI-controller given by

The gain and the integration time are given by source parameters and gain scheduling in the specified operating point.

### Full load controller Pre filters

**Pending**

## Partial load controller (PLC)

### PI Controller

Name: *PLC*. Inputs . Outputs .

The partial load controller is a PI-controller given by

The gain and the integration time are given by source parameters and gain scheduling in the specified operating point.

### Full load controller Pre filters

**Pending**

## Fore-aft tower damper (FATD)

Name: *FATD*. Inputs . Outputs .

FATD is a state feedback controller using each tower state in the fore-aft direction.

The parameter is the position feedback gain, the is the velocity feedback gain.

In turbine software, the position and velocity are not measured directly, but instead estimated based on the measured fore-aft tower acceleration and leaky integrators (i.e. low pass filters). This method affects the system dynamics, but presently, the filters are not included in wtLin.

## Band-stop filters

Name: *FBfilt*. Inputs . Outputs .

wtLin contains source parameters (gp) for each band stop filter on the speed. The parameters can be found under “gp.s.ctr.sp.bs”. In wtLin a transfer function for each of these filters is generated, and all filters is multiplied to get a total transfer function.

The code used by wtLin to generate each transfer function is the same as being used in the turbine software (though, deviations might occur as wtLin and turbine code are not linked together).

## Drive train damper (DTD)

**Review**

Name: *DTD*. Inputs . Outputs .

The drive train damper (DTD) is illustrated below. The basic idea is to introduce a damping term by generator speed feedback. The generator speed is filtered around the drive train frequency and multiplied with a gain to get a damping signal. The signal is gain scheduled to work in different speed operating points. The reason for this is that the damping output signal power rather than torque.

Notice, by using a power output instead of a torque output, then a negative damping term is introduced.

Systems

* : Slow filter to get the operating point. Presently, not included in wtLin as the operating point is selected instead.
* : Normalization of rotational speed to handle different speed operating points.
* : High pass filter to remove low frequencies.
* : Resonance filter.
* : High-pass-low-pass filter. This filter can be used instead of the resonance filter.
* : Open general 2nd order filter. Not used in turbine or in wtLin.
* : Switch to select between the resonance filer and the high-pass-low-pass filter.



The transfer function can be described as below. The time constant is found from “lp.s.mp.dtd.fHpPre”.

The transfer function can be described as below. The time constants and are found from “lp.s.mp.dtd.fHpDtd” and “lp.s.mp.dtd.fLpDtd”. The constant is given by “lp.s.mp.dtd.K\_LPF\_HPF\_DTD”.

The transfer function can be described as below. The gain and the frequencies and are indirectly (via a function) specified by parameters “lp.s.mp.dtd.GDT\_DTD\_dB”, “lp.s.mp.dtd.fDT” and “lp.s.mp.dtd.eta\_BW”. Notice, the similarity to the high-pass-low-pass filter. Though, the high-pass-low-pass filter can only have real poles while the resonance filter can have imaginary poles; hence, a resonance filter.

The transfer function is not included in wtLin as a dynamic system. Instead an operating point is used. (Notice, in turbine code the frequency for the low pass filter is given by “fBW\_OMEGAR\_DTD\_LPFPx” which equals 0.032 Hz for all variants).

The “design speed” for the gain scheduling is given as stated below. The final gain scheduling factor is .

Remark: Notice, that the scaling for DFIG is somewhat “strange” as grid frequency is fixed for all variants and conversion factor to should be . Though, it is not a problem as could be any number expect zero.

# Appendix: Rotor sensitivities

Within wtLin sensitivities are calculated in a certain way. Derivation of some sensitivities are shown in this section.

First some definitions and general formulas.

Rotor area:

Rotational speed with gear ratio :

Tip speed ratio and wind speed: and

Tip speed ratio derivatives: and

Torque coefficient:

Derivative of with respect to wind speed:

Derivative of with respect to rotational speed:

The rotor power can be expressed as

The rotor torque can be expressed as

**Sensitivity**

**Sensitivity**

**Sensitivity**

# Appendix: Future development and notes

* The generator model should take generator speed as input instead of rotor speed. The generator speed can just be scaled inside to . This should be easy to fix.
* Investigate DTD transfer function
  + Check gain scheduling for each type (DFIG and FullScale) by comparing to VTS and discuss with converter team. Notice, VTS might not be correct. Notice, FullScale version seems to be correct.
  + Investigate if dynamics for operating point filter HwLp plays a role, i.e. route input generator speed through filter instead of selecting the operating point.
  + Update DTD with respect to variants. VCS-V and GAPC should not be used for handling variants, instead use DFIG and FullScale.
  + Update wtLin code according to DTD component section.
  + Remove transfer function ‘Hwest’ from wtLin code. This filter is not present in real DTD. Only part of converter teams design document to represent uncertainties.
* Implement Umdamp function as an alternative to DTD. Umdamp is the old drive train damper.
* Investigate if it is a good idea to save old wtLin versions to align with software releases; i.e. difficult to match right software version. The down-side to this is missing wtLin functionality in older versions.
* SSTD should be included.
  + SSTD power: This can be done by changing cnv input to PrefConv and make a new system with inputs Pref and Psstd, and outputs PrefConv.
  + SSTD pitch: Make new input (force) to side-side tower model. The force must not affect the angular displacement!
  + Side-side tower model should also output position and velocity.