

# Advanced Control Design for Wind Turbines

## Part I: Control Design, Implementation, and Initial Tests

A.D. Wright and L.J. Fingersh

*Technical Report*  
NREL/TP-500-42437  
March 2008

NREL is operated by Midwest Research Institute • Battelle Contract No. DE-AC36-99-GO10337



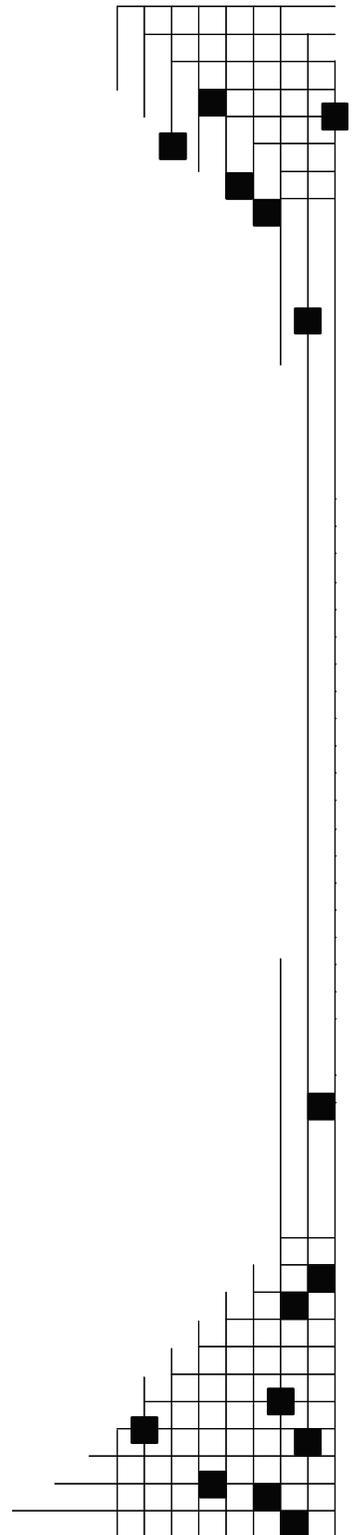
# Advanced Control Design for Wind Turbines

## Part I: Control Design, Implementation, and Initial Tests

A.D. Wright and L.J. Fingersh

Prepared under Task No. WER8.2101

*Technical Report*  
NREL/TP-500-42437  
March 2008



**National Renewable Energy Laboratory**  
1617 Cole Boulevard, Golden, Colorado 80401-3393  
303-275-3000 • [www.nrel.gov](http://www.nrel.gov)

Operated for the U.S. Department of Energy  
Office of Energy Efficiency and Renewable Energy  
by Midwest Research Institute • Battelle

Contract No. DE-AC36-99-GO10337

## NOTICE

This report was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or any agency thereof.

Available electronically at <http://www.osti.gov/bridge>

Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from:

U.S. Department of Energy  
Office of Scientific and Technical Information  
P.O. Box 62  
Oak Ridge, TN 37831-0062  
phone: 865.576.8401  
fax: 865.576.5728  
email: <mailto:reports@adonis.osti.gov>

Available for sale to the public, in paper, from:

U.S. Department of Commerce  
National Technical Information Service  
5285 Port Royal Road  
Springfield, VA 22161  
phone: 800.553.6847  
fax: 703.605.6900  
email: [orders@ntis.fedworld.gov](mailto:orders@ntis.fedworld.gov)  
online ordering: <http://www.ntis.gov/ordering.htm>



## Summary

Wind turbines are complex, nonlinear, dynamic systems forced by gravity, stochastic wind disturbances, and gravitational, centrifugal, and gyroscopic loads. The aerodynamics of wind turbines are nonlinear, unsteady, and complex. Turbine rotors are subjected to a complicated 3-D turbulent wind inflow field, which drives fatigue loading.

Wind turbine modeling is complex and challenging. Accurate models must contain many degrees of freedom to capture the most important dynamic effects. Design of control algorithms for wind turbines must account for these complexities. These algorithms must capture the most important turbine dynamics without being too complex and unwieldy.

Typical large commercial wind turbines are variable speed, and control generator torque in Region 2 to maximize power and control blade pitch in Region 3 to maintain constant turbine power. Simple classical control design techniques such as proportional-integral-derivative (PID) control for pitch regulation in Region 3 are typically used to design the controls for such machines.

Classical control design methods are based on a single input and single output. A disadvantage of classical control methods is that multiple control loops must be used to simultaneously damp several flexible turbine modes. If these controls are not designed with great care, these control loops interfere with each other and cause the turbine to become unstable. The potential to destabilize the turbine grows as turbines become larger and more flexible, and the degree of coupling between flexible modes increases. Using all the available turbine actuators in a single control loop to maximize load-alleviating potential is advantageous. Advanced multi-input multi-output (MIMO) multivariable control design methods, such as those based on state-space models, can be used to meet these multiple control objectives and use all the available actuators and control inputs in a single control loop.

The purpose of this report is to give wind turbine engineers information and examples of the design, testing through simulation, field implementation, and field testing of advanced wind turbine controls. This report will be Part I in a two-part series of reports that detail advanced control design, implementation, and test results. Part I (this report) will highlight the control development process, from forming control objectives, to designing the controller, to testing the controller through analytical simulation, to field implementation and initial field testing. Part II (to be completed later) will give a detailed comparison of results from advanced load alleviating state-space controllers to test results from baseline controllers without load alleviation. The purpose of Part II is to demonstrate through rigorous testing the load mitigating potential of the advanced state-space controllers compared to the baseline control.

## **Acknowledgments**

The authors would like to thank Dr. Michael Robinson of the National Renewable Energy Laboratory for his management support and the managers at the U.S. Department of Energy for project funding and support. In addition, Dr. Karl Stol of the University of Auckland provided many valuable suggestions and contributions in state-space control modeling and testing. Dr. Kathryn Johnson of the Colorado School of Mines provided invaluable suggestions and guidance in the implementation and testing of these advanced controls in the Controls Advanced Research Turbine. The authors would like to thank Dr. Maureen Hand for her work in developing the interface between Simulink and the FAST simulation code. We thank Dr. Jason Jonkman for his valuable suggestions in preparing the description of the baseline controller designs for this report.

Finally, but not least, the authors would like to thank Garth Johnson and Scott Wilde of the National Renewable Energy Laboratory for their invaluable technician support in maintaining the Controls Advanced Research Turbine so that this testing could be performed.

## Common Symbols

$A$ -state matrix,

$\bar{A}$ -state matrix augmented with disturbance states

$A_a$ -state matrix in actuator dynamics model

$A_c$ -state matrix calculated in control synthesis routine

$A_d$ -discrete time version of  $A$

$B$ -control input matrix

$\bar{B}$ -control input gain matrix augmented with disturbance input

$B_a$ -control input gain matrix in actuator dynamics model

$B_c$ -control input gain matrix calculated in control synthesis routine

$B_d$ -wind input disturbance matrix

$B_d$ -discrete time version of  $B$

$C$ -output state matrix

$\bar{C}$ -relates plant output to plant and disturbance states

$C_a$ -relates plant output to states in actuator dynamics model

$C_c$ -C matrix calculated in control synthesis routine

$C_d$ -discrete time version of C

$C_{ont}(s)$ -controller transfer function

$C_p$ -power coefficient

$C_{p_{max}}$ -maximum power coefficient

$C_t$ -tower damping coefficient associated with first fore-aft mode

$D$  -control input transmission matrix

$D_{_c}$  - $D$  matrix calculated in control synthesis routine

$D_{_d}$  -discrete time version of  $D$

$D_d$  -wind input disturbance transmission matrix

$D_{d_{_c}}$  - $D_d$  matrix calculated in control synthesis routine

$E$  -set of eigenvalues of closed-loop system

$F$  -state matrix for disturbance state equation

$F_{ilt}(s)$  -filter transfer function

$F_t$  -pitch control input gain corresponding to first tower fore-aft mode

$G$  -gain in full state feedback law

$G_d$  -gain in full state feedback law associated with disturbance state

$I_{gen}$  -generator mass moment of inertia relative to high-speed shaft

$I_{rot}$  -total rotational moment of inertia due to rotor, gear-box, shafts, generator, etc.

$J$  -quadratic cost function

$K$  -state estimator gain matrix

$K_d$  -disturbance state estimator gain matrix

$k$  -gain multiplying  $\Omega^2$  in Region 2 generator torque expression

$K_D$  -classical controller derivative gain

$K_p$  -classical controller proportional gain

$K_i$  -classical controller integral gain

$K_t$  -tower stiffness coefficient associated with first fore-aft mode

$M_t$  -tower mass coefficient associated with first fore-aft mode

$m$  -power law wind-shear coefficient

$N$  -dimension of state matrix  $A$

$N_{gear}$  -gearbox ratio

$P$  -Solution of Ricatti Equation

$P(s)$  -plant transfer function

$Q$  -symmetric, positive semidefinite weighting on the states  $\underline{x}$

$Q_{aero}$  -aerodynamic torque

$Q_{gen}$  -generator torque

$Q_1$  -generator torque at beginning of Region 2½

$Q_2$  -generator torque at end of Region 2½

$Q_{rated}$  -rated generator torque

$R$  -symmetric, positive definite weighting on the input  $\underline{u}$

$R$  -Rotor radius

$s$  -Laplace variable

$t$  -time

$T_c(s)$  -Closed-loop transfer Function

$\underline{u}$  -control input

$\underline{u}_{op}$  -equilibrium value of control input

$\Delta \underline{u}$  -control input perturbation

$\Delta \underline{u}_a$  - input to actuator dynamics model

$\underline{u}_d$  -disturbance state-space model output

$\underline{u}_{d,op}$  -equilibrium value of disturbance state-space model output

$\Delta\underline{u}_d$  -disturbance state-space model output perturbation

$\Delta\hat{\underline{u}}_d$  -estimated disturbance state-space model output perturbation

$w$  -wind disturbance (uniform over rotor disk)

$\Delta w$  -wind disturbance (uniform over rotor disk) perturbation

$w_0$  -wind speed at control design point (uniform over rotor disk)

$\underline{x}$  -state vector

$\underline{x}_{op}$  -equilibrium value of state vector

$\Delta\underline{x}$  -state vector perturbation

$\Delta\underline{x}_a$  - actuator linear model state vector

$\Delta\hat{\underline{x}}$  -estimated state vector perturbation

$\dot{\underline{x}}$  -time derivative of  $\underline{x}$

$\dot{\underline{x}}_{op}$  -equilibrium value of time derivative of  $\underline{x}$

$\Delta\dot{\underline{x}}$  -time derivative of  $\underline{x}$  perturbation

$\Delta\hat{\dot{\underline{x}}}$  -time derivative of estimated  $\underline{x}$  perturbation

$\underline{y}$  -control (or measured) output

$\underline{y}_{op}$  -equilibrium value of control (or measured) output

$\Delta\underline{y}$  -control (or measured) output perturbation

$\Delta\underline{y}_a$  - output of actuator dynamics model

$\Delta \hat{y}$  -estimated control (or measured) output perturbation

$\underline{z}_d$  -disturbance state

$\Delta \underline{z}_d$  -perturbed disturbance state

$\Delta \hat{\underline{z}}_d$  -estimated perturbed disturbance state

$\Delta \dot{\underline{z}}_d$  -time derivative of perturbed disturbance state

$\Delta \dot{\hat{\underline{z}}}_d$  -time derivative of estimated perturbed disturbance state

$\alpha$  -partial derivative of rotor aerodynamic torque with respect to wind speed

$\delta$  - damping ratio

$\gamma$  -partial derivative of rotor aerodynamic torque with respect to rotor speed

$\lambda$  -Tip-speed ratio

$\lambda_{opt}$  -Optimum value of tip-speed ratio corresponding to  $C_{p_{max}}$

$\theta$  -blade pitch

$\Delta \theta$  -blade pitch perturbation

$\theta_0$  -blade pitch at control design point (equilibrium)

$\dot{\theta}$  -blade pitch rate

$\Delta \dot{\theta}$  -blade pitch rate perturbation

$\dot{\theta}_c$  -commanded blade pitch rate

$\Delta \dot{\theta}_{com}$  -commanded blade pitch rate perturbation

$\Theta$  -matrix relating the disturbance model output to the disturbance states

$\rho$  -air density

$\Omega$  -turbine rotational speed

$\Omega_1$  -turbine rotational speed at beginning of region 2½

$\Omega_2$  -turbine rotational speed at end of region 2½

$\Delta\Omega$  -turbine rotational speed perturbation

$\dot{\Delta\Omega}$  -derivative of turbine rotational speed perturbation

$\int \Delta\Omega dt$  -integral of turbine rotational speed perturbation

$\Omega_0$  -value of rotor speed at control design point

$\omega$  -undamped natural frequency

$\omega_d$  -damped natural frequency

$\zeta$  -partial derivative of rotor aerodynamic torque with respect to rotor collective pitch

# Contents

Summary.....	iii
Acknowledgments.....	iv
Common Symbols.....	v
Contents.....	xi
Tables.....	xiii
Figures.....	xiii
1. Introduction.....	1
1.0 Report Purpose.....	1
1.1 Wind Turbine Control Challenges.....	1
1.2 Control Objectives.....	1
1.3 Typical Industry Controllers.....	4
2. Control Design and Simulation Tools.....	7
2.0 Introduction.....	7
2.1 Control Development Process.....	7
2.2 Variable-Speed Turbine Operating Regions.....	7
2.3 Establish Control Objectives.....	9
2.4 Determine Simplified Dynamic Model.....	10
2.4.1 Linear Model.....	10
2.4.2 Issues in Wind Turbine Model Linearization.....	12
2.5 Apply Control Synthesis Tools.....	13
2.6 Perform Dynamic Simulations.....	16
3. Controls Advanced Research Turbine Baseline Control Design Examples.....	19
3.0 Goals and Contents.....	19
3.1 Description of the Controls Advanced Research Turbine.....	19
3.2 Controls Advanced Research Turbine Generator Torque Design Example.....	20
3.3 Controls Advanced Research Turbine Region 3 Baseline Pitch Control Design.....	27
3.3.1 Basic Control Design.....	27
3.3.2 Gain Scheduling.....	32
3.3.3 Anti-Windup.....	36
3.3.4 Filtering the Generator or Rotor Speed Measurement.....	38
3.3.5 Active Tower Damping Control.....	39
4. Illustrating State-Space Control Design Steps and Tools.....	43
4.0 Goals and Contents.....	43
4.1 Region 3 Collective Pitch Control Design Example.....	43
4.1.1 Control objectives.....	43
4.1.2 FAST linearization.....	44
4.1.3 Control design synthesis.....	46
4.1.4 State estimation realizable controller.....	48
4.1.5 Model Simulation with the State Estimator Controller.....	50
4.2 Region 3 Generator Torque Damping Control Design Example.....	52
4.2.1 Control objectives.....	53
4.2.2 Linear model description.....	53
4.2.3 FAST linearization.....	54
4.2.4 Control design.....	54

4.2.5 Control simulation .....	56
4.3 Independent Blade Pitch Control .....	58
4.3.1 Control objectives .....	59
4.3.2 Linear model description .....	59
4.3.3 FAST linearization .....	60
4.3.4 Control design .....	61
4.3.5 Control simulation .....	63
5. Implementing and Testing a Region 3 Rotor Collective Pitch Controller for the Controls Advanced Research Turbine .....	66
5.0 Introduction .....	66
5.1 Control Design and Implementation .....	66
5.1.1 Actuator model .....	66
5.1.2 Linear model .....	67
5.2 Simulation Tests .....	70
5.3 Further Implementation Issues .....	72
5.4 Field Test Results and Comparisons .....	74
5.5 Further Test Results .....	76
5.6 Lessons Learned .....	78
6. Using Generator Torque Control to Design, Implement, and Test a Region 2 and Region 3 Drive Train Damper .....	79
6.0 Introduction .....	79
6.1 Control Objectives and Structure .....	79
6.2 State-Space Control Design .....	79
6.2.1 Generator torque control design .....	80
6.2.2 Region 2 to 3 generator torque control transition .....	81
6.2.3 Region 3 pitch controller .....	82
6.3 Simulation Tests .....	82
6.4 Implementation Issues .....	84
6.5 Field Test Results and Comparisons .....	88
6.6 Lessons Learned .....	92
7. Conclusions and Future Work .....	93
References .....	96
Appendix A: Input Files for Simulation and Linearization .....	A1
Appendix B. MATLAB Control Design Scripts .....	B1
Appendix C. Simulating Control with Fortran Subroutines Linked with FAST .....	C1

## List of Tables

Table 3.1:	CART Parameter Values.....	22
Table 3.2:	FAST Variable-Speed Generator Model Inputs .....	24
Table 3.3:	CART Tower Parameter Values .....	40
Table 4.1:	States Contained in the Linear Model for Collective Pitch Control Design.....	43
Table 4.2:	States Contained in the Linear Model for Generator Torque Control Design....	54
Table 4.3:	States Contained in the Linear Model for Independent Pitch Control Design ...	60
Table 5.1:	Comparison between Baseline PI and FAST State-Space Controller .....	74
Table 6.1:	Preliminary Comparison of Baseline and FAST State-Space Controllers for Region 2 Operation .....	88
Table 6.2:	Comparison of Baseline and FAST State-Space Controller for Region 3 Operation.....	90

## List of Figures

Figure 1.1:	Wind turbine operating regions .....	2
Figure 1.2:	Typical plot of power coefficient versus TSR .....	3
Figure 1.3:	Typical industry controller.....	4
Figure 2.1:	Variable-speed turbine operating regions .....	8
Figure 2.2:	Variation of control input gains with pitch angle .....	12
Figure 2.3:	State estimator control diagram .....	15
Figure 2.4:	FAST wind turbine block.....	18
Figure 3.1:	The CART.....	19
Figure 3.2:	Variable-speed turbine operating regions .....	20
Figure 3.3:	Cp versus TSR and pitch for the CART .....	21
Figure 3.4:	Details of Region 2½ .....	23
Figure 3.5:	FAST simulation results (red) using the simple VS model compared to results in Figure 3.2 (blue).....	25
Figure 3.6:	Simulink model of the generator torque controller.....	26
Figure 3.7:	Simulink model of the pitch controller .....	31
Figure 3.8:	Response to a step wind input for various values of $\delta$ .....	32
Figure 3.9:	Performance at different operating points.....	33
Figure 3.10:	Variation of control input gains with pitch angle .....	34
Figure 3.11:	Simulink model of the pitch controller .....	35
Figure 3.12:	Performance at different operating points.....	36
Figure 3.13:	Use of anti-windup in preventing rotor overspeed .....	37
Figure 3.14:	Simulink pitch control model with anti-windup .....	37
Figure 3.15:	Simulink pitch control model with filtered speed.....	38
Figure 3.16:	Filter transfer function bode plot .....	39
Figure 3.17:	Simulink controller model with tower feedback.....	41
Figure 3.18:	Simulated tower bending moment in response to step wind input for various values of G .....	42
Figure 4.1:	Rotor aerodynamic torque versus blade pitch angle for various wind-speeds.....	45
Figure 4.2:	Simulink model of the realizable Region 3 pitch controller .....	49
Figure 4.3:	Simulated rotor speed with the DAC controller .....	51

Figure 4.4:	Simulated HSS torque.....	52
Figure 4.5:	Simulink model of controller-turbine FAST System, showing drive-damper realizable controller.....	56
Figure 4.6:	Regulated speed of the closed-loop system.....	57
Figure 4.7:	Simulated HSS torque excited by turbulence.....	58
Figure 4.8:	Simulated generator torque for the turbulence case.....	58
Figure 4.9:	Simulink model of controller-turbine FAST System, showing independent pitch realizable controller.....	63
Figure 4.10:	Simulated Region 3 rotor-speed using independent pitch controller.....	64
Figure 4.11:	Simulated blade-tip flap deflection excited with step wind inputs.....	64
Figure 5.1:	FAST simulated generator speed excited by step winds.....	71
Figure 5.2:	FAST simulated generator speed excited by turbulent winds.....	71
Figure 5.3:	Simulink model of the realizable controller with actuator model.....	72
Figure 5.4:	Measured hub-height wind speed during the PI control case and the FAST control case.....	74
Figure 5.5:	Measured LSS rotational speed during CART operation using the PI controller and the FAST controller.....	75
Figure 5.6:	Measured LSS torque during CART operation using the PI controller and the FAST controller.....	75
Figure 5.7:	Measured pitch rates during CART operation using the PI controller and the FAST controller.....	76
Figure 5.8:	Measured pitch rates and LSS torque during CART operation using the state-space controller.....	77
Figure 5.9:	Measured pitch rates and blade pitch during CART operation using the state-space controller.....	77
Figure 6.1:	Simulink model of the realizable controller.....	83
Figure 6.2:	FAST-simulated generator speed for the baseline and state-space controllers excited by step winds spanning Regions 2 and 3.....	84
Figure 6.3:	Measured generator torque showing stable and unstable behavior during startup for two implemented controllers.....	85
Figure 6.4:	Measured CART data during transition from Region 2 to Region 3.....	86
Figure 6.5:	Measured CART data during operation in Region 3, showing generator pitch interaction in first controller.....	87
Figure 6.6:	Measured CART data for Region 2 control for the baseline and state-space case.....	89
Figure 6.7:	Power spectral density of LSS and generator torque for the Region 2 baseline and state-space cases.....	90
Figure 6.8:	Measured CART data for Region 3 control for the baseline and state-space case.....	91
Figure 6.9:	Power spectral density of LSS and generator torque for the Region 3 baseline and state-space cases.....	91
Figure C.1:	Simulated rotor speed during PI control simulating with the control subroutine and the Simulink control model.....	C-2

# 1. Introduction

## 1.0 Report Purpose

The purpose of this report is to give wind turbine engineers information about designing, implementing, and testing advanced control systems for wind turbines. We want to illustrate use of available control design tools as well as the steps involved in designing, implementing, and field testing advanced controllers.

## 1.1 Wind Turbine Control Challenges

Wind turbines are complex, nonlinear, dynamic systems forced by gravity, stochastic wind disturbances, and gravitational, centrifugal, and gyroscopic loads. The aerodynamics of wind turbines are nonlinear, unsteady, and complex. Turbine rotors are subjected to a complicated 3-D turbulent wind inflow field that drives fatigue loading.

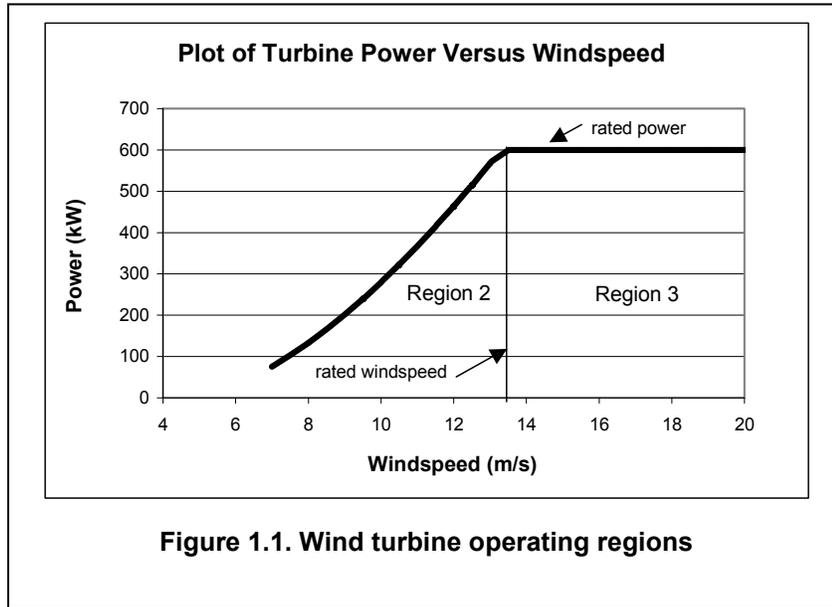
Wind turbine modeling is complex and challenging. Accurate models must contain many degrees of freedom (DOFs) to capture the most important dynamic effects. The rotation of the turbine adds complexity to the dynamics modeling. Off-the-shelf commercial software is not adequate for wind turbine dynamics modeling; specialized dynamic simulation codes modeling of all these nonlinear effects is required.

Design of control algorithms for wind turbines must account for these complexities. These algorithms must capture the most important turbine dynamics without being too complex and unwieldy.

## 1.2 Control Objectives

A wind turbine control system consists of sensors, actuators, and a system that ties these elements together. A hardware or software system processes input signals from the sensors and generates output signals for actuators. The main goal of the controller is to modify the operating states of the turbine to maintain safe turbine operation, maximize power, mitigate damaging fatigue loads, and detect fault conditions.

In typical wind turbines, there are different regions of operation (see Figure 1.1). In Region 2, below rated wind speed, the goal is to maximize turbine power. In Region 3, above rated wind speed, the goal is to maintain turbine power at a constant level (rated power), to limit turbine loads and generator power. Other regions of operation include startup (Region 1) and machine shutdown (not shown).

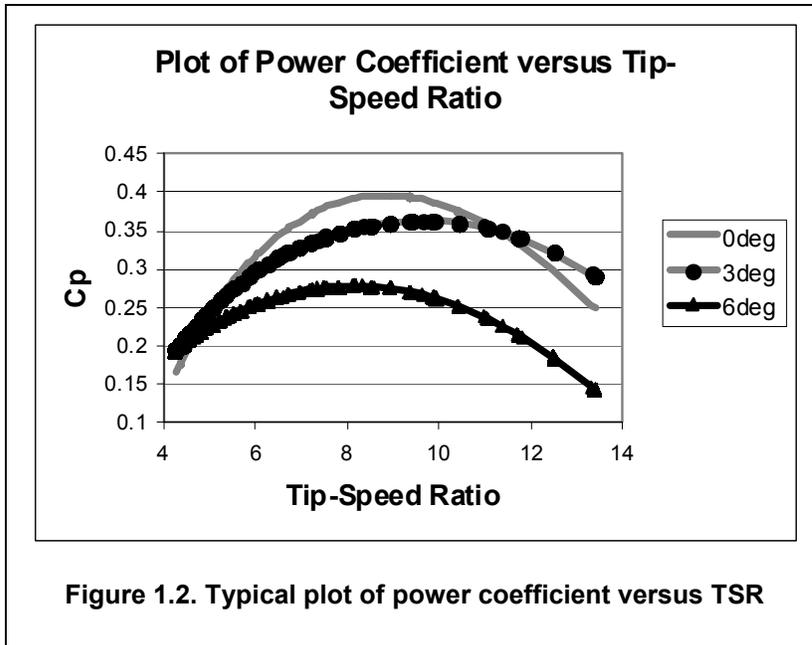


**Figure 1.1. Wind turbine operating regions**

Today's wind turbines employ different control actuation and strategies to achieve these goals. Some turbines achieve control through passive means, such as in fixed-pitch, stall control machines. In these machines, the blades are designed so that power is limited in Region 3 through blade stall. No pitch mechanism is needed in these machines. In Region 2, generator speed is fixed. Typically, control of these machines involves only starting and stopping the turbine.

Rotors with adjustable pitch are often used in constant-speed machines to provide better control of turbine power, than is possible with blade stall. Blade pitch can be regulated to provide constant power in Region 3. The pitch mechanisms in these machines must be fast, to provide good power regulation in the presence of gusts and turbulence.

Operating the turbine at constant turbine rotational speed in Region 2 (through the use of synchronous or induction generators) has consequences for the power output of the machine. To maximize power output in Region 2, the rotational speed of the turbine must vary with wind speed to maintain a constant tip-speed ratio (TSR). Figure 1.2 shows the rotor power coefficient  $C_p$  versus TSR for a typical turbine for different blade pitch angles. In each curve is a maximum  $C_p$  at a certain TSR. For fixed-speed machines, this means that only at a single wind-speed will  $C_p$  be optimum. For all other wind speeds, the turbine operates at a nonoptimum  $C_p$ .



Most large commercial wind turbines allow the rotational speed of the machine to vary with wind speed (the variable-speed machine). This allows the turbine to operate at near optimum  $C_p$  and maximize power over a range of wind speeds. Blade pitch control is used in Region 3 to limit power.

Other control objectives include changing the operating state of the turbine, such as starting and stopping the machine. These functions are performed by the supervisory control system. For example, to start up a variable-speed machine, the mechanical parking brake is released, and the blade pitch angle is reduced from full feather (90-degree pitch angle) to a value that allows the aerodynamic torque to accelerate the rotor from rest. During machine shutdown, the blade pitch is increased rapidly from the run position to full feather, and a mechanical parking brake is engaged.

Most large commercial wind turbines employ active yaw control to orient the machine into the wind. A yaw error signal from a nacelle-mounted wind direction sensor is used to calculate a control error. The control signal is usually just a command to yaw the turbine at a slow constant rate in one direction or the other. The yaw motor is switched on when the yaw error exceeds a certain amount and is switched off when the yaw error is less than some prescribed amount. In (1), independent pitch control was studied as a means of controlling yaw for a large three-bladed turbine.

Another control objective is fault diagnosis. The fault diagnosis capabilities of the controller must include monitoring for component failures, including sensor failures, operation beyond safe operating limits, grid failure or grid problems, and other undesirable operating conditions (such as high vibrations).

We will not describe supervisory control, yaw control, or fault diagnosis in this report, but will describe generator torque control in Region 2 and blade pitch control in Region 3.

We now look at typical industry turbine control.

### 1.3 Typical Industry Controllers

Typical large commercial wind turbines are variable-speed machines, and control generator torque in Region 2 to maximize power and pitch in Region 3 to maintain turbine power. The controls for such machines are typically designed using simple classical control design techniques such as proportional-integral-derivative (PID) control for pitch regulation in Region 3 (2). A typical controller for such a machine is shown in Figure 1.3. Generator torque is controlled using  $Q_{gen} = k\Omega^2$  in Region 2 as shown in the upper control loop of the figure. The measured control input is usually generator-speed. In Region 2, blade pitch is usually held constant.

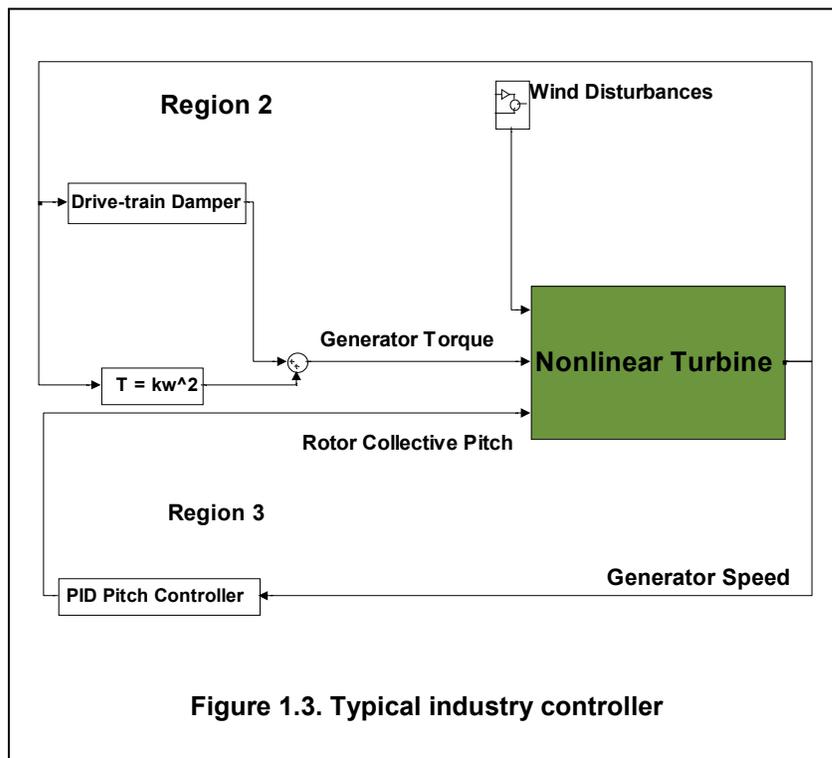


Figure 1.3. Typical industry controller

In Region 3, blade pitch is controlled to maintain constant turbine speed while generator torque is held constant. Classical PID control design techniques are typically used to design the blade pitch controller. Advanced controls can be used to improve the Region 2 energy capture.

Refinements to the Region 2 generator torque control can lead to enhanced energy capture. Fingersh and Johnson (3) reported improved energy capture using a variation of

the Region 2 baseline controller approach named the optimally tracking rotor control. Johnson (4) reported using an adaptive control approach to improve energy capture in Region 2.

An additional goal of control is to mitigate turbine structural dynamic loads. One way to reduce dynamic loads is to design controls that actively damp turbine components. In commercial turbines, an additional generator torque control loop in Region 2 is often used to actively damp the drive train torsion mode of the turbine (see Figure 1.3). In Region 3, classical control design methods have been used to design controllers to add damping to the tower's first fore-aft (f-a) mode with blade pitch (2,5). The pitch control to actively damp tower f-a motion is usually implemented as an additional single input single output (SISO) control loop to the basic speed control loop in Region 3.

Another way to mitigate turbine loads is through independent pitch control, where each blade is pitched independently. In (6) both a classical control and a multivariable control approach were used to design independent pitch controls to mitigate the effects of asymmetric wind distributions across the rotor disk. In the classical design approach, two separate SISO control loops were used to mitigate the tilt- and yaw- oriented loads in the fixed frame with independent pitch. In (7), this work was extended with alternative sensors to measure the asymmetric loading on the rotor. Good results were obtained when suitable sensors were used.

A disadvantage of classical control methods is that multiple control loops must be used to add active damping to several flexible turbine modes or to mitigate the effects of asymmetric wind variations with independent pitch. If these controls are not designed with great care, the control loops interfere with each other and destabilize the turbine. The potential to destabilize the turbine increases as turbines become larger and more flexible, and the degree of coupling between individual control loops increases.

Using all the available turbine actuators in a single control loop to maximize load-alleviating potential would be advantageous. Advanced multi-input multi-output (MIMO) multivariable control design methods, such as those based on state-space models, can be used to meet these objectives and use all the available actuators and sensors in a reduced number of control loops. In (8), a multivariable approach was used to design both an independent pitch controller to mitigate the effects of asymmetric wind disturbances across the rotor disk as well as a collective pitch controller to perform Region 3 speed regulation and active tower f-a damping. The independent pitch control was performed in a separate control loop from the speed regulation and tower f-a damping control. Even though two separate control loops were used in (8), the multivariable control design approach resulted in fewer control loops compared to classical control design methods. In (9), loads were reduced significantly with the use of state-space periodic controllers in Regions 2 and 3.

The real proof of control performance is obtained when controls are implemented and tested in the field. In (10), multivariable controls were tested on the Controls Advanced Research Turbine (CART) at the National Renewable Energy Laboratory (NREL). In (11), state-space controls for speed regulation and drive train damping were implemented

and tested. In (12), generator torque was used to add active damping to the drive-train torsional mode in both regions 2 and 3. In (13) multivariable MIMO controls were implemented and tested for active tower damping, with good load alleviation results.

This report provides wind turbine engineers information and examples of the design, testing through simulation, field implementation, and field testing of advanced wind turbine controls. It is Part I in a two-part series that details advanced control design, implementation, and test results. Part I highlights the control development process, from forming control objectives, to designing the controller, to testing the controller through analytical simulation, to field implementation and initial field testing. Part II (to be completed later) will include a detailed comparison of results from advanced load alleviating state-space controllers to test results from baseline controllers without load alleviation. The purpose of Part II is to demonstrate through rigorous testing the load mitigating potential of the advanced state-space controllers compared to the baseline control.

This report will be organized as follows.

- Chapter 2 examines and outlines the control development process. It includes discussions of setting control objectives, the control development process, and the control design and simulation tools that will be used in later sections of this report. The primary focus of this chapter will be to develop controls for a variable-speed wind turbine.
- Chapter 3 illustrates the design and simulation of baseline pitch and generator torque controllers (see Figure 1.3) that are still widely used in industry. The goal of this chapter is to illustrate use of the control design and simulation tools.
- Chapter 4 illustrates the design and simulation of advanced state-space controllers with three control design examples:
  - A Region 3 rotor collective pitch controller for speed regulation and drive train torsional damping
  - A Region 3 generator torque controller for drive train torsional damping
  - A Region 3 independent blade pitch controller for rotor load alleviation.
- Chapters 5 and 6 illustrate field implementation and tests of two advanced state-space controllers. The goals of these chapters are to illustrate the problems and lessons learned in advanced controls field implementation and testing.
- Conclusions and state future work. We will not give a detailed comparison of test results from the advanced controllers to test results from baseline torque and pitch controllers. This will be performed in a future Part II report.

## 2. Control Design and Simulation Tools

### 2.0 Introduction

Chapter 1 described the challenges of modern control design for current and future wind turbines. Controls must be designed to meet multiple control objectives for these complex nonlinear systems. The control designer must strike a balance between designing the controller to be complex enough to meet the intended control objectives and simple enough that understanding results and debugging is not too difficult. The control designer should start with simple controls and add needed complexity in steps instead of attempting to design, test, and implement a highly complex MIMO controller.

### 2.1 Control Development Process

Developing advanced controls for wind turbines is a process. Several steps are needed to design, test, and implement a wind turbine control system:

1. Determine the control objectives.
2. Develop a simplified dynamic model from which to design the controller.
3. Apply specialized control synthesis tools.
4. Perform dynamic simulations with the controller in the loop to test closed-loop system performance.
5. Repeat steps 1 through 4, pending results of step 4.
6. Implement control algorithms in controller software for field testing on a real turbine.
7. Bench test controller hardware.
8. Test the controller in a field test turbine.
9. Repeat steps 1 through 8 until desired control performance is achieved in field tests.

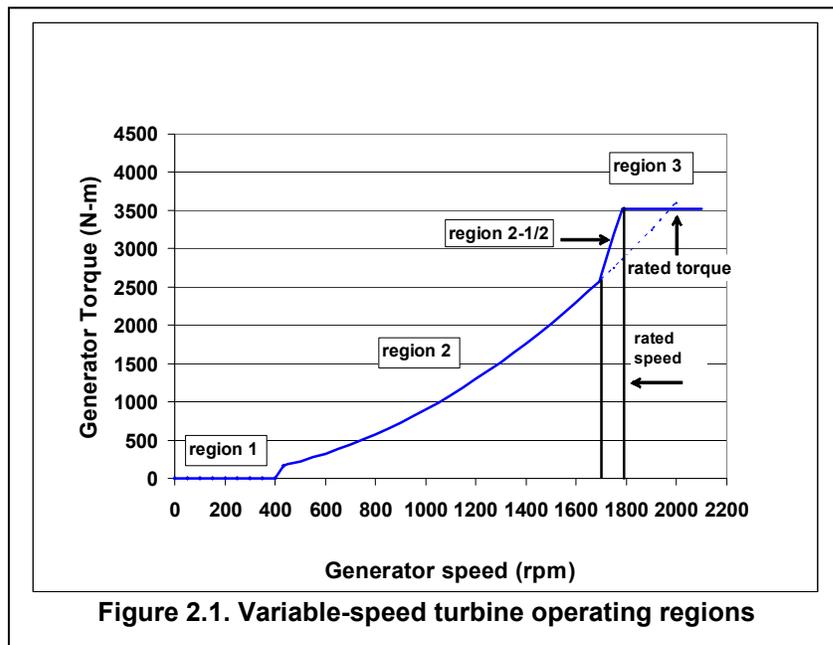
Each step is detailed and the tools are outlined in the following sections. We focus our controls development on the variable-speed turbine; we now discuss regions of operation for this type of wind turbine in more detail.

### 2.2 Variable-Speed Turbine Operating Regions

Typical variable-speed wind turbines have different regions of operation (Figure 2.1) (3). In this figure, we see generator torque as a function of generator speed (measured on the high-speed end of the gearbox) for the turbine described in Chapter 3. Turbine start-up

occurs in region 1, for generator-speeds between 0 and 430 revolutions per minute (rpm). In this region the generator torque is zero. The startup sequence may look like this: once the turbine supervisory control system decides that the wind speed is sufficient for startup, the pitch angle of the blades (using pitch actuators or motors) is changed from full feather (the pitch value when the machine is stopped, at approximately 90 degrees) to a pitch angle when the turbine operates in Region 2 (typically called the run-pitch position). For the machine that will be described in Chapter 3, this value is  $-1$  degree. This small pitch angle results in sufficient aerodynamic torque to overcome bearing friction allowing the rotor to start up from rest. Once the generator speed has accelerated to 430 rpm (this value is turbine dependent), the generator torque is switched on and power is produced normally. Now the turbine is operating in Region 2.

The Region 2 torque curve in Figure 2.1 intersects rated torque (where the dotted blue line intersects rated torque at approximately 2000 rpm) at a rotor speed that is significantly higher than rated speed. It would be nice to operate the turbine on the Region 2 optimum  $C_p$  curve up to where it intersects rated torque, but operation of the turbine at these high rotor speeds (above rated speed) would result in a high blade tip speed and unacceptable noise emissions. Our desire is to allow the turbine to reach rated torque at a lower rotor speed, typically at or close to rated speed (1800 rpm for this example).



Power production in Region 2 continues as long as the generator speed is 430 to 1700 rpm (these values are turbine dependent). In this region blade pitch is held constant at its run-pitch value, and generator torque control is used to vary the speed of the turbine to maintain constant TSR corresponding to optimum  $C_p$ , thus maximizing energy capture. We will show additional details of generator torque control in this region for a specific turbine in Chapter 3.

If we did not insert this new region, and maintained Region 3 generator torque at the point where the Region 2 torque curve intersects rated speed (1800 rpm), the generator torque would be significantly lower (approximately 2900 Newton-meters [N-m]) than rated torque (approximately 3500 N-m); i.e., the power production in Region 3 would be too low. To ensure that the machine has the desired rated power, Region 3 must have rated torque and a new region must be inserted to connect Region 2 to Region 3. This new region (3) begins at a rotor speed lower than rated speed and reaches rated torque at rated speed or slightly below, to ensure a smooth transition. We will show details of the generator torque controls in these operating regions for a particular variable-speed machine in Region 3. We will also show simulation with these generator torque controls in these operating regions. We will not be concerned with Region 1 or shutdown, since this report focuses on control designs for Regions 2, 2½, and 3.

In Region 3, generator torque is simply held constant at rated torque. In some machines Region 3 generator torque control is set to maintain constant power instead of constant torque, with generator torque inversely proportional to rotor speed (14). If rotor speed is tightly controlled to rated speed in Region 3, this type of control will be almost identical to setting generator torque to maintain constant torque in Region 3.

As far as pitch control is concerned, the pitch is held constant (at its run-pitch value) in Region 2, because the generator torque is being controlled in such a way as to allow the machine to operate on the optimum Region 2 torque curve. No pitch control is necessary in this region. The real pitch control takes place in Region 3.

The objective of pitch control in Region 3 is to control pitch and regulate rotor speed to the rated speed set point. The pitch control must be fast, to account for variations in wind speed. Another issue in designing wind turbine pitch control is excitation of flexible modes of the turbine. We want to perform fast pitch control to regulate speed in the presence of wind speed disturbances without exciting flexible modes. We will examine these issues further in subsequent chapters. Another issue will be smooth transition of pitch from Region 2 to Region 3.

Now that we have described some details of the operating regions of a typical variable-speed turbine, we can describe formulation of control objectives.

### **2.3 Establish Control Objectives**

The first step is to define the objectives of the controller. This depends on the turbine configuration. For example, we might want to use rotor collective pitch to regulate turbine speed in Region 3. We might want to maximize power in Region 2. Another necessary objective is to maintain stable closed-loop behavior over the full range of operating conditions. We might also want to add active damping to low-damped flexible modes. We must also perform this control within the stipulated actuator limits.

For other turbine configurations, such as fixed-speed machines, the control objectives may be different. In Region 3, we may want to use pitch control instead of rotor speed to limit power. For the control design examples of this report, we will focus on controls for

the variable-speed turbine, specifically generator torque in Region 2, blade pitch in Region 3, and control transition between Regions 2 and 3 (Region 2½).

## 2.4 Determine Simplified Dynamic Model

The second step is to develop simplified dynamic models for use in control design. How complex should the model be? If it is too simple, important dynamics will be excluded, leading to possible design of an unstable closed-loop system or a control algorithm that does not perform as intended. On the other hand, an overly complex model will lead to a control system that is too complicated and difficult to design, implement, test, and debug. The simplified model used for control design should depend on the control objectives we identified in the first step. For example, if our sole objective in Region 3 is to use blade pitch to regulate turbine speed, only a very simple model is required. If we also want to add active damping of flexible turbine modes, additional modeling detail will be needed.

Different codes model turbine dynamics using different methods. Large multibody dynamics codes (15) divide the structure into numerous rigid body masses and connect these parts with springs and dampers. This approach leads to dynamic models with hundreds or thousands of DOFs. The order of these models must be greatly reduced to make them practical for control design synthesis. In addition, these nonlinear models need to be linearized to apply linear control theory.

In another approach, an assumed modes method is used to discretize the wind turbine structure. With this approach, the most important turbine dynamics can be modeled with just a few DOFs. Designing controllers based on these models is much simpler, and captures the most important turbine dynamics, leading to a stable closed-loop system. Debugging these controllers and the models from which they are designed is much easier, leading to faster design and development time.

Such a code is the FAST dynamics code (16). This code is useful in designing and simulating control systems (17). It has recently been modified to produce linear state-space models of turbine systems and has been extensively tested and validated (16).

### 2.4.1 Linear Model

Most of the simplified models that we will use in our control designs will be linear time-invariant state-space models. FAST can be used to produce these linear models, which take the form:

$$\underline{\Delta \dot{x}} = A\underline{\Delta x} + B\underline{\Delta u} + B_d \underline{\Delta u}_d \tag{2.1}$$

$$\underline{\Delta y} = C\underline{\Delta x} + D\underline{\Delta u} + D_d \underline{\Delta u}_d .$$

where

$\underline{\Delta x}$  is the state vector,

$\underline{\Delta u}$  is the control input vector,

$\underline{\Delta u_d}$  is the disturbance input vector,

$\underline{\Delta y}$  is the control (or measured) output,

$A$  represents the state matrix,

$B$  the control input gain matrix,

$B_d$  the disturbance input gain matrix,

$C$  relates the measured output  $\underline{\Delta y}$  to the turbine states,

$D$  relates the measured output to the control input,

$D_d$  relates the measured output to the disturbance states.

$\underline{\Delta \dot{x}}$  represents the time derivative of  $\underline{\Delta x}$ ,

$\underline{\Delta x}$ ,  $\underline{\Delta \dot{x}}$ ,  $\underline{\Delta y}$ ,  $\underline{\Delta u}$ , and  $\underline{\Delta u_d}$  (perturbed values) represent small perturbations from the calculated operating point values  $\underline{x_{op}}$ ,  $\underline{\dot{x}_{op}}$ ,  $\underline{y_{op}}$ ,  $\underline{u_{op}}$ , and  $\underline{u_{dop}}$  (16).

The beauty of FAST is that different DOFs can be switched on or off. This means that simplified linear models that contain a subset of the total DOFs modeled with FAST can be extracted.

The linearization routines follow a procedure similar to that used by the Symbolic Dynamics (SymDyn) code, which is a controls-oriented HAWT analysis tool developed by researchers at NREL (18). The structural model of FAST, however, is of higher fidelity than that of SymDyn. The linearization process consists of two steps: (1) compute a periodic steady-state operating point condition for the DOFs, and (2) numerically linearize the FAST model about this operating point to form periodic state matrices. The output state matrices can then be azimuth-averaged for nonperiodic or time-invariant control design.

The first step in this linearization process is to determine a steady-state operating point about which FAST calculates linearized state matrices. An operating point is a set of values for the DOF displacements, velocities, accelerations, control inputs, and disturbance inputs that characterize a steady-state condition of the wind turbine. For a wind turbine operating in steady winds, this solution is periodic, that is, the operating point values depend on the rotor azimuth position. For control design we typically generate periodic linear models at several points around the rotor disk and average them to obtain a state-space model averaged with respect to blade azimuth position. For more details on the procedure for calculating this steady-state operating point, see (16).

Once a periodic steady-state solution has been found, FAST numerically linearizes the complete nonlinear aeroelastic model about this operating point. Since the operating point is periodic with the rotor azimuth position, the linearized representation of the model is also periodic. For time-invariant linear control, a more accurate model is obtained if we output the linearized model at a number of azimuth steps and then average the resulting matrices together, rather than using one azimuth location. We use a special averaging tool described in (16).

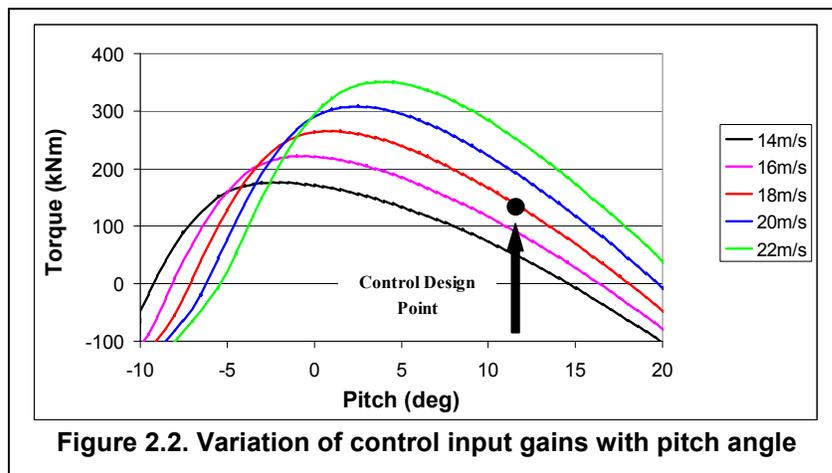
For all the controls illustrated in this report, we use linear time invariant control design methods (19). We plan to use linear models generated from FAST to upgrade the capability to perform periodic control as in (20).

### 2.4.2 Issues in Wind Turbine Model Linearization

We will be generating various linear models, as we will show in Chapter 4 on State-Space Control Design. Before we do that, it is wise to point out some of the issues encountered when linearizing a nonlinear wind turbine model.

An important issue is the variation of control input gain and disturbance input gain matrices  $B$  and  $B_d$  with turbine operating point. We will show that these gain matrices are related to the partial derivative of rotor aerodynamic torque with respect to blade pitch and wind speed. Figure 2.2 shows a plot of the rotor aerodynamic torque versus blade pitch angle for various wind speeds, for the machine to be described in Chapter 3. The control input gain matrix  $B$  is directly related to the slopes of these curves, at any particular value of pitch and wind speed.

To design controls in Region 3, we will typically choose a control design point somewhere in Region 3. In Figure 2.2, this control design point is at a pitch angle of 11 degrees and a wind speed of 18 meters/second (m/s). At this turbine design point the torque versus pitch angle curve has a large negative slope, meaning that the control input gain matrix is nonzero. Performing the first step in the FAST linearization process, determining a steady-state operating point about which FAST calculates linearized state matrices proceeds rapidly without any problems. We will show example cases highlighting the FAST linearization process and code inputs in Chapter 4 and in the Appendices.



If we choose a point for model linearization that has a smaller pitch angle, the slope of the torque versus blade pitch angle decreases (choose the 18 m/s wind speed curve; at a pitch angle of 1 degree, the slope of this curve is approximately zero). If we choose this operating point to be a linearization point, convergence to a trim solution will be almost impossible and may become unstable. Because this represents an unstable equilibrium position, the slope of the curve at this point is zero and the pitch input gain matrix is zero. A linearization point that will lead to a convergence of the trim solution to a stable equilibrium point must be chosen carefully.

Variation of the disturbance input gain matrix  $B_d$  is less important, as the value of this matrix is usually positive over a wide operating range.

Another issue with this variation of control input and disturbance input gain matrices is the performance of a control system with turbine operating point. If the controller is designed at an operating point midway between the lowest and highest wind speeds of Region 3 (such as the linearization point shown in Figure 2.2), we can expect the performance of the controller to be different than the designed controller at other turbine operating points. The performance will be good only for small perturbations of the turbine operating point from the control design (or linearization) point.

Thus, testing the control performance through simulation is important. We want to test the controller for a range of turbine operating points away from the control design point. The first step, however, is to design the control correctly at the chosen point. We now discuss control design and synthesis tools.

## 2.5 Apply Control Synthesis Tools

Once the simplified dynamic model has been generated, we apply the control synthesis tools. For this work, we will rely on control synthesis tools that use and interface with MATLAB's Control System Toolbox (27).

The complexity of the synthesis tools depends on the control objectives and complexity of the dynamic model used to design the controller, as already described. It also depends on the control method used for control design. If we are designing a simple proportional-integral (PI) controller for pitch regulation in Region 3, simple tools can be used. If we are designing a complex MIMO controller, more complex synthesis tools will be needed.

We will depend heavily on the use of full state-feedback for active damping of flexible turbine modes. For more information on this control design method, refer to (17). Basically, we formulate the control law as a linear combination of the system states:

$$\Delta \underline{u}(t) = G \Delta \underline{x}(t) \tag{2.2}$$

If the system consisting of  $(A, B)$  in (2.1) is controllable (17), this feedback law can be used to place the poles of this system arbitrarily in the complex plane. This allows us to place plant poles to improve system response and improve damping (17). A big part of the control design is to calculate the gain matrix  $G$  used in the control law.

To calculate the gain matrix  $G$ , we can use either pole placement (17) or linear quadratic regulation (LQR) (19). With LQR, we find a unique linear feedback control signal that will minimize the following quadratic cost function.

$$J = \int_0^{\infty} (\Delta \underline{x}(t)^T Q \Delta \underline{x}(t) + \Delta \underline{u}(t)^T R \Delta \underline{u}(t)) dt \quad (2.3)$$

where,

$\Delta \underline{x}(t)$  represents the system states,

$\Delta \underline{u}(t)$  represents the control inputs

$Q$  contains weightings for the states, and

$R$  contains weightings for actuator usage.

Fast state regulation and low actuator use are competing objectives; therefore the  $Q$  and  $R$  weightings allow us to trade off performance objectives with actuator use.

The gain matrix  $G$  can then be calculated as:

$$G = -R^{-1} B^T P,$$

where

$R^{-1}$  is the inverse of  $R$ ,

$B^T$  is the transpose of  $B$  in (2.1),

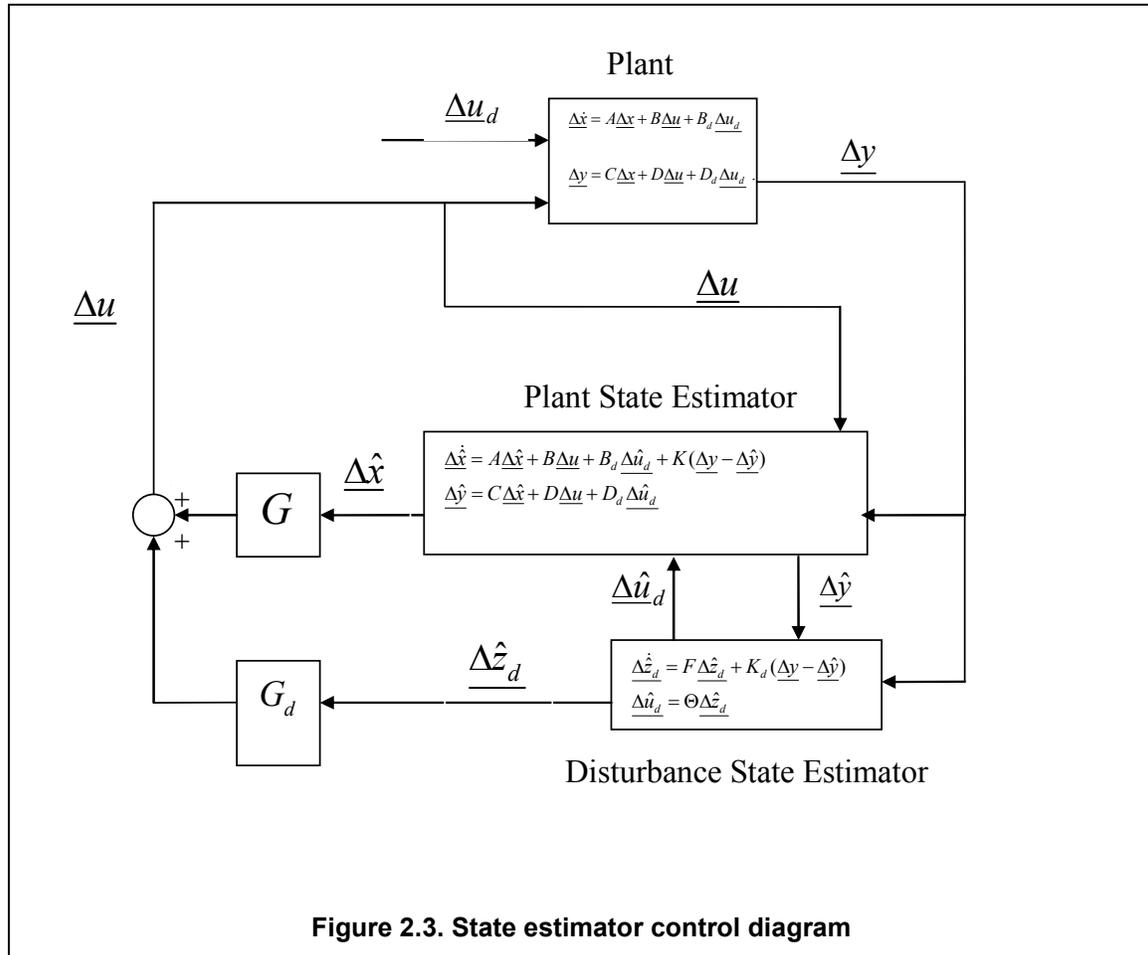
$P$  is the solution to the Riccati equation (19):

$$A^T P + P A + Q = P B R^{-1} B^T P. \quad (2.4)$$

The MATLAB routine LQR determines the values for  $G$ ,  $P$ , and  $E$  where  $E$  is the set of eigenvalues of the closed loop system. Through the feedback control law [Equation (2.2)] the new state matrix is  $A + B G$  (17). We will use special MATLAB scripts that have already been written to perform either pole placement or LQR.

If we were to use full state feedback as the final control design, we would have to measure every state contained in the linear model described by (2.1). Most commercial turbines are not instrumented to the extent needed to measure all these states, especially as the order of the model increases. Observability allows us to use state estimation to estimate the states contained in the linear model based on just a few turbine measurements (17). The resulting controller using state estimation is called a realizable controller (17, 20).

Figure 2.3 shows a control diagram for the state estimator controller. There are two inputs to this controller: the measurement signal  $\Delta y$  and the control input  $\Delta u(t)$ . For more information on state estimation for wind turbines, see (17). We will see examples of state estimator controllers and their Simulink models in Chapter 4.



So far we have ignored the effects of wind disturbances  $\Delta u_d$ . We know that wind turbines must operate in a highly turbulent wind environment. Turbulent winds cause fluctuations in the blade aerodynamic forces, and thus influence the power, torque, and cyclic loads of the machine. We need an approach that counteracts or accommodates these disturbances and permits full-state feedback and state estimation.

Disturbance accommodating control (DAC) is a way to reduce or counteract persistent disturbances (21). Its basic idea is to augment the usual state estimator-based controller to recreate disturbance states via an assumed-waveform model; these disturbance states are used as part of the feedback control to reduce (accommodate) or counteract any persistent disturbance effects (21).

The disturbance model is assumed in the state-space form:

$$\begin{aligned}\dot{\underline{z}}_d(t) &= F\underline{z}_d(t) \\ \underline{u}_d(t) &= \Theta\underline{z}_d(t)\end{aligned}\tag{2.5}$$

Steps to synthesizing controllers using DAC are:

1. Assess controllability of the system  $(A, B)$  to allow pole placement. If the system is controllable, choose plant poles to enhance damping and improve system response as desired.
2. Calculate gains  $G$  to give the desired poles chosen in step 1. Poles can be placed through either pole placement or LQR.
3. Form the feedback law  $\underline{u}(t) = G\underline{x}(t) + G_d\underline{z}_d(t)$ . Choose the disturbance gain  $G_d$  to exactly cancel wind speed disturbances if possible; otherwise, choose this gain to mitigate the disturbance effects as much as possible.
4. Calculate the augmented state matrices  $(\bar{A}, \bar{B}, \bar{C})$  and assess the observability of  $(\bar{A}, \bar{C})$  (17).
5. If observability is achieved, choose state estimator poles (including wind disturbance states) to achieve the desired behavior.
6. Now that the plant and state estimator gain matrices have been calculated, a state-space model of the controller alone (or an equivalent transfer function) can be determined.

Here,  $\bar{A}$ ,  $\bar{B}$ , and  $\bar{C}$  represent the normal  $A$ ,  $B$ , and  $C$  matrices augmented with the disturbance states, as described in (17).

We will use special scripts and files to perform these control design steps in MATLAB, which is easy to check for controllability and observability and to perform pole placement or LQR.

Once we synthesize the controller, we perform dynamic simulations with the controller in the loop to test closed-loop system performance.

## 2.6 Perform Dynamic Simulations

The next step in the control development process is to perform dynamic simulations with the controller inserted into the loop. We want to simulate a variety of operating conditions to test closed-loop performance. This step is crucial before proceeding to implementing and testing the controller on the real machine in the field.

These test simulations must be performed as rapidly as possible to decrease the control design time. The same issues apply to simulation as apply to the models we use for control design. We want to simulate with a turbine model that captures the most

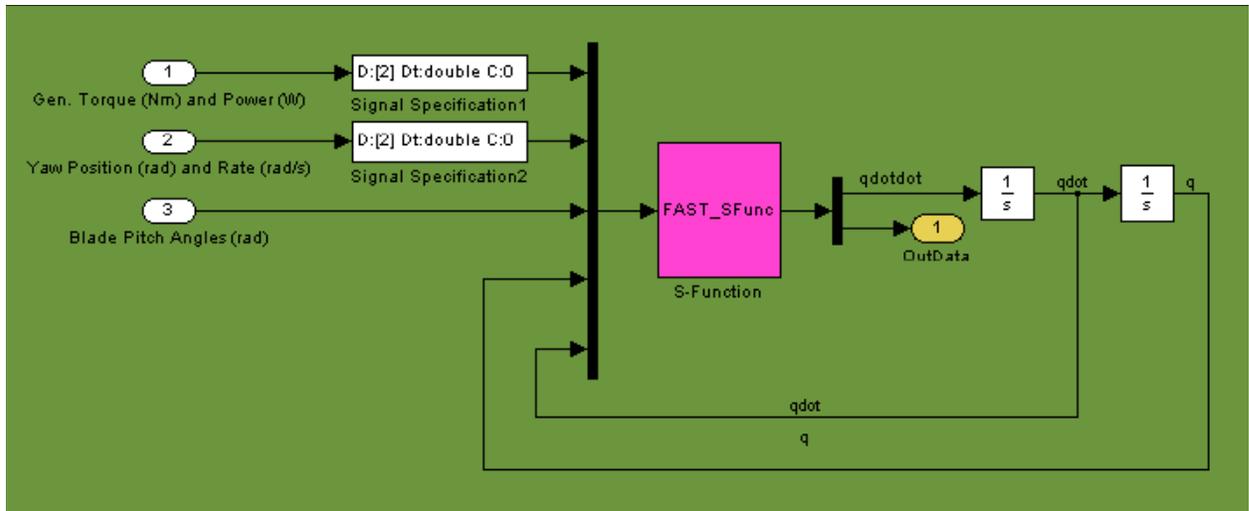
important turbine dynamics and is not overly complex. An assumed modes approach allows us to model the most important turbine dynamics with relatively few DOFs and low complexity. Execution time with these codes is much faster than with large multipurpose dynamics codes. We choose the FAST dynamics code (16) for our closed-loop simulation tool.

FAST uses Kane's method (22) to set up equations of motion, which are solved by numerical integration. The implemented method makes direct use of the generalized coordinates, eliminating the need for separate constraint equations. FAST uses the AeroDyn subroutine package developed by Windward Engineering to generate aerodynamic forces along the blade (23).

The FAST code models the wind turbine as a combination of rigid and flexible bodies. For example, two-bladed, teetering-hub turbines are modeled as four rigid and four flexible bodies. The rigid bodies are the Earth, nacelle, hub, and optional tip brakes (point masses). The flexible bodies include blades, tower, and drive shaft. The model connects these bodies with several DOFs. These DOFs can be turned on or off individually in the analysis by simply setting a switch in the input data file.

The crucial step is simulating with a FAST model of the turbine with the controller included in the loop. There are two methods of inserting the active controls into the loop: through user-defined control subroutines that are compiled and linked with FAST during creation of the executable, and through a MATLAB/Simulink/FAST interface (16).

Simulink is a popular simulation tool for control design that is distributed by The Mathworks, Inc. in conjunction with MATLAB. Simulink can incorporate custom Fortran routines in a block called an S-Function. The FAST subroutines have been linked with a MATLAB standard gateway subroutine to use the FAST equations of motion in an S-Function that can be incorporated in a Simulink model. This introduces tremendous flexibility in wind turbine controls implementation during simulation. Generator torque control, nacelle yaw control, and pitch control modules can be designed in the Simulink environment and simulated while making use of the complete nonlinear aeroelastic wind turbine equations of motion available in FAST. The wind turbine block (Figure 2.4) contains the S-Function block with the FAST equations of motion. It also contains blocks that integrate the DOF accelerations to achieve velocities and displacements. Thus, the equations of motion are formulated in the FAST S-function but solved using one of the Simulink solvers.



**Figure 2.4. FAST wind turbine block**

The interface between FAST and Simulink is similar to the interface developed for the SymDyn code (18). The structural model of FAST, however, is of higher fidelity than that of SymDyn. For more details about the FAST-Simulink interface, see the FAST User's Guide (16), pages 35-37. An example model and simulation using FAST-Simulink will be shown in Chapter 3 for the baseline blade pitch and generator torque control designs to be described there.

The control designer can rarely go through these steps just once. Most often, the steps listed in Section 2.1 form an iterative process. The control designer may find that when simulations are performed, the system is stable in closed-loop with just a few turbine modes switched on during simulation. When simulated with additional DOFs, the system may be unstable. The complete process, beginning with establishing control objectives, producing a simplified linear model, control system synthesis, and performing detailed simulations will have to be repeated.

Steps 6 to 9 in Section 2.1 are involved with implementing the controller in the field test turbine software and performing field testing of the controller. These steps have their own complexities and issues. We will delay detailed descriptions of these steps until Chapters 5 and 6 of this report, involved with field implementation and tests of two state-space control examples.

Two of the most common forms of turbine control are full-span blade pitch control and generator torque control. We illustrate these control simulations in Chapter 3, where we show the design and simulation of a baseline PID Region 3 pitch controller and a baseline Region 2 generator torque controller. We will give examples of the design and simulation of advanced state-space controllers in Chapter 4. We now show design and simulation of these baseline generator torque and blade pitch controls in Chapter 3.

## 3. Controls Advanced Research Turbine Baseline Control Design Examples

### 3.0 Goals and Contents

The goals of this chapter are to illustrate the design of a simple baseline controller, such as that used in industry and described in Chapter 1. We illustrate the design of a Region 2 torque controller and a Region 3 pitch controller. Another goal of this chapter is to illustrate use of FAST for simulating the closed-loop system. The intent here is to illustrate use of the tools before proceeding on to more advanced control designs.

Since we focus on controls for the CART, we briefly describe this machine.

### 3.1 Description of the Controls Advanced Research Turbine

The CART (Figure 3.1) is a two-bladed, teetered, upwind, active-yaw wind turbine. This machine is used as a test bed to study aspects of wind turbine control technology for medium- to large-scale machines (3).

The CART is variable speed, and each blade can be independently pitched with its own electromechanical servo. The pitch system can pitch the blades up to 18 degrees per second (deg/s) with pitch accelerations up to 150 deg/s/s. The squirrel cage induction generator with full power electronics can control torque from minus rating (motoring) to plus rating (generating) at any speed. The torque control loop has a high rated bandwidth of 500 radians per second (rad/s).



Figure 3.1. The CART

Rated electrical power (600 kilowatts at a low-speed shaft [LSS] speed of 41.7 rpm) is maintained in Region 3 in a conventional variable-speed approach. Power electronics are used to command constant torque from the generator and full-span blade pitch controls the rotor speed.

The machine is equipped with a full complement of instruments that gather meteorological data at four heights. Blade-root flap and edge-strain gages, tower-bending gages, and LSS and high-speed shaft (HSS) torque transducers gather load data. Accelerometers in the nacelle measure the tower's f-a and side-side (s-s) motion. Absolute position encoders gather data on pitch, yaw, teeter, LSS, and HSS positions.

These data are sampled at 100 Hz. The custom-built control system collects these data and controls the turbine at a control loop cycle rate of 100 Hz. This system is personal computer based and very flexible.

### 3.2 Baseline Generator Torque Design Example

The control objective here is to use generator torque to maintain optimum TSR in Region 2, thus maintaining peak  $C_p$  and maximizing power. In Region 2 we want to hold pitch constant. For this baseline torque controller, we do not want to satisfy any other control objectives such as mode damping or load mitigation.

We briefly review the operating regions of the variable-speed turbine (already discussed in Section 2.2) (Figure 3.2). In below-rated wind speeds (Region 2), blade pitch is held constant and generator torque control is used to vary the speed of the turbine to maintain constant TSR corresponding to optimum  $C_p$ , thus maximizing energy capture. In above-rated wind speeds (Region 3), generator torque is held constant at rated torque, and blade-pitch control is used to limit aerodynamic power to maintain constant turbine speed. A transition region is included between Regions 2 and 3 (Region 2½) to allow the machine to reach rated torque at rated speed. If there were no Region 2½ and the machine were not allowed to exceed rated speed, the rated power of the turbine would be too low. This new region begins at a rotor speed  $\Omega_1$  and reaches rated torque at rated speed or slightly below ( $\Omega_2$ ) rated speed.

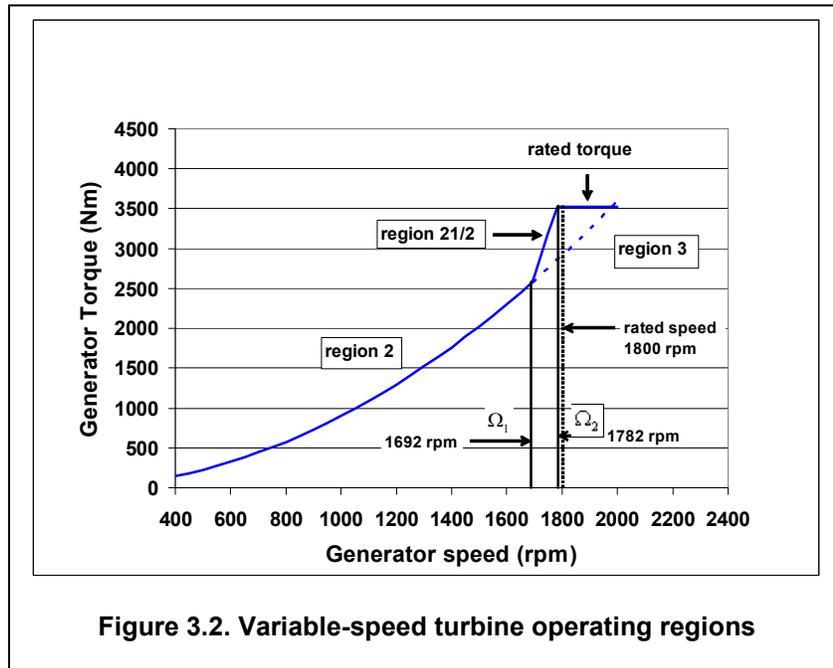


Figure 3.2. Variable-speed turbine operating regions

The model used for Region 2 generator torque is simple. To maintain optimum TSR in Region 2, the generator torque must be varied as the square of the rotor speed (2):

$$Q_{gen} = k\Omega^2, \quad (3.1)$$

where,

$$k = \frac{1}{2} \rho \pi R^5 \frac{C_{p_{max}}}{(\lambda_{opt})^3}. \quad (3.2)$$

Here,

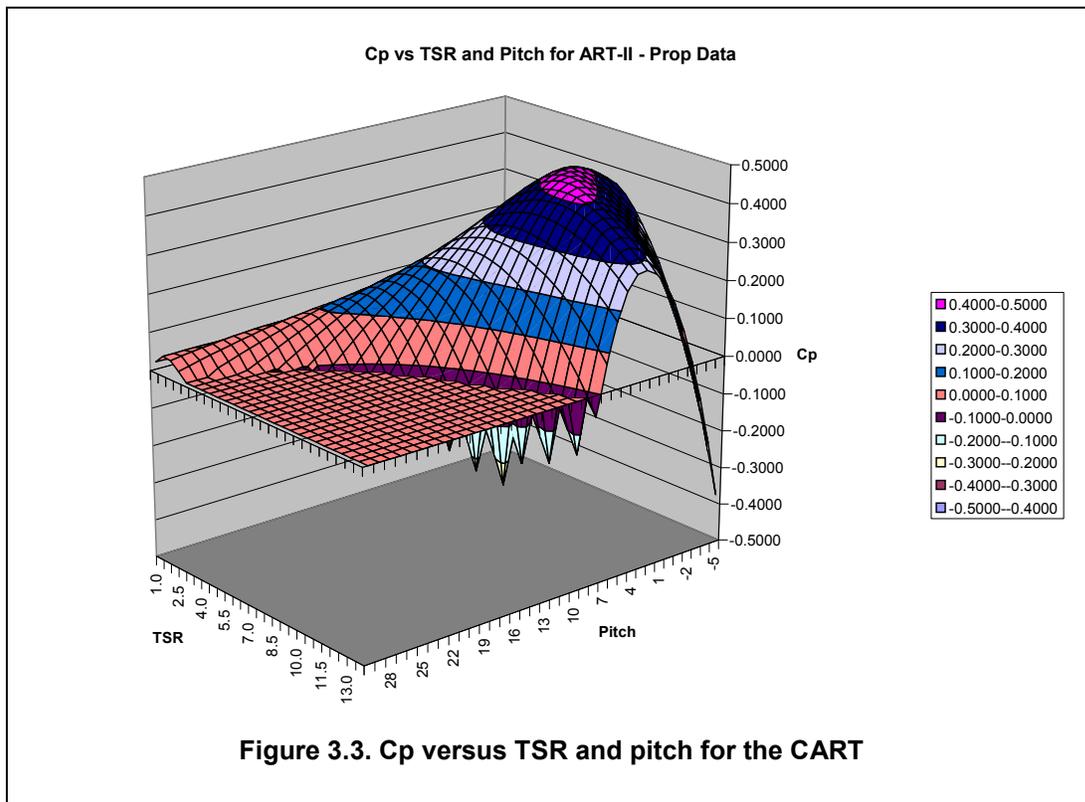
$\rho$  is the air density,

$R$  is the rotor radius,

$\Omega$  is the rotor speed (or generator speed), and

$C_{p_{max}}$  is the maximum power coefficient, corresponding to optimum TSR  $\lambda_{opt}$  at a particular blade pitch angle.

The parameters  $C_{p_{max}}$ ,  $\lambda_{opt}$  and the blade pitch angle at which  $C_{p_{max}}$  occurs are determined by examining a  $C_p$  versus TSR and pitch surface. This surface is usually determined through simulation, by using an aerodynamics code such as WT\_Perf (24) to generate values for this surface (see Figure 3.3). Table 3.1 shows these and other parameters needed for Region 2 generator torque control for the CART. The pitch angle



for  $C_{p_{max}}$  was determined to be -1 degree for the CART.

**Table 3.1. CART Parameter Values**

$R$	21.64 m
$\rho$	1.02 kg/m <sup>3</sup>
$C_{p_{max}}$	0.3659
$\lambda_{opt}$	7.5
Rated Torque	3524.4 N-m
Rated Speed	1800 rpm
$N_{gear}$	43.165

Evaluating (3.2) with the CART parameter values gives  $k \cong 6594.6 \text{ N-m-s}^2$ . Thus, in Region 2 the generator torque can be expressed  $Q_{gen} = 6594.6 \Omega^2 \text{ N-m}$ . This expresses the generator torque on the LSS side of the gearbox. In addition,  $\Omega$  is expressed in units of rad/s in the above equation. For input to the simulation code FAST, the generator torque is expressed on the HSS side of the gearbox and  $\Omega$  in units rpm instead of rad/s.

The constant  $k$  must then be multiplied by  $\frac{1}{N_{gear}^3} \left( \frac{\pi}{30} \right)^2$ , where  $N_{gear}$  is the gearbox ratio. Now we get  $Q_{gen} = .0008992 \Omega^2 \text{ N-m}$ , plotted as the Region 2 torque curve in Figure 3.2.

Figure 3.2 shows how the Region 2 torque curve crosses the rated torque line at a higher rotor speed (1980 rpm) than the rated speed (1800 rpm). Using  $Q_{gen} = k\Omega^2$  for generator torque results in a value below rated torque at rated speed. We want the generator torque to be equal to rated torque at rated speed. This means that we must insert a new region depending linearly on rotor speed, starting at  $\Omega_1$  and reaching rated torque at or slightly below rated speed (we reach rated torque at  $\Omega_2$ , which is slightly below rated speed in this example). This new Region 2½ is shown in detail in Figure 3.4.

The generator torque for this region can be expressed:

$$Q_{gen}(\Omega) = Q_1 + \frac{Q_{rated} - Q_1}{\Omega_2 - \Omega_1} (\Omega - \Omega_1), \quad (3.3)$$

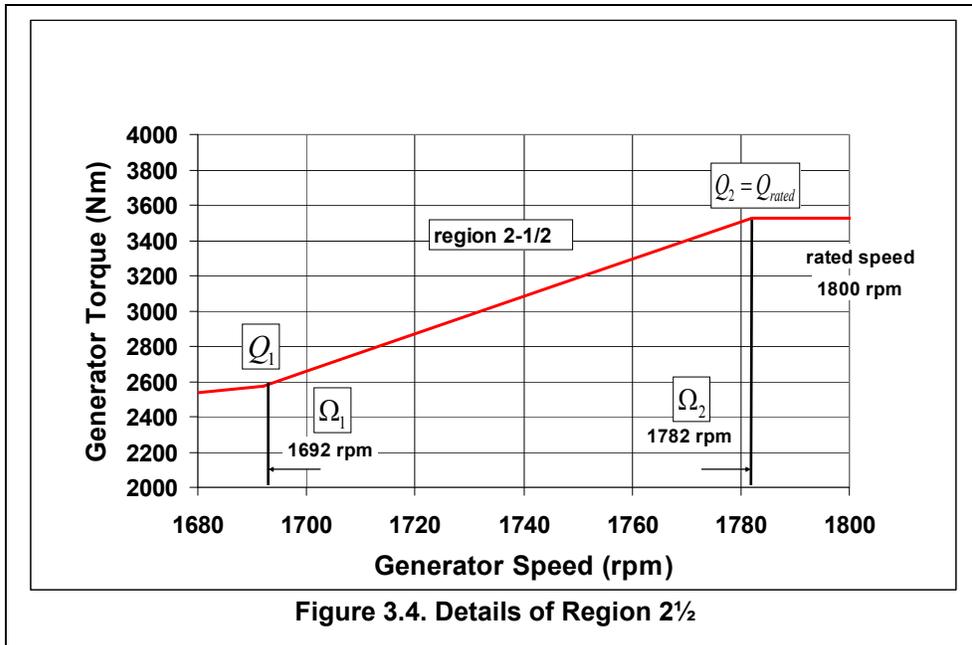
where,

$\Omega$  is rotor speed,

$Q_1$  is the generator torque at the rotor speed in which this region starts ( $\Omega_1$ ),

$Q_{rated}$  is rated torque, and

$\Omega_2$  is the rotor speed in which we reach rated torque.



Above rated speed, the generator torque is set equal to rated torque  $Q_{rated}$ .

We now simulate this variable-speed torque control in FAST by two methods. The first method uses a simple variable-speed generator model, defining all the necessary parameters from the FAST input file. Table 3.2 shows the simple variable-speed generator model parameters from the FAST input file for the CART. The inputs VS\_RtGnSp, VS\_RtTq, VS\_Rgn2K, and VS\_SIPc are described in the FAST User's Manual (16) on pages 26-27. Table 3.2 shows these values for the CART.

The entire FAST input file for this machine is shown in Appendix A, starting on page A1, with the Aerodyn input file listed beginning on page A4 and the wind input file listed beginning on page A5. First, we set VSContrl to 1, indicating that we are using a simple variable-speed generator model, defined from the FAST inputs. The input VS\_RtGnSp is the rated generator speed for this simple variable-speed generator control. We set this value slightly below the actual rated speed of the CART. The parameter VS\_RtTq is the rated generator torque, equal to the value shown in the table. VS\_Rgn2K is the Region 2 torque constant  $K$ . The last input VS\_SIPc allows us to model Region 2½. We now show how to calculate this input for the CART.

Using equation (3.3), suppose we want Region 2½ to begin at a generator speed of 1691.98 rpm ( $\Omega_1$  in Equation (3.3)) and reach rated torque at 1781.98 rpm ( $\Omega_2$  in Equation (3.3)). At 1691.98 rpm, the torque  $Q_1$  is found from  $Q_{gen} = K\Omega_1^2$  and has the value 2574.23 N-m. Rated torque  $Q_{rated}$  is 3524.36 N-m. Thus in Region 2½ the generator torque can be expressed as:

Table 3.2. FAST Variable-speed Generator Model Inputs.

1	VSContrl	- Variable-speed control mode {0: none, 1: simple VS, 2: user-defined from routine UserVSCont, 3: user-defined from Simulink} (switch)
1781.98	VS_RtGnSp	- Rated generator speed for simple variable-speed generator control (HSS side) (rpm) [used only when VSContrl=1]
3524.36	VS_RtTq	- Rated generator torque/constant generator torque in Region 3 for simple variable-speed generator control (HSS side) (N-m) [used only when VSContrl=1]
.0008992	VS_Rgn2K	- Generator torque constant in Region 2 for simple variable-speed generator control (HSS side) (N-m/rpm <sup>2</sup> ) [used only when VSContrl=1]
23.05	VS_SlPc	- Rated generator slip percentage in Region 2 1/2 for simple variable-speed generator control (%) [used only when VSContrl=1]

$$Q_{gen}(\Omega) = 2574.23 + 10.557(\Omega - 1691.98) \text{ N-m} \quad (3.4)$$

The equation for determining VS\_SlPc follows the same idea as the simple induction generator model described in the FAST User's Manual (16), pages 11 and 26. This simple induction generator model is a simple torque speed curve. We can use the same type of model here by selecting the parameters in this model to match our specified Region 2½ parameters. Here we replace SIG\_SlPc with VS\_SlPc:

$$\Omega_2 = \text{SIG\_SySp} \cdot (1 + 0.01 \cdot \text{VS\_SlPc})$$

where,

SIG\_SySp is the Synchronous (zero-torque) generator speed (rpm) and

VS\_SlPc is the rated generator slip percentage (%).

Our goal is to determine VS\_SlPc for input to FAST. We must first determine SIG\_SySp, the value of generator speed at which the generator torque is zero in this linear region. This value can be determined from (3.4) by setting the generator torque to zero and solving for generator speed, giving the value 1448.139 rpm.

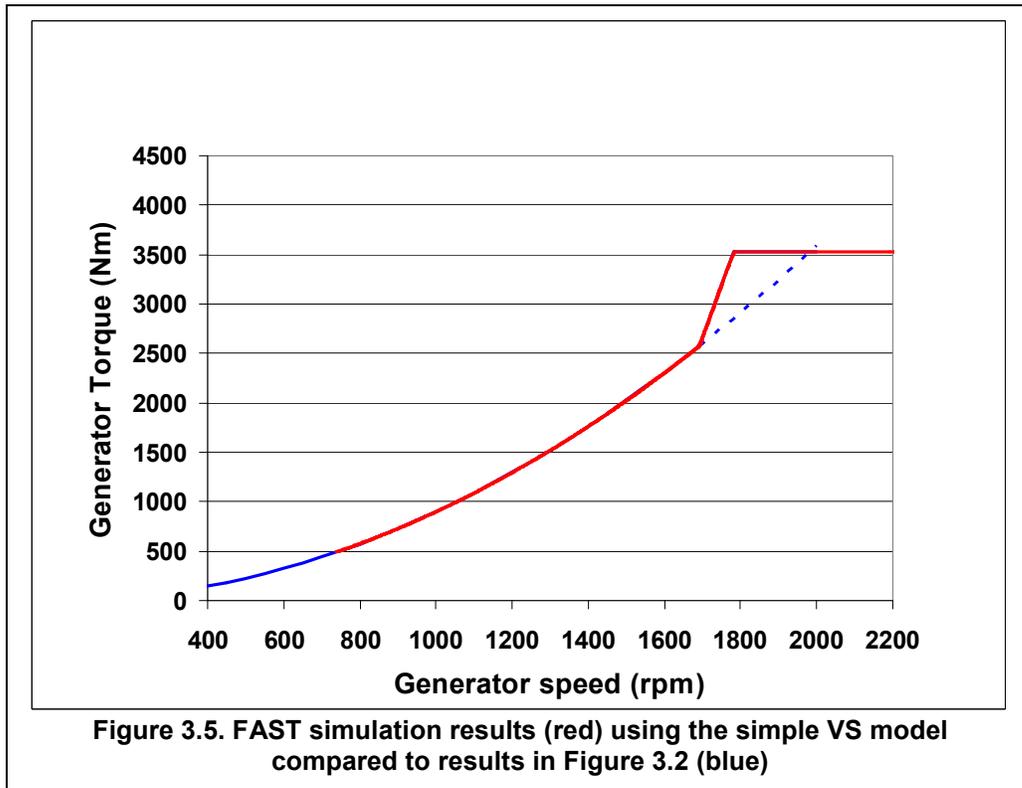
Solving for VS\_SlPc, we get (with  $\Omega_2 = 1781.98$ ):

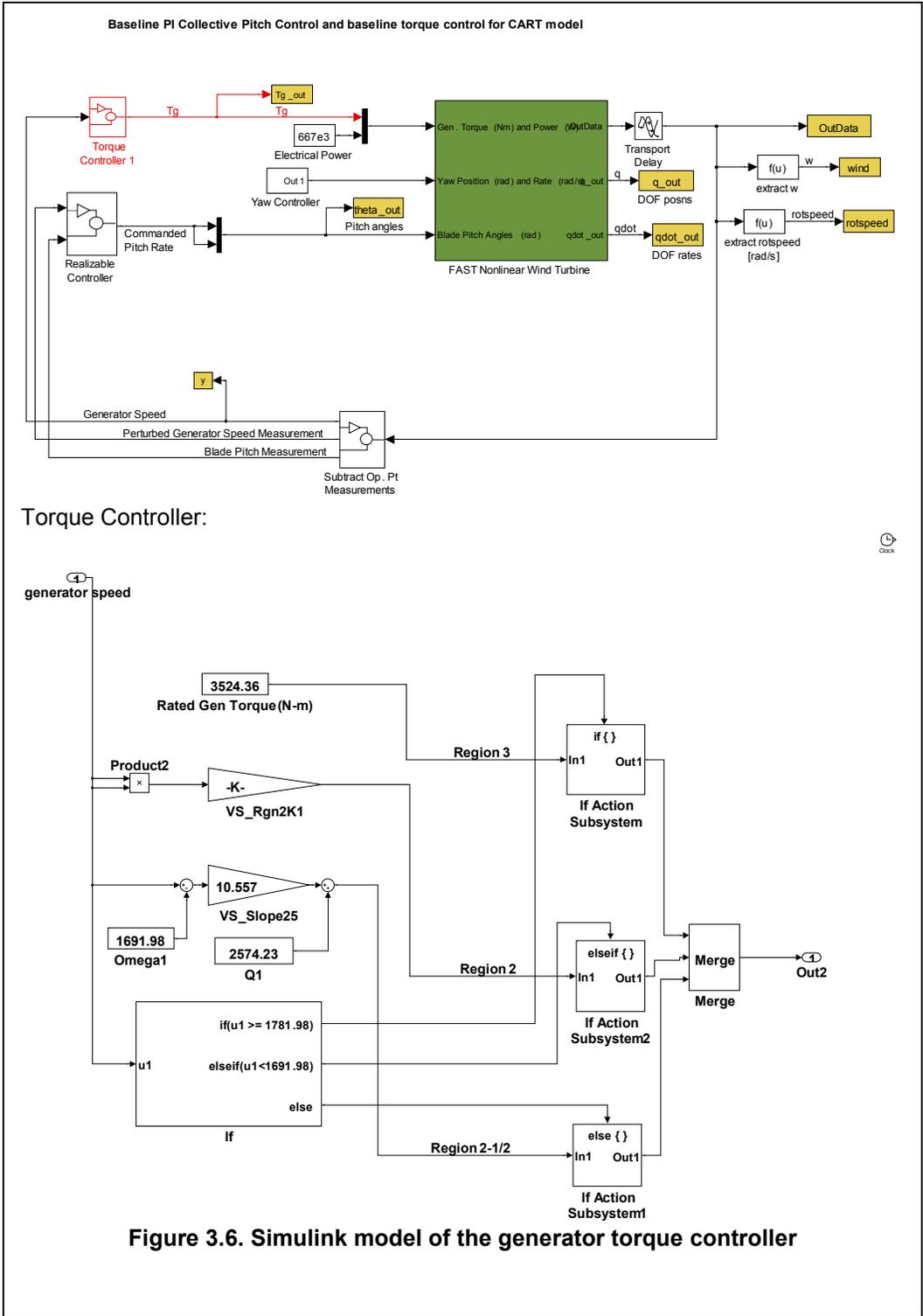
$$VS\_SIPc=100\times\left(\frac{\Omega_2}{SIG\_SySp}-1\right)=23.05.$$

The value in Table 3.2 for VS\_SIPc reflects this calculation. We have set VS\_RtGnSp equal to  $\Omega_2$  (1781.98 rpm) instead of rated speed (1800 rpm). Currently FAST does not distinguish between rated generator speed and  $\Omega_2$ , the end point of the Region 2½ linear interpolation. For small differences between these two values, this should not be a serious limitation.

Simulating this control, we use step winds to excite the FAST turbine model. These step winds cause the turbine operating point to begin in Region 2, pass through Region 2½, and end in Region 3. In the FAST model the only DOF switched on for this simulation is generator speed. Figure 3.5 shows the resulting generator torque versus generator speed for this simulation. In this plot, the red curve is the FAST simulated generator torque and the blue curve is the targeted generator torque from these equations. The results are identical, showing that we have implemented the desired controller in FAST.

The other method for simulating this system uses a Simulink model of the controller interfaced with FAST. For this case, we set VSContrl to 3 in the FAST input file. Figure 3.6 shows the Simulink generator torque controller.





In the upper part of the figure we see the overall controllers linked to the FAST model. The part we focus on here is the Torque Controller1, shown in red. The signal that enters this controller is generator speed.

In the lower part of the figure, we see the details of the Torque Controller1 box. We see the generator speed signal that is used in this controller. The different branches for Regions 2, 2½, and 3 can be clearly seen in this model. The Region 3 branch gives constant generator torque equal to rated torque. The Region 2 branch forms the squared generator speed, and then multiplies this value by the VS\_Rgn2K1 torque constant (the same value as input to the FAST input file). The third branch is for Region 2½, and performs the same calculations as equation 3.3) above. The decision about which branch to execute is based on the generator speed at any time, seen in the “if” block in this controller.

Simulation with this Simulink model gives results which are identical to the case of running the simulation with the simple generator model.

This shows how these tools can be used to perform steps 1 through 4 in the control development process for a simple generator torque controller. The control objectives for this controller are solely to maximize power in Region 2 using a very simple expression (model) for generator torque and to maintain constant generator torque in Region 3. A transition region is created (Region 2½) to tie these two regions together. If additional control objectives are formulated, such as actively damping flexible modes or mitigating rotor dynamic loads, this process becomes more complicated; thus, more complex models and controls are needed.

In this section we have shown the baseline generator torque controller for Region 2. What about development of a baseline pitch controller for Region 3? We now illustrate the development and simulation of this controller.

### 3.3 Region 3 Baseline Pitch Control Design

In this section, we demonstrate the design and simulation of a baseline PID rotor collective pitch controller for Region 3 CART operation. We use a FAST-Simulink model of the closed-loop system to describe simulating this controller. We also use a user-written subroutine to describe simulating this control in FAST.

#### 3.3.1 Basic Control Design

The goal of Region 3 pitch control is to regulate rotor speed to a certain set point (41.7 rpm for the CART). We maintain constant generator torque in Region 3 and use blade pitch to control rotor speed. A useful linear model for this simple control design is described in (17), p. 73, and has the form:

$$\Delta\dot{\Omega} = A\Delta\Omega + B\Delta\theta + B_d\Delta w \quad (3.5)$$

where  $A = \frac{\gamma}{I_{rot}}$ ,  $B = \frac{\zeta}{I_{rot}}$ ,  $B_d = \frac{\alpha}{I_{rot}}$ .  $I_{rot}$  is the total rotational inertia (due to the rotor, gearbox, shafts, generator, etc.).

Here

$$\gamma = \frac{\partial Q_{aero}}{\partial \Omega}, \zeta = \frac{\partial Q_{aero}}{\partial \theta}, \text{ and } \alpha = \frac{\partial Q_{aero}}{\partial w}$$

where,

$Q_{aero}$  is rotor aerodynamic torque,

$\Omega$  is rotor speed,  $\theta$  is blade pitch angle, and

$w$  is the hub-height uniform wind speed disturbance across the rotor disk.

Equation (3.5) is written in terms of perturbed values of these variables. These perturbations are assumed to represent small deviations of these variables away from their equilibrium values at steady state.

Our goal is to use PID pitch control to regulate turbine speed. We can describe this control by expressing the pitch perturbation  $\Delta\theta$  in Equation (3.5) as a summation of a term proportional to perturbed rotor speed, a term proportional to the integral of perturbed rotor speed, and a term proportional to the derivative of perturbed rotor speed. This is the standard PID control expression:

$$\Delta\theta(t) = K_p \Delta\Omega(t) + K_I \int \Delta\Omega(t) dt + K_D \Delta\dot{\Omega}(t). \quad (3.6)$$

Our goal in control design will be to determine appropriate values for the gains  $K_p$ ,  $K_I$ , and  $K_D$  to maintain a stable closed-loop system and achieve good response. The following paragraphs present a procedure and rationale for choosing these parameters to give desired closed-loop response characteristics. First, we need a model of the closed-loop system in the Laplace or s-domain.

Transforming both sides of Equation (3.6), we obtain an expression for the pitch perturbation in the Laplace domain as:

$\Delta\theta(s) = K_p \Delta\Omega(s) + K_I \frac{1}{s} \Delta\Omega(s) + K_D s \Delta\Omega(s)$ , where  $\Delta\Omega(s)$  and  $\Delta\theta(s)$  are the Laplace transforms of  $\Delta\Omega(t)$  and  $\Delta\theta(t)$ , respectively.

Laplace transforming both sides of (3.5) and moving the term associated with  $A\Delta\Omega$  to the left-hand side of the equation, gives:

$$\begin{aligned} \Delta\Omega(s)[s - A] &= B\Delta\theta(s) + B_d \Delta w(s) \\ &= B(K_p \Delta\Omega(s) + K_I \frac{1}{s} \Delta\Omega(s) + K_D s \Delta\Omega(s)) + B_d \Delta w(s) \end{aligned}$$

where  $\Delta w(s)$  is the Laplace transform of  $\Delta w(t)$ .

To investigate closed-loop system stability and response and to select appropriate values for the gains, we determine the closed-loop transfer function  $T_c(s)$  between the output measurement  $\Delta\Omega(s)$  and the disturbance input  $\Delta w(s)$ :

$$T_c(s) = \frac{\Delta\Omega(s)}{\Delta w(s)} = \frac{B_d s}{(1 - BK_D)s^2 + (-A - BK_p)s + (-BK_i)} \quad (3.7)$$

with parameters in (3.7) previously defined.

The denominator of this transfer function gives important information about the stability of this system. To have a stable closed-loop system, the roots of the equation (denominator of the closed-loop transfer function):

$(1 - BK_D)s^2 + (-A - BK_p)s + (-BK_i) = 0$  must all lie in the left-half of the complex plane (these roots must have negative real parts). This is equivalent to requiring that the coefficients of  $s$  in the above equation must all be positive (25) page 284, i.e.

$$1 - BK_D > 0, \quad -A - BK_p > 0, \quad \text{and} \quad -BK_i > 0.$$

We must now choose a suitable Region 3 operating point in which to evaluate the turbine parameters  $A$  and  $B$ . For the first case, we choose the wind speed, rotor speed, and blade pitch angle to be:  $w_0 = 18$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 11$  degrees. We can determine these turbine parameters at this operating point by running a linearization analysis with FAST. This will be demonstrated in Chapter 4 when we discuss the topic of state-space control design. For now, we give the results of this linearization:

$$A = -0.194, \quad B = -2.650, \quad \text{and} \quad B_d = 0.069.$$

The resulting characteristic equation becomes:

$$(1 + 2.650K_D)s^2 + (0.194 + 2.650K_p)s + (2.650K_i) = 0 \quad (3.8)$$

This gives the conditions for stability as:

$$K_D > -0.3774, \quad K_p > -0.0732, \quad \text{and} \quad K_i > 0.$$

Positive values of  $K_D$ ,  $K_p$ , and  $K_i$  increase the effective inertia, damping, and stiffness of the system described by (3.8). This shows us the effects of feedback in the form of (3.6) on this closed-loop system.

This gives us a method to select these gains to ensure system stability, but it does not necessarily give us guidelines for choosing these gains to give the desired response. What are some techniques for choosing values of  $K_D$ ,  $K_p$ , and  $K_i$  to give acceptable performance? We look at the characteristic equation in general form again:

$(1 - BK_D)s^2 + (-A - BK_P)s + (-BK_I) = 0$ . It is often advantageous to convert this to the form:

$$s^2 + 2\delta\omega s + \omega^2 = 0, \quad (3.9)$$

where  $2\delta\omega = \frac{-A - BK_P}{1 - BK_D}$ , and  $\omega^2 = \frac{-BK_I}{(1 - BK_D)}$ .

Solving for  $K_I$  and  $K_P$  we get:

$$K_I = \frac{-\omega^2(1 - BK_D)}{B}, \text{ and } K_P = -\frac{A}{B} - \frac{2\delta\omega(1 - BK_D)}{B}. \quad (3.10)$$

The roots of Equation (3.9) are  $s = -\delta\omega \pm \omega\sqrt{\delta^2 - 1}$ . For the underdamped case, when  $\delta < 1$ , we have two complex conjugate roots:  $s = -\delta\omega \pm j\omega_d$ , where  $\omega_d = \omega\sqrt{1 - \delta^2}$ . Here,  $\omega$  is called the undamped natural frequency,  $\omega_d$  the damped natural frequency, and  $\delta$  the damping ratio.

For the critically damped case, when  $\delta = 1$ , we have the two repeated roots  $s = -\omega \pm j0$ .

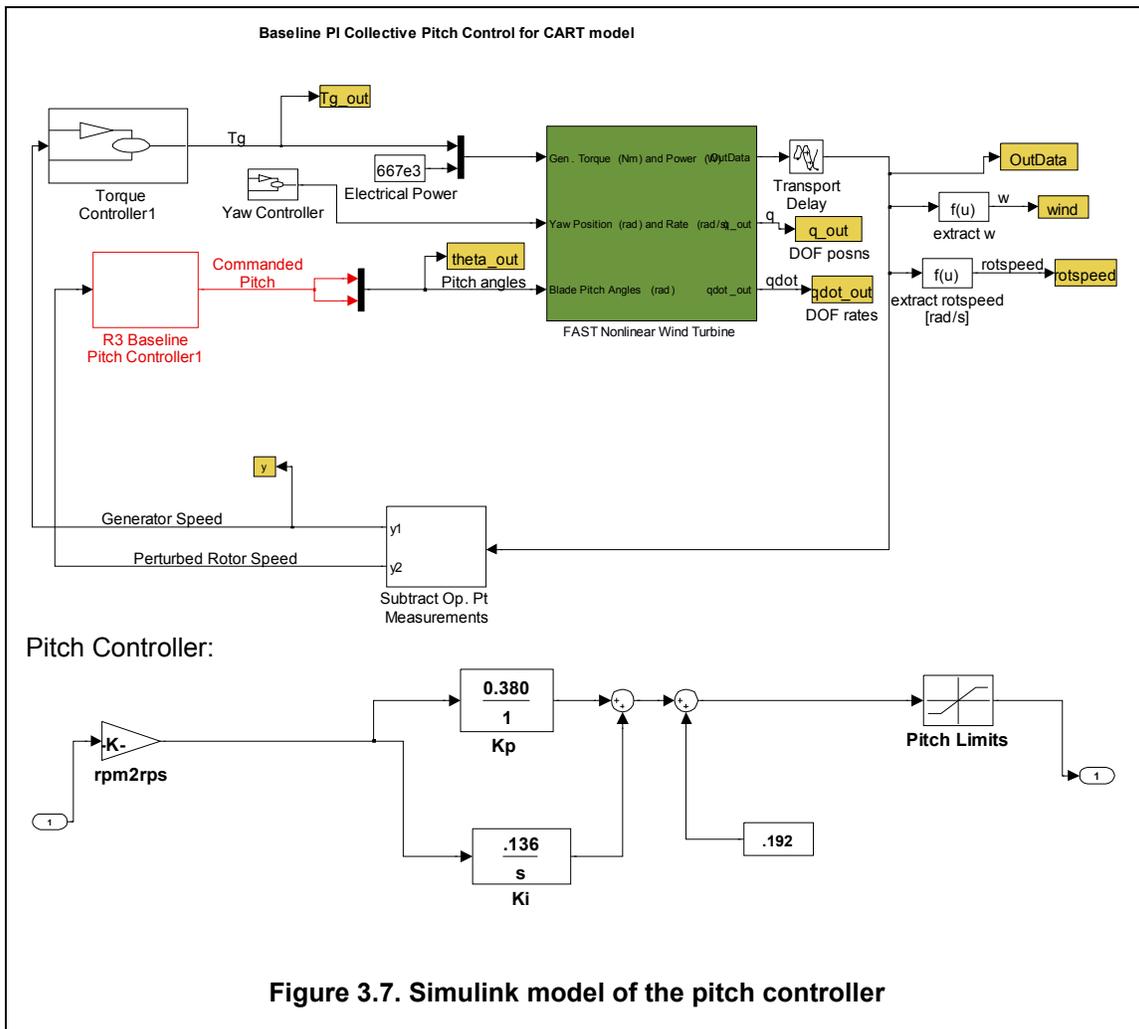
For the overdamped case, when  $\delta > 1$ , we have the two real roots:  $s = -\delta\omega \pm \omega\sqrt{\delta^2 - 1}$ . Different control performance can be achieved by selecting different values for the parameters  $\delta$  and  $\omega$ .

How do we choose values for  $\delta$  and  $\omega$ ? Risoe (14) suggests choosing  $\delta$  to have values in the range 0.6 to 0.7;  $\omega$  should be set to 0.6 for good performance. Choosing values for these parameters will probably be turbine dependent. For illustration purposes, suppose we want to achieve even higher performance by selecting  $\delta = 1$ . Let  $\omega = 0.6$  r/s. The calculations become much easier if we fix a value for one of these gains and calculate the other two gains based on that gain and the values chosen for  $\delta$  and  $\omega$ . For example, choose  $K_D = 0$ , then the other two gains can be calculated from Equation (3.10) as:

$$K_I = 0.136, \text{ and } K_P = 0.380s.$$

We can repeat this exercise for other values of  $K_D$ .

To verify control performance with these gains, we must now simulate the closed-loop system. Figure 3.7 shows a Simulink model for this controller. The upper part of the figure shows the overall R3 (Region 3) Baseline Pitch Controller1 model linked with the FAST Nonlinear Turbine model. The signal entering this box is perturbed rotor speed (rotor speed minus the set point 41.7 rpm). The lower part of the figure shows details of this controller box.



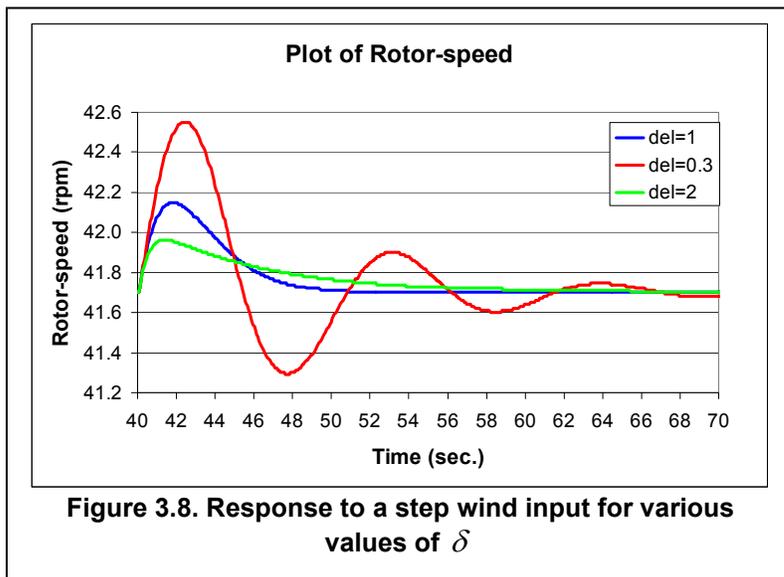
First, the perturbed rotor speed is converted from the units rpm to rad/s.

The rotor speed signal then enters the PI control boxes, with gains  $K_p$  and  $K_i$  (we do not include a derivative term, because we set the gain for derivative control to zero). The results from these two boxes are then summed to achieve a total pitch command. We then add the equilibrium pitch value from the linearization point (11 degrees or 0.192 radians). We then apply pitch limits of -1 degree (-0.0175 rad) at the lower limit, and 90 degrees (1.5708 rad) for the upper limit. If the turbine operation dips down into Region 2 (because of a decrease in wind speed), the pitch should saturate at this lower limit (-1 degree).

To evaluate performance, we simulate the response to a step change in wind speed and consider time-domain response characteristics such as rise time, settling time, overshoot, decay rate, steady-state offset, and frequency domain characteristics such as gain and phase margins (25), pp. 30-37. We will not discuss all the performance parameters here.

We simulated this controller with a unit step wind input occurring at time 40 s (at 40 s the wind changes from 17 m/s to 18 m/s). This wind input file is shown in Appendix A, page A6. The FAST input file for this simulation is the same as shown on page A1.

Figure 3.8 shows the predicted rotor speed response to this step wind input. We ran cases in which we chose the damping ratios  $\delta = 1$  as well as  $\delta = 0.3$  and  $\delta = 2.0$ . In all cases we chose  $\omega = 0.6$  r/s. For  $\delta = 0.3$  we see a damped oscillation, as the damping ratio is less than 1, and the solution involves two complex conjugate roots. For  $\delta = 1$  we get a critically damped response, which represents the fastest decay time that we can achieve. Increasing  $\delta$  to 2 results in a longer decay time, as the solution now contains two exponentials corresponding to two real roots. The root that lies closest to the imaginary axis in the complex plane will dominate the solution (the root closest to the imaginary axis will lie to the right of the two roots obtained when  $\delta = 1$ , resulting in less damping), causing a longer decay time than the solution corresponding to critical damping.



We have designed a linear controller at a single Region 3 operating point. How does this controller perform for operating points that do not match the control design point? We now investigate this question.

### 3.3.2 Gain Scheduling

The techniques just described allow us to design a simple Region 3 pitch control system, based on the simple linear model shown above. We calculate gains based on the values of turbine parameters A and B at the operating point

$w_0 = 18$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 11$  degrees. We investigate the performance of the controller at other operating points. Figure 3.9 shows the response (red curve) when we apply a step wind input (again 1 m/s step change) at 40 s resulting in operation at a lower pitch angle (case 4) than the previous example (case 1). This turbine operation point is close to the Region 2 to Region 3 transition point. Now, the maximum overshoot is much greater and the performance at this operating point is poorer than what we designed (the overshoot and decay time are greater).

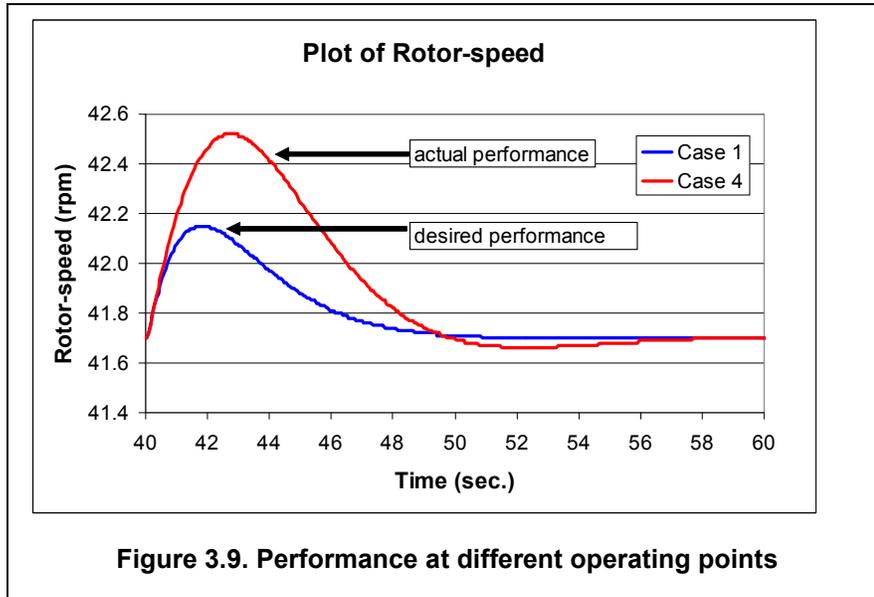
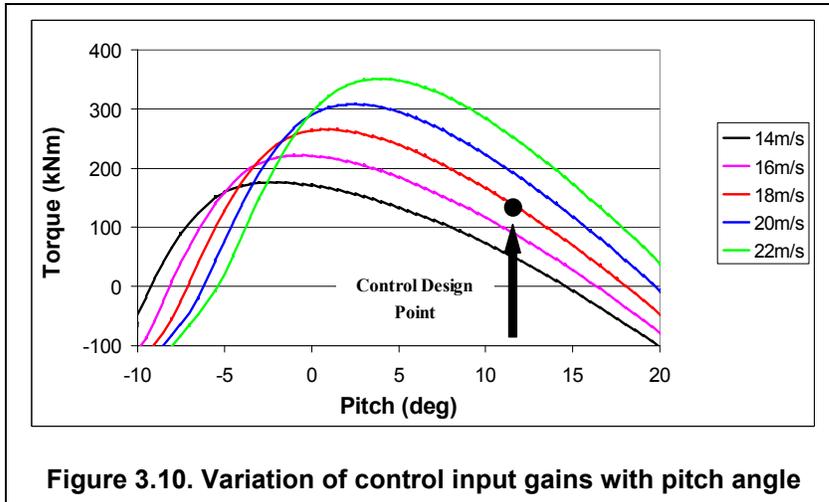


Figure 3.9. Performance at different operating points

The change in performance is caused by the variation of pitch control input gains with blade pitch angle and wind speed. The control input gain  $B$  in equation (3.5) is directly related to the partial derivative of aerodynamic torque with respect to blade pitch angle ( $\frac{\partial Q_{aero}}{\partial \theta}$ ). Figure 3.10 shows the rotor aerodynamic torque for various wind speeds and blade pitch angles. The control input gain (the slope of these curves) changes with pitch angle. The gains are small for low pitch angles in which the turbine is transitioning from Region 2 into Region 3. The gains increase as the pitch angle increases (the slopes are negative). If we design the control gains to have a particular performance at the indicated control design point, we cannot expect the same performance for smaller pitch angles. We could redesign the controller for a lower pitch angle operation point, but when the pitch angle increases we would not achieve the performance we designed to at the low pitch angle.

The solution to this problem is to schedule the gains as a function of blade pitch angle. Each PI gain is multiplied by a function GK of the form:



$$GK(\theta) = \frac{1}{\left(1 + \frac{\theta}{KK}\right)}$$

where,

$\theta$  is the pitch angle and

$KK$  is pitch angle chosen further into Region 3 (14). This technique is described further in (14) and (26).

We first design  $K_i$  and  $K_p$  to have desired performance (choosing the values of  $\delta$  and  $\omega$  as done in the example above) at a control design point close to the point of entry from Region 2 into Region 3. We perform a FAST linearization to determine the new values of  $A$  and  $B$  at this new control design point. The value  $KK$  is chosen to be the pitch angle for which  $B$  has doubled in value from its value at the new control design point. We now illustrate this method for the CART.

Let us choose a Region 3 operating point close to the point of entry from Region 2 into Region 3. An example is  $w_0 = 13.7$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 0.53$  degrees. At this operating point, the values for  $A$  and  $B$  are:

$$A = -0.116 \text{ and } B = -0.462.$$

We choose  $\delta = 1$  and  $\omega = 0.6$  r/s as we did for the previous case. This gives:

$$K_i = 0.780, \text{ and } K_p = 2.348s.$$

We now find an operating point at which the value of  $B$  is approximately twice its value at the control design point  $w_0 = 13.7$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 0.53$  degrees. We can run the FAST linearization at various wind speeds, trimming on pitch (16), to find that at the operating point  $w_0 = 14.1$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 2.62$  degrees we achieve a

value of  $B = -0.930$ . At this operating point  $B$  is roughly twice its value at  $w_0 = 13.7$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 0.53$  degrees. We choose  $KK = 2.62$  deg. Thus:

$$GK(\theta) = \frac{1}{\left(1 + \frac{\theta}{2.62}\right)}$$

Figure 3.11 shows a Simulink model of this revised pitch controller with gain schedule. We multiply the two gains  $K_i$  and  $K_p$  by the factor  $GK$ , which is calculated in the box labeled  $GK2$ .

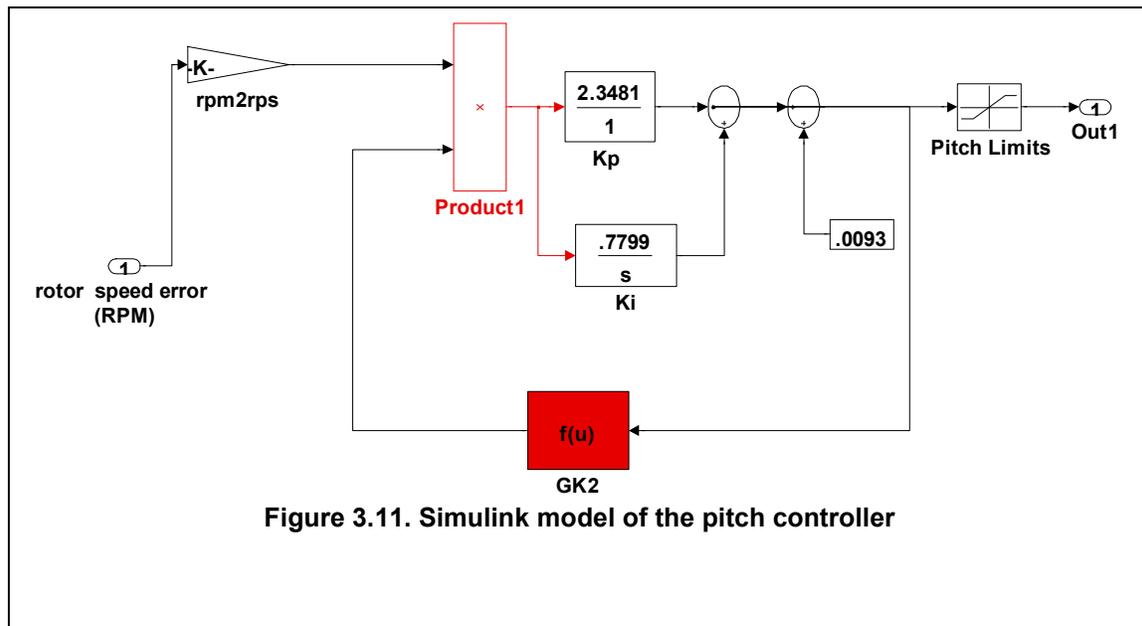
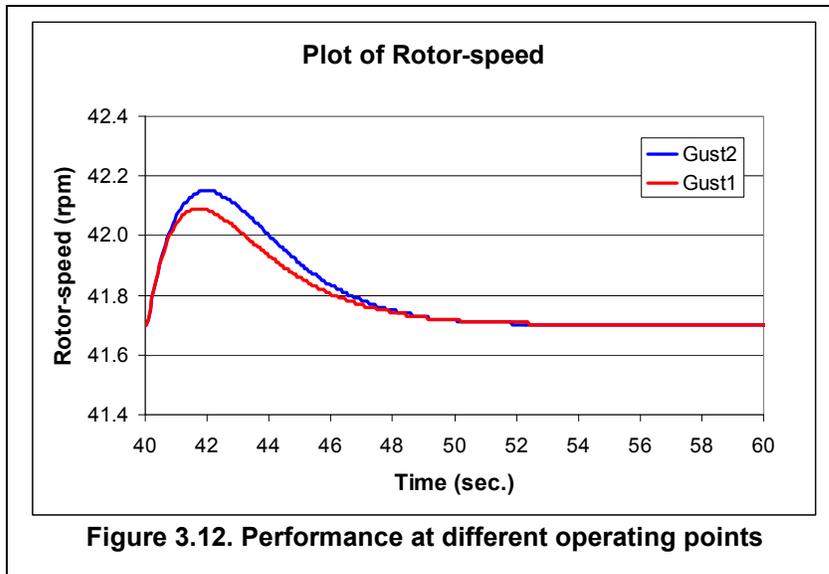


Figure 3.11. Simulink model of the pitch controller

We simulate with two step wind cases. In the first case, the wind speed increases from 14 m/s to 15 m/s at  $t = 40$  s. In the second case the wind speed changes from 17 m/s to 18 m/s at  $t = 40$  s. Figure 3.12 shows the response from  $t = 40$  to  $t = 60$  s. Now the performance for the two wind speed cases is much closer than the results of Figure 3.9 without gain scheduling.

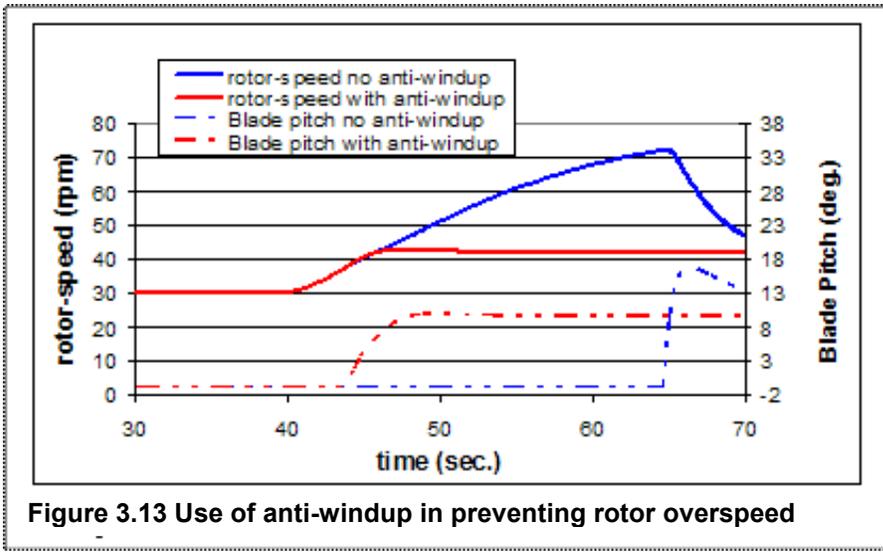
An alternative method of computing the parameters for use in the above expression is described in (26), page 104, based on a best fit least squares estimate of the pitch sensitivity for various blade pitch angles (26).



### 3.3.3 Anti-Windup

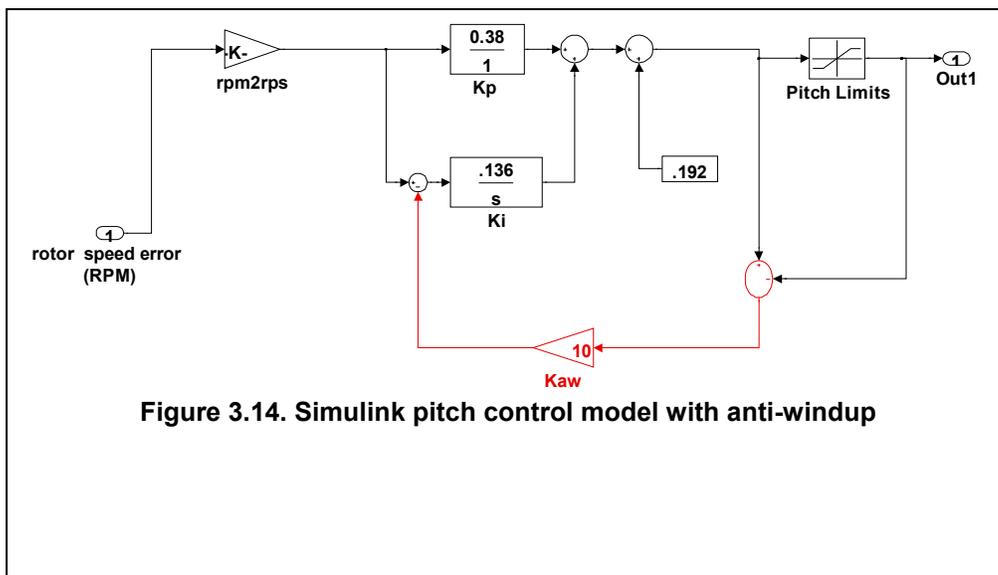
Another issue is the performance of the pitch control system when a gust of wind causes the turbine to suddenly change from Region 2 to Region 3. In Region 2, the blade pitch is saturated at the lower pitch limit of  $-1^\circ$ . Figure 3.13 shows a large overspeed (rotor speed no anti-windup) when a step change in wind speed from 9 m/s to 17 m/s is applied (at 40 s) to the rotor. Before the gust is applied, the rotor speed is below the Region 3 set point of 41.7 rpm and pitch is saturated at  $-1^\circ$ . A negative speed error is being fed to the integrator part of the controller. The integrator continuously integrates this negative error resulting in a large negative pitch angle, with the pitch angle limited to  $-1^\circ$ . When a gust of wind is applied to the rotor, the rotor speed will increase from its equilibrium value before the gust was applied. When the rotor speed reaches a value greater than 41.7 rpm, a positive speed error is fed to the integrator. It takes a long time for this positive speed error contribution to cancel the effects of the negative pitch angle contribution that has been built up from integration of these negative speed errors. In Figure 3.13 we see a large delay between the time that the gust was applied (40 s) and the time that the pitch angle (blade pitch no anti-windup) becomes positive (65 s) and begins to actively regulate speed. During this long delay time, the rotor speed has increased to higher than 70 rpm. Such performance cannot be tolerated.

The solution is integrator anti-windup and can be implemented easily. Figure 3.14 shows implementation of anti-windup into the Simulink model of this turbine, shown as the feedback with gain KAW in the figures. This anti-windup term is fed back to the integrator only. This prevents the integrated speed error from accumulating when the rotor is operating in Region 2. Through trial and error, we chose a gain of 10 rad/s/rad in these simulations. The value for this gain may be turbine dependent. When the pitch angle is not saturated, this anti-windup feedback term is zero, since the pitch angle exiting the pitch limits box is equal to the pitch angle entering that box. The signal entering the KAW gain is then zero.



Now when the large step-change in wind speed is applied to this model, the large rotor overspeed is eliminated, as we see in the red plot in Figure 3.13 (rotor speed with anti-windup). We see that the blade pitch for this case (blade pitch with anti-windup) provides proper actuation just a few seconds after application of the gust in order to maintain rotor speed at the 41.7 rpm set point.

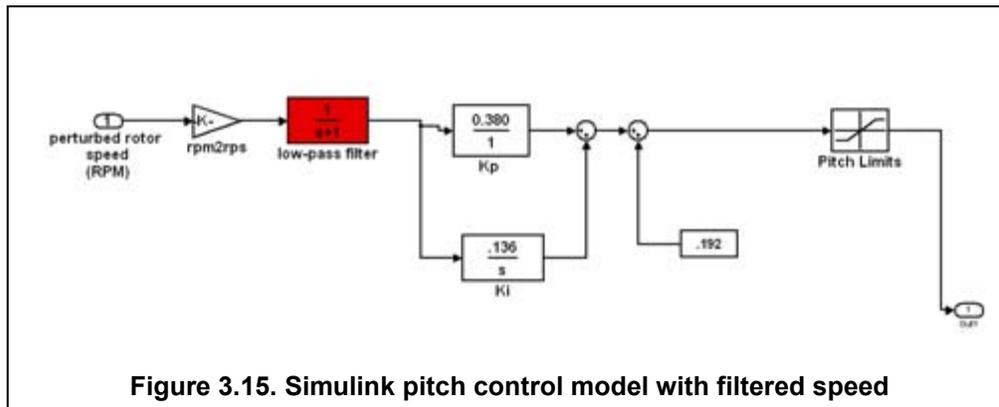
These steps describe how to develop and simulate a standard industry baseline controller. In some cases the inputs may need to be filtered to the controller (measured generator or rotor speed) to prevent instabilities. We now show when this may be necessary.



### 3.3.4 Filtering the Generator or Rotor Speed Measurement

The previous sections highlighted simulations performed with the baseline Region 3 PID controller. In those simulations, the only active DOF in FAST was the generator DOF. In (17), it was shown that simulations performed with a Region 3 pitch controller may destabilize the first drive train torsion mode when that DOF is switched on during simulations. We will show state-space control designs that stabilize this mode in Chapter 4.

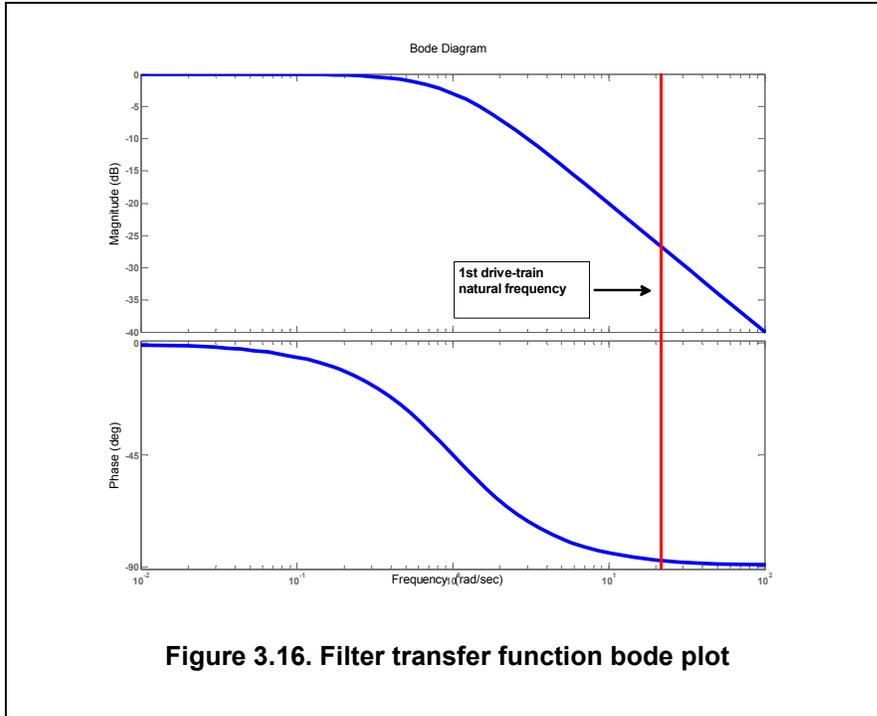
For baseline PID pitch controllers, a solution to this stability problem involves use of a low-pass filter. The measured rotor or generator speed can be filtered before being input to the controller. Without filtering, the measured generator or rotor speed signal will contain oscillations at the first drive train torsion frequency. When the PID controller attempts to regulate speed, it will attempt to regulate these rotor speed perturbations at the first drive train torsion frequency. Because the controller does not have the needed information to provide stabilizing control of this mode, it destabilizes this mode. When a low-pass filter is applied to the rotor or generator speed signal, the oscillations at this frequency are filtered out, and the controller no longer responds at this natural frequency. The result is stable control in this mode. Figure 3.15 shows the PI pitch controller with the added low-pass rotor speed filter. Simulation with this controller with the first drive train torsion mode switched on during simulation results in stable behavior.



How does one select a filter? In this case we selected the first-order filter with transfer function:

$$F_{fil}(s) = \frac{1}{s + 1}.$$

This low-pass filter was selected based on the natural frequencies of flexible modes. We want to filter out fluctuations in the rotor speed signal at the first drive train torsion natural frequency, approximately 22 r/s in the CART. Examining a bode plot of this filter transfer function (Figure 3.16) is helpful. The bode plot of this filter shows good attenuation (-26 dB) at this natural frequency, while at low frequencies (0 to 0.1 r/s) there is no attenuation. This means that low-frequency portions of the rotor speed signal will



**Figure 3.16. Filter transfer function bode plot**

not be filtered, which is important for the overall speed regulation performance of the controller.

A mode that benefits from active damping with pitch control is the tower's first f-a mode. We now describe design of a simple add-on control to the baseline pitch controller to add active damping to this mode.

### **3.3.5 Active Tower Damping Control**

As mentioned in Chapter 1, some industry baseline Region 3 controllers attempt to use pitch control to actively damp the tower f-a motion. Another control loop is added to the basic PID speed control loop. To design this controller for active tower damping, we assume that the flexible tower can be approximated by a linear modal representation with the dominant tower motion described by the tower first f-a mode. The equation of motion can be written:

$$M_t \Delta \ddot{x} + C_t \Delta \dot{x} + K_t \Delta x = F_t \Delta \theta$$

where,

$\Delta x, \Delta \dot{x}, \Delta \ddot{x}$  are the perturbed tower f-a deflection, velocity, and acceleration in the first bending mode,

$M_t, C_t, K_t$  are the first bending mode modal mass, damping, and stiffness coefficients, and

$\Delta\theta$ , and  $F_t$  are the perturbed pitch input and input gain.

Now we assume that the perturbed pitch input is proportional to tower velocity to add active tower damping:

$$\Delta\theta = G\Delta\dot{x}.$$

We will adjust the amount of tower damping by our choice of  $G$ .

Thus:

$$M_t\Delta\ddot{x} + C_t\Delta\dot{x} + K_t\Delta x = F_tG\Delta\dot{x}$$

or

$$M_t\Delta\ddot{x} + (C_t - F_tG)\Delta\dot{x} + K_t\Delta x = 0.$$

Taking the Laplace transform of both sides we have:

$$\Delta x(s)[M_t s^2 + (C_t - F_tG)s + K_t] = 0.$$

The characteristic equation is  $M_t s^2 + (C_t - F_tG)s + K_t = 0$ , or

$$s^2 + 2\delta\omega s + \omega^2 = 0, \text{ where } 2\delta\omega = \frac{C_t - F_tG}{M_t} \text{ and } \omega^2 = \frac{K_t}{M_t}.$$

The roots of the characteristic equation are:  $s = -\delta\omega \pm \omega\sqrt{\delta^2 - 1}$

We now give an example for the CART. Table 3.3 shows the tower mass, damping, stiffness, and control input parameters for the following operating point:

$w_0 = 18$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 11$  deg. These values were determined by linearizing a FAST model with just the tower first f-a mode switch on. We will explain running a FAST linearization analysis in Chapter 4.

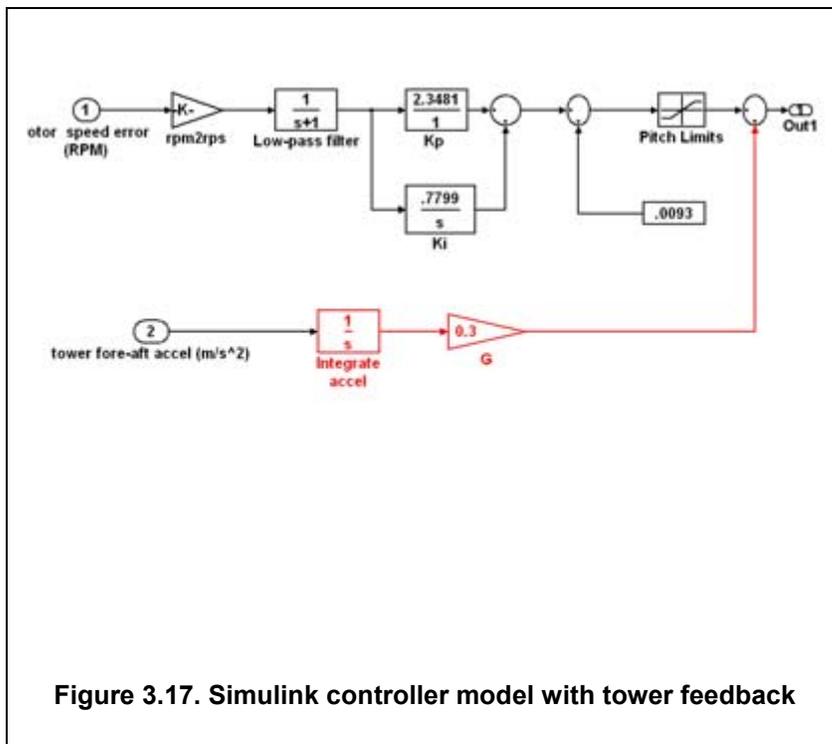
**Table 3.3. CART Tower Parameter Values**

$M_t$	54606 kg.
$C_t$	7589 kg/s
$K_t$	1652000 kg/s <sup>2</sup>
$F_t$	-349267 kg-m/s <sup>2</sup>

For this case, the undamped natural frequency is  $\omega=5.50$  r/s. If we set  $G=0$  (no pitch control case), the damping ratio is  $\delta=0.0126$ .

Suppose we want to use pitch control to achieve critical damping ( $\delta=1$ ). We can calculate the value of  $G$  necessary for this amount of damping by  $G = \frac{C_t - 2M_t\delta\omega}{F_t}$ , giving  $G=1.698$ .

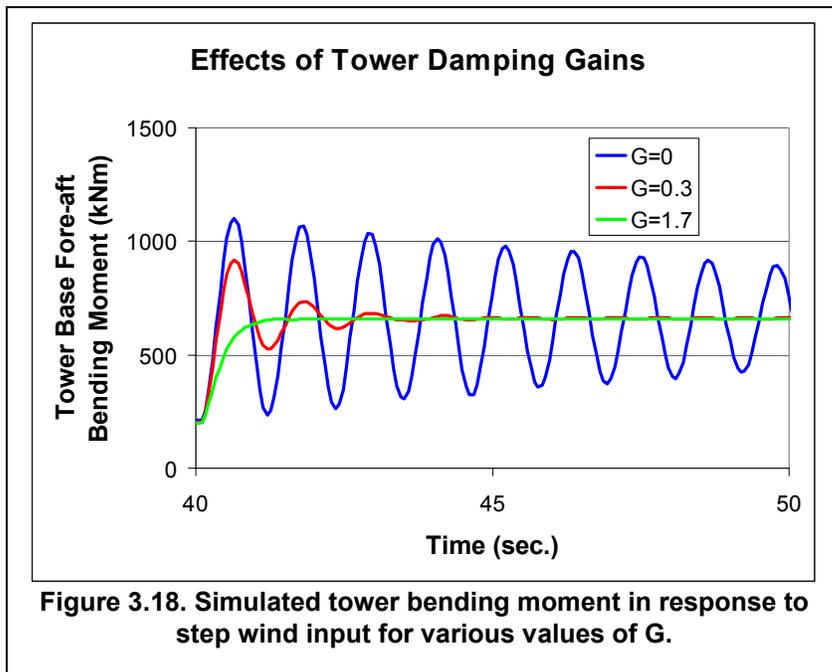
Figure 3.17 shows the Simulink model used to simulate this case, with an added control loop to perform this tower damping. For this control, we measure tower f-a acceleration and then integrate that signal and multiply by the gain  $G$ . We then add this increment of pitch control onto the baseline PI pitch control.



**Figure 3.17. Simulink controller model with tower feedback**

Figure 3.18 shows the effect of various values of  $G$  on the tower base f-a bending moment as simulated with FAST for this machine. At  $t = 40$  s a step change in wind speed is applied. Note the highly oscillatory response for  $G=0$  ( $\delta=0.0126$ ). The response becomes more highly damped and decays more rapidly with higher gains  $G$ . For critical damping ( $G=1.7$ ), no oscillations occur.

These steps show how to add active tower f-a damping to the baseline pitch controller. One must be cautious in designing such a controller to be sure other tower modes, such as the first side-side mode, do not become unstable with this control. Simulations should be performed with both modes switched on in FAST to ensure these modes are stable.



These descriptions have shown how to use a MATLAB-Simulink model of the controller interfaced with FAST to simulate the closed-loop system. Another way to simulate the closed-loop system with FAST is to use the pitch control subroutine compiled and linked with the code. This method is useful for engineers who choose not to use the FAST-Simulink simulation capability. This method, along with a Fortran subroutine for pitch control (to be compiled and linked with FAST), is described in Appendix C.

In this chapter we have shown the design and simulation of Region 2 and Region 3 baseline controllers. Such controllers are standard in industry. As wind turbines become lighter and more flexible, control objectives such as load mitigation and active damping of coupled modes will become important. Advanced state-space controls can be used to advantage to meet these multiple control objectives. We now describe some state-space control design and simulation cases in the next chapter.

## 4. Illustrating State-Space Control Design Steps and Tools

### 4.0 Goals and Contents

The goal of this chapter is to illustrate the steps involved in designing a state-space controller. We also illustrate the use of the control design and simulation tools. We illustrate state-space control design and simulation for three examples: (1) collective blade pitch control to regulate speed and actively damp drive train torsion in Region 3; (2) generator torque control to actively damp drive train torsion in Region 3; and (3) independent blade pitch control for load reduction in Region 3. We begin with the collective blade pitch control algorithm.

### 4.1 Region 3 Collective Pitch Control Design Example

#### 4.1.1 Control Objectives

The goal for blade pitch control in Region 3 is to use rotor collective pitch to regulate turbine speed. For this example we provide active damping of the first drive train torsion mode. We also include the rotor's first symmetric flap mode in the linear model used for control design. This mode is coupled to the first drive train torsion mode as shown in (17). We also assume a uniform wind disturbance input. For more detail about this turbine model and a detailed description of the control design, see (17), page 57. Table 4.1 shows a list of the turbine states contained in  $\Delta x$  for this model.

State	Description
$\Delta x_1$	perturbed drive train torsional deflection
$\Delta x_2$	perturbed rotor first symmetric flap mode displacement
$\Delta x_3$	perturbed generator rotational speed
$\Delta x_4$	perturbed drive train torsional velocity
$\Delta x_5$	perturbed rotor first symmetric flap mode velocity

We now describe how we perform the FAST linearization for this model.

#### 4.1.2 FAST Linearization

A general turbine linear model can be described by (see Section 2.3)

$$\underline{\Delta \dot{x}} = A\underline{\Delta x} + B\underline{\Delta u} + B_d \underline{\Delta u}_d \tag{4.1}$$

$$\underline{\Delta y} = C\underline{\Delta x} + D\underline{\Delta u} + D_d \underline{\Delta u}_d .$$

where,

$\underline{\Delta x}$  is the state vector,

$\underline{\Delta u}$  is the control input vector,

$\underline{\Delta u}_d$  is the disturbance input vector, and

$\underline{\Delta y}$  is the control (or measured) output.

$A$  represents the state matrix,

$B$  the control input gain matrix, and

$B_d$  the disturbance input gain matrix.

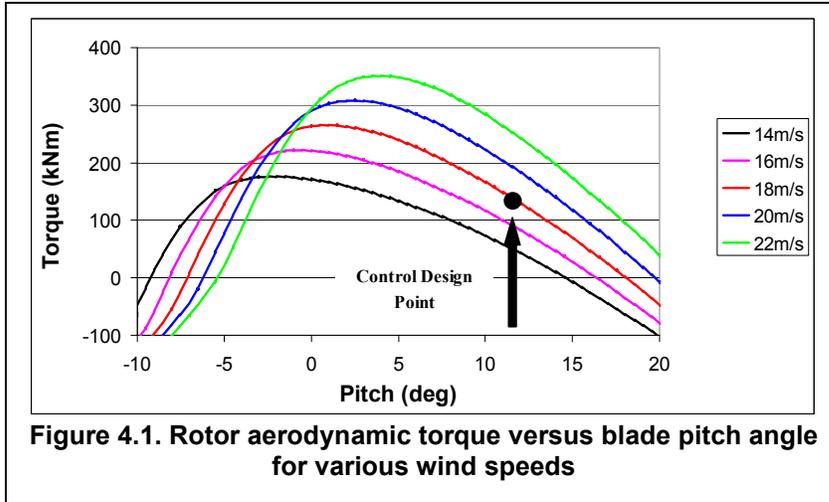
$C$  relates the measured output  $\underline{\Delta y}$  to the turbine states.

$D$  relates the control input to the output.

$D_d$  relates the measured output to the disturbance states.

In this notation,  $\underline{\Delta \dot{x}}$  represents the time derivative of  $\underline{\Delta x}$ .

To determine a linearized state-space model for control design, we run FAST with appropriate DOFs switched on to model the states shown in Table 4.1. In the FAST input file used to simulate linearization, we switch on the first flapwise blade mode DOF, the drive train rotational-flexibility DOF, and the generator DOF (we set FlapDOF1, DrTrDOF, and GenDOF to True, all others to False). We want to perform model linearization about the control design point (Figure 4.1). This figure shows rotor aerodynamic torque versus wind speed and blade pitch angle.



The turbine parameter input file for performing this FAST linearization simulation is named CARTnewlin.fad and is listed in Appendix A, starting on page A6. We set RotSpeed to 41.7 rpm, the BIPitch(1) and BIPitch(2) values to 11 degrees to reflect the initial conditions for the linearization simulation. Setting these values ensures that the simulation converges to the desired trim solution in a reasonable simulation time. The wind speed file used for this simulation is the same as shown on page A6, except that now the wind speed is set equal to 18 m/s throughout the file. The cartnewlin.fad file listing on page A6 shows that we have set AnalMode to 2: Create a periodic linearized model. We also declare the output HSShftV in the output list, because we will need it as output in the Simulink model to be described.

The corresponding file for the aerodynamic inputs for this simulation is named AeroDyn01lin.ipt and is listed on page A9. We run the linearization with a constant wind speed (no turbulence, shear, etc.). We have specified a constant wind speed input contained in the AeroDyn wind input file (not shown) Wind/CONST18mps.wnd.

The file containing the FAST linearization parameters is called CART\_Linear.dat, listed on page A10. We have specified the parameters NInputs, CntrlInpt, Ndisturbs, and Disturbnc to have the values 1, 4, 1, and 1 respectively, corresponding to the rotor collective pitch control input and the horizontal hub height wind speed disturbance.

Running this simulation produces an output file named cartnewlin.lin. This file contains the periodic state matrices of the linearized system, the periodic operating point states and state derivatives, the periodic operating point output measurements, the constant operating point values of the control inputs and wind inputs, and other information useful for postprocessing. Through postprocessing, we obtain azimuth averaged state matrices (16) necessary for our control design.

One postprocessing tool is the MATLAB script eigenanalysis.m, as described in (16), page 43. This script calculates azimuth-averaged state matrices AvgAMat, AvgBMat, AvgBdMat, AvgCMat, AvgDMat, and AvgDdMat. These matrices correspond to the

state matrices  $A$ ,  $B$ ,  $B_d$ ,  $C$ ,  $D$ , and  $D_d$  contained in equation (4.1). This script also uses the AvgAMat matrix to perform an eigenanalysis of the system. Running this analysis gives us the open-loop poles for this averaged linear model.

A point of clarity needs to be added here. The states we need in the linear model for control design include the rotor first symmetric flap mode displacement and velocity. We do not include the states for the first flap mode of each blade because the state-space system with these states is uncontrollable using rotor collective blade pitch as the control input (17). If the states for the first flap mode of each blade are retained in the linear model, independent pitch control must be used to satisfy controllability (17).

We switched on the first flap mode in the FAST input file for linearization. The linear model produced from this FAST simulation includes the states corresponding to the first flap mode of each blade. These are not the states needed for our control design model. To create the states that correspond to the rotor first symmetric flap mode from the blade 1 and 2 flap states, we need to apply a transformation as described in (17), page 57. This transformation essentially forms the rotor first symmetric flap mode from these states. This transformation will be applied when we run the MATLAB script used for control design.

In addition, we have switched on the generator DOF in FAST for this linearization. This means that one of the states contained in the FAST generated linear model produced from this linearization will be the generator azimuth state, which is not included in the list of states shown in Table 4.1. We eliminate the generator azimuth state and include the generator speed state in the linear model to be used for control design because if we retain the generator azimuth state, but measure only generator speed, the resulting state-space system is unobservable (17). Measuring generator speed is a typical turbine measurement on most commercial machines, so we will not measure generator azimuth angle. This will also be performed in the MATLAB control synthesis script.

### **4.1.3 Control Design Synthesis**

The MATLAB script we use to design this controller (LQR\_design\_DAC6states.m) is listed in Appendix B, starting on page B1. This script can be used to design either a full state feedback controller or a realizable state estimator controller, based on measuring only generator speed. This script is based on DAC design (17).

The first part of the script reads state matrices produced by the FAST linearization simulation. This script then forms azimuth-averaged state matrices in the same manner as the script eigenanalysis.m.

Next, the special matrices  $A_{_c}$ ,  $B_{_c}$ ,  $B_{d\_c}$ ,  $C_{_c}$ ,  $D_{_c}$ , and  $D_{d\_c}$  are formed for use in the MATLAB LQR routines. These matrices correspond to the states shown in Table 4.1; the transformation to form the rotor first symmetric flap mode from the blade 1 and 2 first flap modes is applied. In addition, the generator azimuth state is eliminated and the generator speed state is retained.

After checking for controllability, the routine performs LQR using the calculated  $A_c$  and  $B_c$  matrices, as well as the values input for  $R$  and  $Q$ .

We choose weights in  $Q$  by a trial-and-error approach. First, we choose a set of weights and run the control synthesis routine in MATLAB. Then we note the location of the resulting closed-loop poles. We repeat this procedure by adjusting the different weights until we obtain the desired closed-loop poles.

For example, with  $R=1$  and  $Q = \begin{bmatrix} 0.00001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$ , the MATLAB LQR

routine calculates closed loop poles at  $-1.99 \pm 22.87i$ ,  $-4.18 \pm 13.33i$ , and  $-2.00$ . The first pole pair corresponds to the first drive train torsion mode, the second pole pair corresponds to the first rotor symmetric flap mode, and the fifth pole corresponds to the generator speed state. The open-loop values for these poles are  $-0.26 \pm 22.56i$ ,  $-3.61 \pm 13.50i$ , and  $-0.19$ . We moved the poles further to the left in the complex plane with this control to improve damping and transient response.

The resulting gain matrix is

$$G = [-14.25 \quad 0.049212 \quad 0.70936 \quad -0.12907 \quad 0.0066814].$$

Next, the gain corresponding to the step wind disturbance is calculated in the control synthesis routine. Recall our general disturbance model:

$$\begin{aligned} \dot{\underline{z}}_d(t) &= F\underline{z}_d(t) \\ \underline{u}_d(t) &= \Theta\underline{z}_d(t) \end{aligned} \tag{4.2}$$

For step wind disturbances, it can be shown that  $F = 0$  and  $\Theta = 1$  (17).

Assuming step wind disturbances, the MATLAB script then calculates the wind disturbance gain  $G_d$  by  $G_d = -B^{-1}B_d\Theta = -B^{-1}B_d$ , where  $B$ , and  $B_d$  are as in equation 4.1),  $\Theta$  is as in equation 4.2 (equal to 1 for step disturbances), and  $G_d$  is the gain corresponding to the wind disturbance state in the feedback law  $\underline{u}(t) = G\underline{x}(t) + G_d\underline{z}_d(t)$  (17).

At this point, one has designed a full state-feedback controller with the feedback law

$$\underline{u}(t) = G\underline{x}(t) + G_d\underline{z}_d(t). \tag{4.3}$$

To implement a full-state feedback controller on a real turbine, all states contained in the full state feedback control law (4.3) must be measured. This is not practical in commercial turbine applications. Typical commercial turbine sensors include a rotor or generator rotational speed sensor, blade strain gages to measure blade flap- or edgewise-bending moments, tower-top accelerometers, etc.

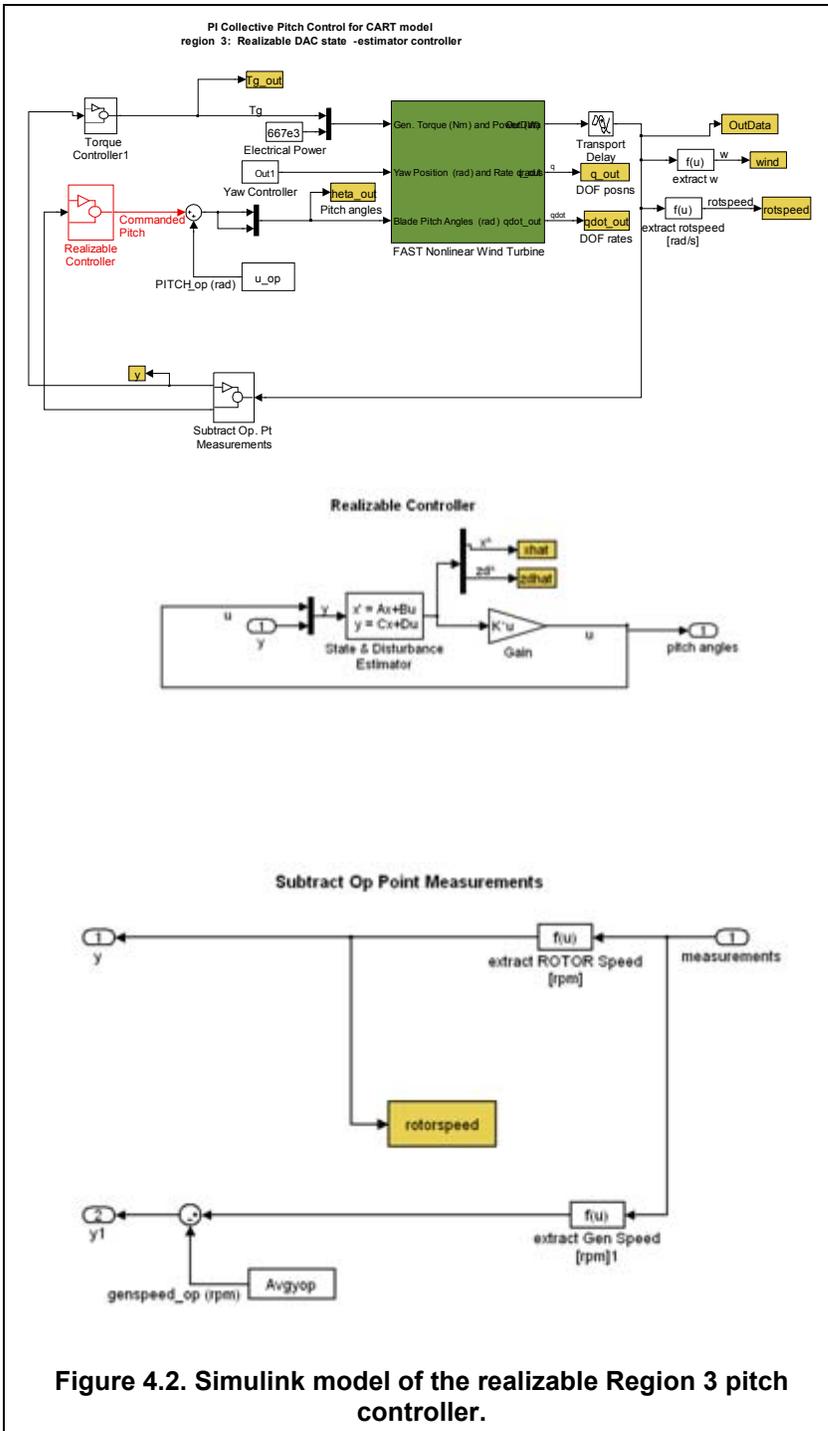
To circumvent this difficulty, we use state estimation, which is based on limited turbine measurements. We can design a state estimator controller to perform the same function as the full state feedback controller.

We now describe the state estimation realizable controller.

#### ***4.1.4 State Estimation Realizable Controller***

We now design a state estimator controller to perform the same function as the full state feedback controller just designed.

The MATLAB script (see Appendix B) is used to design this state estimator controller, using DAC. This controller design is based on only generator or rotor speed as the control measurement. The steps involved in producing this state estimator controller start with design of the full state feedback controller in the MATLAB script. Now, after these steps, the additional steps to produce the state estimator controller include calculating state estimator gains and forming a state space model of just the controller. All these steps are performed in the MATLAB script. Once the state estimator controller has been designed, a state space model of the controller alone is calculated in the MATLAB script. This model is then inserted into the Simulink model (see Figure 4.2) in the Realizable Controller box.



**Figure 4.2. Simulink model of the realizable Region 3 pitch controller.**

Let us first describe calculation of the state estimator gains in the MATLAB script. We use pole placement to perform this step in the MATLAB script file. The state estimator poles are placed by a trial-and-error approach. We select the pole locations for these state estimates and then run the control synthesis routine to obtain the resulting controller. We then simulate the closed-loop system by inserting the controller into the loop. We then adjust state estimator pole locations to improve controller performance.

A general rule is to start by placing the state estimator poles further to the left in the complex plane than the states being estimated. In this example, the closed-loop plant states have pole locations (by using the LQR routine) at  $-1.99 \pm 22.87i$ ,  $-4.18 \pm 13.33i$ , and  $-2.00$ . The state estimator poles are selected so their locations in the complex plane are further to the left (as indicated by the real parts of these poles) than the states being estimated (17). For example, we choose state estimator pole locations to be:

$pbar = [-15+22i \quad -15-22i \quad -9+13i \quad -9-13i \quad -10 \quad -11]$ . These poles correspond to the first drive train torsion mode, the rotor first symmetric flap mode, the generator speed, and the wind disturbance. This value for  $pbar$  is hardwired into the MATLAB script, but can be changed depending on the desired location for these poles. These pole locations will ensure rapid convergence of the state estimates to the turbine states being estimated. In the above example, we selected pole locations to give significant damping to the flexible modes, and to improve transient response. For more information, see (17).

Running this script results in state estimator gains:

$$Kbar = \begin{bmatrix} 0.0085488 \\ 0.87028 \\ 0.14816 \\ -9.6464e-005 \\ -7.0866 \\ 7.141 \end{bmatrix}.$$

After this step, the script augments the various matrices needed for the final controller design with augmented values corresponding to the wind disturbance state. The final step is to calculate the  $A$ ,  $B$ ,  $C$ , and  $D$  matrices for the controller alone (17).

These matrices are then imbedded in the pitch controller block in the Simulink model to be used for model simulation, which we now describe. This is shown in Figure 4.2 and labeled as Realizable Controller, highlighted in red.

One could also use LQR to design the state estimator gains. We will show some examples of this in a later section.

#### **4.1.5 Model Simulation With The State Estimator Controller**

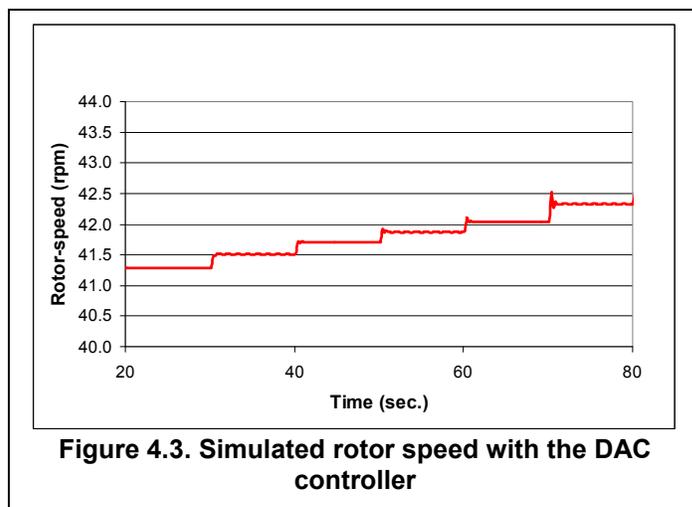
We will use the Simulink model (see Figure 4.2) to simulate the closed-loop system consisting of the turbine model and the controller just designed. In the upper part of the figure, the overall model includes the realizable pitch controller. The measured perturbed generator speed (difference between actual generator speed and reference generator speed set point) is passed into this block. This measured value is formed in the block called Subtract Op Pt Measurement; the details of this block are shown at the bottom of the figure (in this block we also measure rotor speed as needed by the generator torque controller). Figure 4.2 shows the realizable controller block and the state and disturbance

estimator. The disturbance state is already included in the state space model for the realizable controller. Two inputs are passed to this block: the measured perturbed generator speed, and the commanded blade pitch control input (17). The output from this block is then passed through a gain block, calculated in the MATLAB script. This gain is exactly the gain matrix  $G$  shown above (with the wind disturbance gain included).

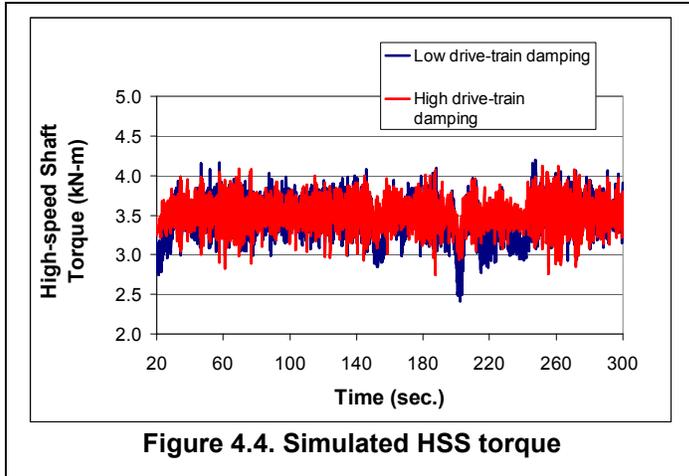
The resulting pitch angles that are produced in this block are small perturbations from the equilibrium value control design point. We add this equilibrium pitch value ( $u_{op}$  shown in the upper part of the figure) to achieve the total pitch, which is then passed into the FAST nonlinear wind turbine block. The final pitch angles passed into FAST have the unit radians (rad).

Speed regulation performance of the system excited by step winds is shown in Figure 4.3. These step winds start at 14 m/s at 20 s and ramp up to 24m/s at 70 s. The speed regulation to 41.7 rpm occurs only at the control design point for a wind speed of 18 m/s between 50 and 60 s.

Another control objective is to add active damping to the first drive train torsion mode. This should help mitigate drive train torsion loads. To test this capability we ran simulations with this closed-loop controller/ turbine system excited by turbulent winds.



We ran two controllers, each based on the linear model and control design techniques just described. One controller applied a large amount of active damping to the first drive train torsion mode and the second controller applied a reduced amount of drive train damping. Figure 4.4 shows the simulated HSS torque loads from the two simulations. In general, the controller applying the larger amount of damping results in reduced torque loads in the HSS. This demonstrates that actively damping this mode significantly mitigates the load.



In Chapter 5 we show field implementation and tests of this controller. We will see that this controller performs poorly when the turbine operates in the low wind speed end (12 to 13 m/s) of Region 3, close to Region 2½. This is caused by variation of control gain with turbine operating point and the fact that in this transition region, the control input gains are small and large pitch actuation rates are needed to actively damp this mode. As the wind speed increases and the turbine operates closer to the control design point (wind speed of 18 m/s, pitch angle of 11 degrees, rotor speed of 41.7 rpm), the controller performs as designed. To remedy this problem gain scheduling could be used to switch between a controller designed closer to the Region 2 to 3 transition point and the controller designed above. Gain scheduling for state-space controllers is an active area of research and needs more investigation before we can give guidelines for its use. Another method is to change actuators. Perhaps a different actuator will not have the issue of variation of control gains with turbine operating point.

Such an actuator is generator torque. As shown in (17), the control gains for this actuator are constant, not depending on turbine operating point. This makes this actuator a better choice for performing the function of active drive train damping. We now give an example of designing a generator torque controller having the objectives: (1) maintain nearly constant generator torque in Region 3; and (2) perform active drive train torsional damping in Region 3.

## 4.2 Region 3 Generator Torque Damping Control Design Example

Generator torque can also be used as a control input to add damping to the drive train torsion mode. Using generator torque to perform this function decreases the demand on the pitch control system. In this Region 3 control example, we employ two controllers: the pitch controller to regulate turbine speed, and the generator torque controller to perform active drive train damping. These two controllers are designed as separate control loops. We now demonstrate the control design and simulation for these controllers.

### **4.2.1 Control Objectives**

The primary goal in Region 3 is to maintain rated torque by maintaining nearly constant generator torque and using rotor collective pitch to regulate speed. The generator torque controller and the blade pitch controller are designed separately, as separate control loops.

The new part of this generator torque control is to add active damping of the first drive train torsion mode. We must still maintain nearly constant generator torque, because we are in Region 3. We now allow the generator torque to vary in small perturbations from the nominal constant torque value to actively damp drive train torsion (first mode).

Blade pitch control will still be used to regulate turbine speed in response to wind speed variations. Instead of using pitch control to add active damping to the drive train, we now perform this control objective with generator torque. In this way, demand is removed from the pitch control system for performing active drive train damping.

We have seen that one issue with blade pitch control is the variation of control gains with turbine operating point. This can lead to degradation of performance, as we have seen with the simple baseline PI controller in Chapter 3. It can also degrade performance of the controller designed in Section 4.2 in the transition region between Region 2 and Region 3. This is because the control input gains become very small in this region. The beauty of using generator torque control is that the control input gain is constant for all turbine operating points (17). This should result in improved performance, which we will see when we examine implementation results in Chapter 6 for this controller.

For the Region 3 pitch controller, one can use the designed baseline PI controller, as shown in Section 3.3. Alternatively, one can use a DAC state-space pitch controller as in Section 4.1. The DAC is redesigned with zero LQR weights on the states corresponding to the first drive train torsion mode so the pitch controller will not attempt to actively damp this mode (17). This function is assigned to the generator torque controller. We will use the baseline PI controller designed in Section 3.4 to regulate the overall speed in this section.

The generator torque controller design is based on a reduced state-space model that contains only the states needed to describe the first drive train torsion mode and generator speed.

### **4.2.2 Linear Model Description**

The generator torque feedback law is formed based on the states shown in Table 4.2. As described in (17), page 73, the control input  $\underline{\Delta u}$  shown in Equation (4.1) will be generator torque instead of rotor collective pitch.  $B$  will reflect the generator torque control input gain, which is now constant with turbine operating point. The disturbance input  $\underline{\Delta u_d}$  will not be accounted for in the generator torque control design, since it is already accounted for in the DAC pitch control design, or is compensated by the integral term in the PI controller designed in Section 3.4.1. We do not include a disturbance state in the control model for the Region 3 generator torque controller.

State	Description
$\Delta x_1$	perturbed drive train torsional deflection
$\Delta x_2$	perturbed generator rotational speed
$\Delta x_3$	perturbed drive train torsional velocity

We linearize the model about a turbine operating point that is identical for the case of the rotor collective pitch control designed in Section 4.2, namely:  $w_0 = 18$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 11$  degrees.

#### **4.2.3 FAST Linearization**

The FAST input file for this linearization is the same as shown in Appendix A, page A6, except that we have switched off the first flapwise blade mode DOF for this case. The Aerodyn input file for this linearization is the same as on page A9. The file containing FAST linearization parameters (cartlinear.dat) is the same as that listed on page A10, except we have set Trim Case equal to 2 to trim on generator torque. We have also set the input CntrlInpt to 3, reflecting generator torque as the control input instead of collective pitch.

With pitch set to 11 degrees and rotor speed set to 41.7 rpm in the FAST input file, and the wind speed constant at 18 m/s, FAST will trim to the correct generator torque for Region 3 (3524.7 N-m). Once a periodic operating point trim solution is determined, the simulation proceeds to obtain linear state-space matrices at several points around the rotor disk.

Running the MATLAB script eigenanalysis.m results in azimuth averaged state-space matrices. An eigenanalysis of the AvgAMat matrix gives us the open-loop poles for this linear model. They are located at  $-0.01 \pm 22.44i$  and  $-0.103$ . The first pole pair corresponds to the first drive train torsion mode; the last pole corresponds to the generator speed state. Again, the first drive train torsion mode is very lightly damped; the real part of this complex eigenvalue is  $-0.01$ . We hope to use generator torque control to increase the damping in this mode.

#### **4.2.4 Control Design**

The MATLAB script LQR\_design\_GenTorq\_3states.m that we use to design this controller is listed in Appendix B, page B4. This script can be used to design either a full state feedback controller or a realizable state estimator controller, based on measuring only generator speed.

The first part of the script reads state matrices produced by the FAST linearization simulation that we just ran. This script then forms azimuth-averaged state matrices in the same manner as the script `eigenanalysis.m`.

Next, the special matrices  $A_{_c}$ ,  $B_{_c}$ ,  $B_{d_{_c}}$ ,  $C_{_c}$ , are formed for use in the MATLAB pole placement or LQR routines. These matrices correspond to the states shown in Table 4.2. The generator azimuth state is eliminated and the generator speed state is retained in this linear model for control design. Again, as for the pitch control example of Section 4.2, if we retain the generator azimuth state but measure only generator speed, the resulting state-space system is unobservable (17). This means we cannot use state estimation to estimate plant states from the chosen measurement.

After reading in the azimuth averaged linear state-space matrices and checking for controllability, the routine uses the calculated  $A_{_c}$  and  $B_{_c}$  matrices to perform LQR, as well as the values input for  $R$  and  $Q$ . Choosing the weights in  $Q$  is an iterative process. First, we choose weights and then run the control synthesis routine. We observe the resulting values for the closed-loop poles. We repeat this process until the poles are located in the complex plane at the desired locations.

For example, with  $R = I$  and  $Q = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.00001 & 0 \\ 0 & 0 & 4000000 \end{bmatrix}$  the MATLAB LQR routine calculates  $G = [40.186 \quad 0.058941 \quad -1969.8]$ .

The closed-loop poles have the values  $-0.67 \pm 22.43i$  and  $-0.102$ . We have moved the poles corresponding to the first drive train torsion mode further to the left in the complex plane to add active damping. As we will see in Chapter 6 on implementing and field testing this controller, adding this damping reduces drive train torque loads.

Next we calculate state estimator gains. For this controller we do not include a wind disturbance state, so the number of state estimator gains is equal to the number of plant states. We again use pole placement to place the state estimator poles at:

$$pbar = [-15+22i \quad -15-22i \quad -11].$$

Placing the state estimator poles at these values results in the state estimator gains:

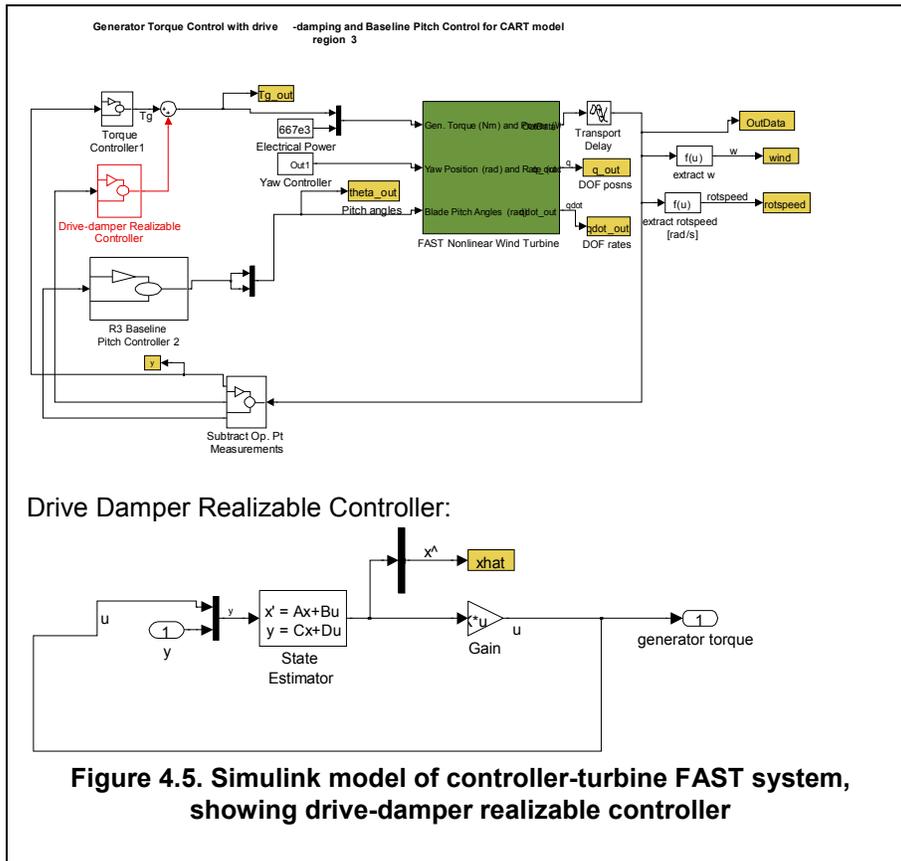
$$Kbar = \begin{bmatrix} 0.0031942 \\ 0.099168 \\ -0.073146 \end{bmatrix}.$$

The final step is to calculate the  $A$ ,  $B$ ,  $C$ , and  $D$  matrices for the controller alone, described in (17). These matrices are then imbedded in a torque controller block in the Simulink model now to be described.

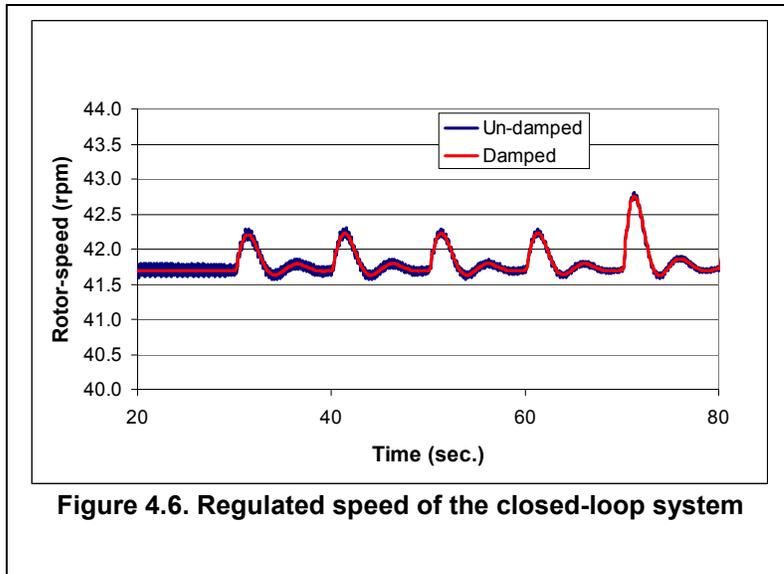
### 4.2.5 Control Simulation

We use a Simulink model to demonstrate simulation of this controller turbine system. Figure 4.5 shows the overall Simulink model connected to the FAST S-function. We highlight the new generator torque control drive train damper in red. We can see that it simply adds to the commanded generator torque signal coming out of the baseline torque controller designed in Section 3.3.

Figure 4.5 also shows inside the drive damper realizable controller box. Again, this box contains the state-space model of the controller. Inputs to this state-space controller are the perturbed generator speed and the commanded generator torque. Here,  $y$  is the measured generator speed coming from the Subtract Op. Pt. Measurements block. The generator torque comes from the commanded generator torque passing out of the state estimator block of the controller and then multiplied by the Gain block. In this diagram there are two inputs to the state estimator controller: the measurement  $y$  and the control input  $u$ .

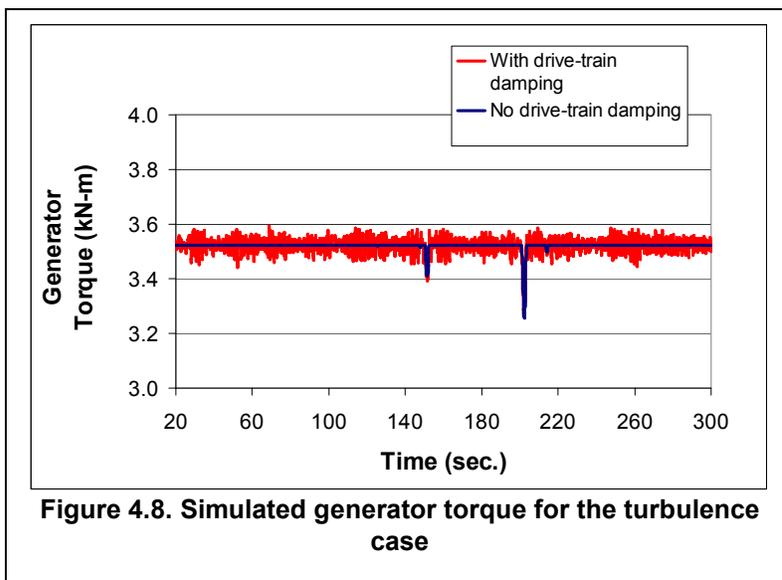
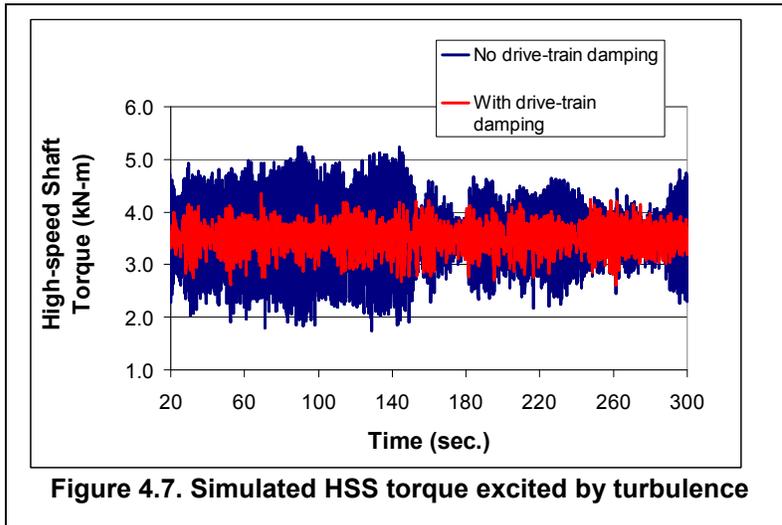


We simulate with the same step winds as for the rotor collective pitch controller (Section 4.2). Speed regulation is shown in Figure 4.6. We show one simulation with drive damping, and one without drive damping. The case without drive damping is easily simulated by cutting the connection between the Drive Damper Realizable Controller box and the summation point, which sums this output to the output from the Torque Controller1 baseline torque controller box. The damped case shows smoother rotor speed response compared to the undamped case because damping was added to the first drive train torsion mode.



A better measure of the drive train damping qualities of this generator torque controller can be seen by examining HSS torque (see Figure 4.7). Turbulent wind inputs have been used to drive this simulation. Figure 4.7 shows simulated HSS torque for the case with drive train damping and the case without drive train damping. We see a dramatic reduction in torque loads for the damped case versus the no damping case. The high-frequency fluctuations for the undamped case in Figure 4.7 are due to oscillations at the first drive train torsion natural frequency. By moving the poles corresponding to the first drive train torsion mode further to the left in the complex plane, the generator torque controller adds active damping to this mode; shaft torque loads show a corresponding decrease.

Figure 4.8 shows commanded generator torque for these two cases. Of course for no drive train damping, the generator torque is constant, except when the wind speed decreases so that the turbine transitions into Region 2. For the damped case, the generator torque consists of the mean plus small perturbations at the first drive train torsional frequency because of the added requirement of active damping of this mode.



As these two examples show, advanced controls can be designed to actively damp low-damped turbine modes. For a mode such as the first drive train torsion mode, generator torque may be a better actuator choice than blade pitch, as the control input gains do not vary with turbine operating point.

Another control objective may be to mitigate loads caused by asymmetric wind variations across the rotor disk. Collective pitch control or generator torque control do not meet this objective. We need to vary the pitch of each blade independently. We now look at the design and simulation of independent blade pitch control.

### 4.3 Independent Blade Pitch Control

Independent blade pitch can be used to mitigate turbine blade loads in the presence of wind shear and other asymmetric disturbances across the rotor disk. In (17), independent blade pitch was used in combination with a specialized wind disturbance waveform, which represents the linear shear variation across the rotor disk. This controller was

shown to mitigate the once per revolution (1P) blade loads caused by wind shear (17). Here we design one control loop to perform the Region 3 pitch control objectives: regulate rotor speed and mitigate asymmetric wind variations across the rotor disk. We now show detailed design and simulation of this independent blade pitch control system.

#### **4.3.1 Control Objectives**

The primary goals in Region 3 are to maintain rated torque and to regulate speed with blade pitch.

The new part of this control is to add independent blade pitch to mitigate asymmetric wind variations across the rotor disk. A rotor collective pitch component will still be implicit in the independent pitch controller that mitigates the uniform wind disturbance and attempts to regulate turbine speed (we do not employ a separate control loop for the collective pitch). Now, the pitch of each blade will be controlled independently to mitigate asymmetric wind variations across the rotor disk. This control requires an additional measurement to ensure observability of the state-space system (17). Here we add measurement of the blade-tip flap deflections, but we could also measure the flap-bending moments on each blade.

The design of this controller will follow exactly the description given in (17), pages 88-95. We will illustrate obtaining the linear model, designing the controller, and simulating this closed-loop system in the presence of wind disturbances that are asymmetric across the rotor disk.

#### **4.3.2 Linear Model Description**

The independent blade pitch feedback law is formed based only on the states in FAST delineated in Table 4.3. The control input  $\underline{u}$  will be the pitch of each blade instead of rotor collective pitch. Now,  $B$  will have two entries, reflecting the pitch control input gain of each blade (which should be equal unless there are slight variations in blade properties). In addition, the disturbance input  $\underline{u}_d$  will consist of two values: (1) a disturbance describing the linear shear variation across the rotor disk; and (2) the uniform wind disturbance already described in section 4.2. For more details, see (17), pages 84-95.

We will linearize the model about a turbine operating point that is identical for the case of the rotor collective pitch control designed in Section 4.1:  $w_0 = 18$  m/s,  $\Omega_0 = 41.7$  rpm, and  $\theta_0 = 11$  degrees. This linearization point is chosen for the same reasons as described in the last two control design examples.

State	Description
$\Delta x_1$	perturbed blade1 1 <sup>st</sup> flap deflection
$\Delta x_2$	perturbed blade2 1 <sup>st</sup> flap deflection
$\Delta x_3$	perturbed generator rotational speed
$\Delta x_4$	perturbed blade1 1 <sup>st</sup> flap velocity
$\Delta x_5$	perturbed blade2 1 <sup>st</sup> flap velocity

### **4.3.3 FAST Linearization**

The FAST linearization input file for this case is the same as shown for the last case in Appendix A, page A6. Now however, we switch off the drive train rotational flexibility DOF (DrTrDOF). We also add to the output list the outputs: TipDxc1 and TipDxc2 – the blades 1 and 2 out of plane tip deflections. These outputs will be needed in the Simulink model to simulate the closed-loop system.

In this case we trim on rotor collective pitch even though we are designing controls for independent pitch. The fact that we now have individual pitch is reflected by now having two control inputs instead of just one, as specified by setting NInputs = 2. Now, the parameter CntrlInpt is set to 5 and 6, reflecting the individual pitch of blade 1 and individual pitch of blade 2.

We must mention that in the FAST linearization, we use only one wind disturbance input, namely horizontal hub-height wind speed by setting NDisturbs = 1 and Disturbnc = 1. Linearization with just this disturbance gives us enough information to form the disturbance gains for the linear shear variation across the rotor disk (17). We do not request the vertical power law wind shear disturbance in the FAST linearization.

We run the FAST linearization, trimming to an equilibrium point with a generator torque of 3525.7 N-m at a wind speed of 18 m/s, pitch angle 11 degrees, and rotor speed 41.7 rpm. After running this simulation, we invoke the MATLAB script eigenanalysis.m, which calculates azimuth averaged state matrices contained in 4.1.

An eigenanalysis of the AvgAMat matrix gives us the open-loop poles for this linear model. They are located at  $-3.63 \pm 13.53i$ ,  $-3.64 \pm 13.28i$ , and  $-0.194$ . The first pole pair corresponds to the rotor first symmetric flap mode, the second corresponds to the rotor first asymmetric flap mode, and the last pole to generator speed state. Both symmetric and asymmetric rotor flap modes have significant damping, as can be seen by the real parts of these poles ( $-3.63$  and  $-3.64$ ). The goal in this control design is not to increase the damping in these modes, but to attenuate the effects of 1P disturbances (caused by the linear shear variation). The blade first flap modes are included in this model because measuring the displacement of the rotor asymmetric mode (or some other measure of the effects of the asymmetric wind variation across the disk such as difference in blade root

flap-bending moment between blade 1 and blade 2) ensures observability of this state-space model (17). Inclusion of the disturbance state describing the linear shear variation across the rotor disk increases the number of measurements needed to ensure observability of this state-space system.

#### 4.3.4 Control Design

The MATLAB script `LQR_design_IndepPitch8states.m` that we use to design this controller is listed in Appendix B, beginning on page B7. This script can be used to design either a full state-feedback controller or a realizable state-estimator controller, based on measuring generator speed and blade tip deflections (17).

The first part of the script reads state matrices produced by the FAST linearization simulation that we just ran. This script then forms azimuth-averaged state matrices in the same manner as the script `eigenanalysis.m`.

Next, the special matrices  $A_c$ ,  $B_c$ ,  $B_d_c$ , and  $C_c$  are formed for use in the MATLAB pole placement or LQR routines. These matrices correspond to the states shown in Table 4.3. As in the previous control design examples, the generator azimuth state is eliminated and the generator speed state is retained.

After checking for controllability, the routine uses the calculated  $A_c$  and  $B_c$  matrices, as well as the values input for  $R$  and  $Q$ , to perform LQR.

For example, with  $R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  and  $Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & .00001 & 0 \\ 0 & 0 & 0 & 0 & .00001 \end{bmatrix}$

the MATLAB LQR routine calculates:

$$G = \begin{bmatrix} 0.0111 & 0.0111 & 0.6314 & 0.0021 & 0.0015 \\ 0.0111 & 0.0111 & 0.6314 & 0.0015 & 0.0021 \end{bmatrix}.$$

There are two channels of gains, one for each blade, as there are now two control inputs. These gains result in closed-loop poles placed at  $-4.11 \pm 13.55i$ ,  $-3.92 \pm 13.20i$ , and  $-1.85$ . We have moved the pole corresponding to the generator speed state further to the left in the complex plane to improve transient response.

Next we calculate state estimator gains. We use LQR to perform this step in the MATLAB script file. We use two measurements for state estimation: measured generator speed and the measured tip deflection of the first asymmetric flap mode (17), namely:

$y = \begin{bmatrix} 0 & 0 & 412.2 & 0 & 0 \\ 0.4873 & -0.4873 & 0 & 0 & 0 \end{bmatrix}$ . The first row of  $y$  reflects the measurement of generator speed. The value 412.2 reflects that we measure the generator rotational speed on the HSS side in rpm. The generator speed state has the units rad/s and is measured on the low-speed side. The second row reflects measurement of the first asymmetric flap mode tip deflection, which is a linear combination of the states contained in the linear model, as  $\frac{1}{2}(\Delta x_1 - \Delta x_2)$ .

For the LQR state estimator gain calculation, we set  $Re = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  and

$$Qe = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & .0006 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}, \text{ where } Re \text{ and } Qe \text{ are the state}$$

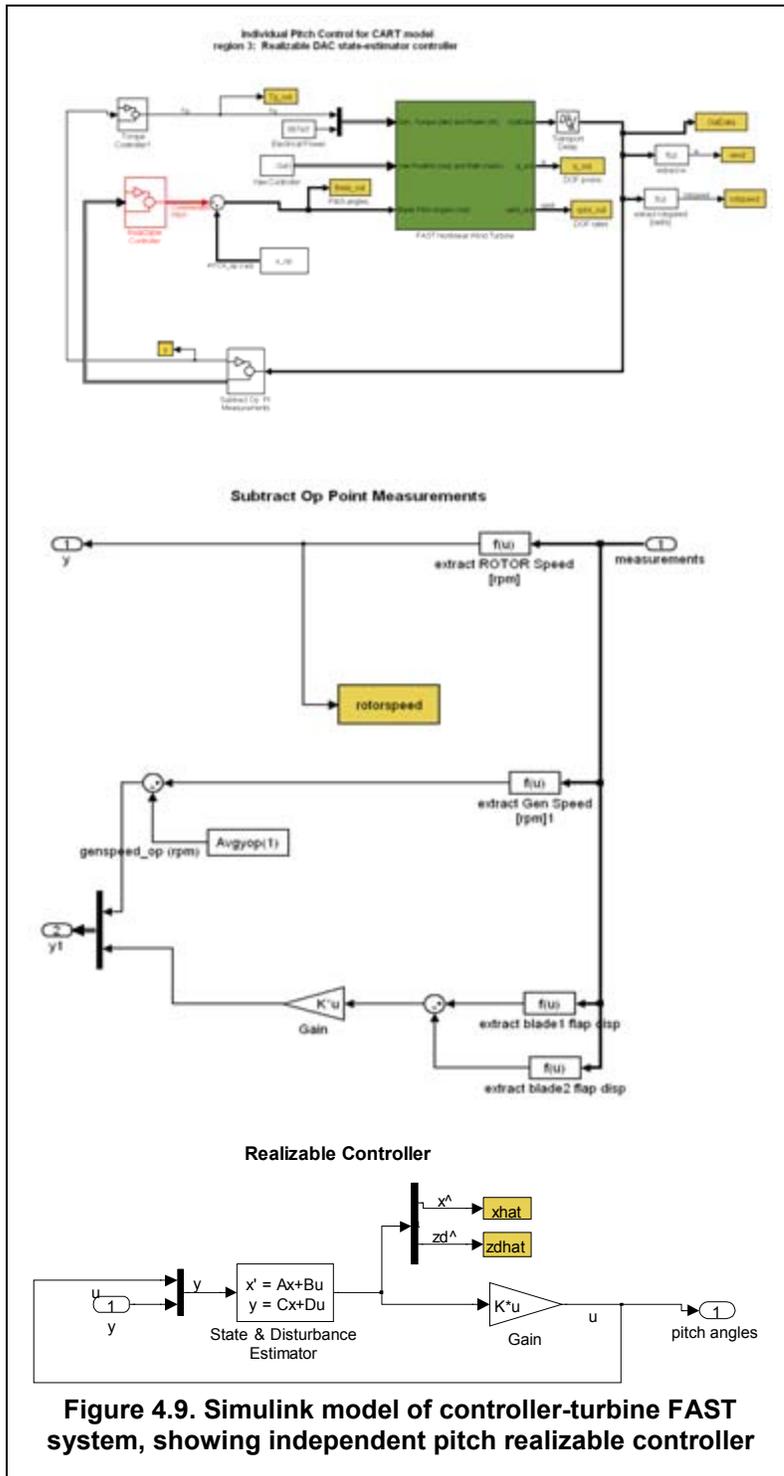
estimator weightings in the MATLAB LQR routine. This gives state estimator gains:

$$Kbar = \begin{bmatrix} 0.24171 & 3.0814 \\ 0.24181 & -3.0814 \\ 0.041862 & -1.1589e-007 \\ 0.35212 & 4.6269 \\ 0.35553 & -4.6269 \\ -0.0012259 & 38.704 \\ -0.009323 & -22.434 \\ 3.1623 & 0.00012951 \end{bmatrix}.$$

After this step, the script augments the various matrices needed for the final controller design with augmented values corresponding to the wind disturbance states. The final step is the calculation of  $A$ ,  $B$ ,  $C$ , and  $D$  for the controller alone. These matrices are then imbedded in the pitch controller block (Realizable Controller) in the Simulink model to be used for model simulation, which we now describe.

### 4.3.5 Control simulation

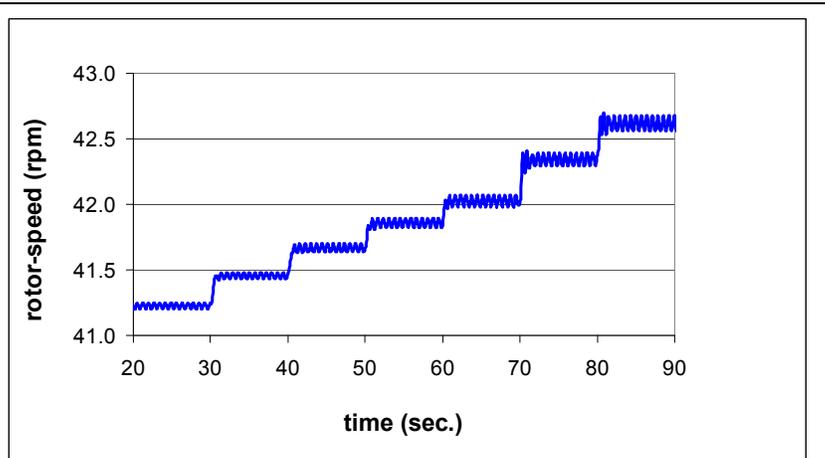
We use a Simulink model to simulate this controller-turbine system. Figure 4.9 (upper figure) shows the overall Simulink model connected to the FAST S-function. We highlight the part added here in red. Figure 4.9 also shows the Subtract Op Point



**Figure 4.9. Simulink model of controller-turbine FAST system, showing independent pitch realizable controller**

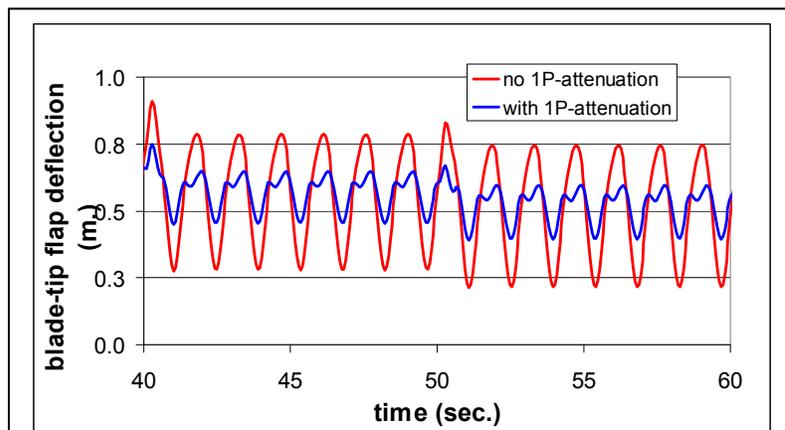
Measurements block and the Realizable Controller block. The measurement is the perturbation in generator speed (perturbation from the operating point speed value) and the perturbed tip deflection of the first asymmetric flap mode (17).

We simulate with a wind file that has step winds plus a power-law vertical wind shear. Speed regulation is similar to the regulation achieved with the rotor collective pitch controller designed in Section 4.1, as reflected in Figure 4.10.



**Figure 4.10. Simulated Region 3 rotor speed using independent pitch controller.**

The second control objective here is mitigation of the effects of wind shear variations across the rotor disk. This disturbance mitigation is shown in Figure 4.11, where we plot blade-tip flap deflection. We show two plots, the first with no linear shear attenuation (no 1P-attenuation), and the second with linear shear disturbance attenuation (with 1P-attenuation). To obtain the case with no 1P-attenuation, we reran the control synthesis routine with the disturbance gain corresponding to the shear variation set to zero. Setting this gain to zero is equivalent to running a collective pitch controller, because differential



**Figure 4.11. Simulated blade-tip flap deflection excited with step wind inputs**

pitch is commanded only to mitigate the 1P disturbance. We can see the dramatic reduction in the blade-tip flap deflection when the shear disturbance is attenuated.

In this chapter we have presented the design and simulation of three state-space controllers. Implementing such controllers on a field test turbine presents additional challenges and complexities. We now describe implementation of a DAC rotor collective pitch controller for control in Region 3 in Chapter 5, and a generator torque controller for actively damping first drive train torsion (in Regions 2 and 3) in Chapter 6. We now describe implementation of the DAC pitch controller.

## 5. Implementing and Testing a Region 3 Rotor Collective Pitch Controller for the Controls Advanced Research Turbine

### 5.0 Introduction

In this chapter we discuss the implementation and testing of the Region 3 rotor collective pitch controller developed in Chapter 4 (Section 4.1). We show the steps we have taken to design this controller and perform simulation tests. We describe implementation and preliminary field tests of this controller.

### 5.1 Control Design and Implementation

The control objectives are to regulate turbine speed in Region 3 (assuming constant generator torque) and to actively damp the first drive train torsion mode. At first it seemed possible to directly implement the controller designed in Section 4.1. We learned through examination of CART test data that we needed to include an actuator dynamics model in the control loop for this machine. We now describe this model and how we incorporate it into the control design model.

#### 5.1.1 Actuator Model

The dynamics of an electromechanical pitch servo are influenced by many subsystems, including internal servo controllers (converting pitch rate commands to motor current), motor dynamics, blade torsional stiffness, and aeroelastic coupling. Individual servomotors pitch each blade based on pitch rate (not pitch angle). Separate hardware controllers convert these pitch rate signals into pitch torque commands and, at a lower level, motor current. A model that includes all these elements could be complex. Alternatively, because the actuator dynamics are fast relative to those of the rest of the turbine, we use a simple linear model relating pitch rate commands and actual pitch rates (we are working with pitch rates, not pitch angles; we need a model relating commanded and actual pitch rates). From least-squares fitting to field-test data of the CART, a time constant for this model is calculated to be 1/60 s (10). The dynamics of the pitch actuator can be simplified to the following first-order transfer function involving pitch rate

$$\frac{\dot{\theta}(s)}{\dot{\theta}_{com}(s)} = \frac{60.0}{s + 60.0} \quad (5.1)$$

where  $\dot{\theta}_{com}(s)$  is the commanded pitch rate.

It is important to include the actuator effects in the control design so these dynamics can be accounted for. This prevents us from directly implementing the controller designed in Section 4.1. If we do not include these dynamics, the control performance will be different than we expect. One way of accounting for the actuator effects is to include these dynamics in the overall linear model used for control design. We append the linear model with states representing the actuator states. A linear model of the actuator dynamics is needed. We modify the overall state matrices with the necessary entries to add the linear dynamics of the actuator. This requires us to redesign the controller of

Section 4.1, now that we have additional states in the linear model for control design (the actuator states just described).

We can proceed to design the resulting controller (including actuator effects) by using full-state feedback just as we did for the simpler controller of Section 4.1. The main stipulation is that the composite system (made up of the turbine states plus actuator states) must be controllable. In addition, if we want to use state estimation, the resulting composite system must also be observable. We now describe this composite linear model.

### 5.1.2 Linear Model

We first review the linear model of the turbine plant without actuator states. This linear model can be expressed in state-space form as

$$\begin{aligned}\Delta \dot{\underline{x}}_m &= A_m \Delta \underline{x}_m + B_m \Delta \underline{u}_m + B_{d_m} \Delta \underline{u}_d \\ \Delta \underline{y}_m &= C_m \Delta \underline{x}_m\end{aligned}\tag{5.2}$$

where,

$\Delta \underline{x}_m$  is the state vector,

$\Delta \underline{u}_m$  is the rotor collective pitch control input,

$\Delta \underline{u}_d$  is the uniform wind disturbance input,

$\Delta \underline{y}_m$  is the measured output,

$A_m$  is the state matrix,

$B_m$  is the control input distribution matrix,

$B_{d_m}$  is the disturbance input distribution matrix, and

$C_m$  relates the measured output  $\Delta \underline{y}_m$  to the turbine states.

The states, control input, disturbance input, and measurement are perturbations from the model linearization point.

As discussed, we model the collective pitch actuator as a first-order linear system, which determines pitch rate given pitch rate commands. Because we are interested in pitch angle as the interface to our plant model, we construct a state-space actuator model with two states (pitch angle and pitch rate). The input to this model will be commanded pitch rate, and the output will be blade pitch angle.

Such a model is described by:

$$\begin{aligned}\Delta \dot{\underline{x}}_a &= A_a \Delta \underline{x}_a + B_a \Delta \underline{u}_a \\ \Delta \underline{y}_a &= C_a \Delta \underline{x}_a\end{aligned}\quad (5.3)$$

where,

$$\Delta \underline{x}_a = \begin{bmatrix} \Delta \theta \\ \Delta \dot{\theta} \end{bmatrix}, \Delta \underline{u}_a = \Delta \dot{\theta}_{com}, \Delta \underline{y}_a = \Delta \theta, \quad (5.4)$$

$$A_a = \begin{bmatrix} 0 & 1 \\ 0 & -60 \end{bmatrix}, B_a = \begin{bmatrix} 0 \\ 60 \end{bmatrix}, C_a = [1 \quad 0].$$

Here,

$\Delta \theta$  is blade pitch angle (perturbation),

$\Delta \dot{\theta}$  is blade pitch rate (perturbation),

$\Delta \dot{\theta}_{com}$  is commanded blade pitch rate (perturbation),

$\Delta \underline{x}_a$  is the actuator linear model state vector,

$\Delta \underline{u}_a$  is the input to this model, and

$\Delta \underline{y}_a$  is the output from the model.

Combining the plant model Equation (5.2) and actuator model Equation (5.3) gives:

$$\begin{aligned}\Delta \dot{\underline{x}} &= A \Delta \underline{x} + B \Delta \underline{u} + B_d \Delta \underline{u}_d \\ \Delta \underline{y} &= C \Delta \underline{x}\end{aligned}\quad (5.5)$$

where  $A = \begin{bmatrix} A_m & B_m C_a \\ 0 & A_a \end{bmatrix}$ ,  $B = \begin{bmatrix} 0 \\ B_a \end{bmatrix}$ ,  $C = \begin{bmatrix} C_m & 0 \\ 0 & C_a \end{bmatrix}$ , and  $B_d = \begin{bmatrix} B_{d_m} \\ 0 \end{bmatrix}$ .

Also,  $\Delta \underline{x} = \begin{bmatrix} \Delta \underline{x}_m \\ \Delta \underline{x}_a \end{bmatrix}$ ,  $\Delta \underline{y} = \begin{bmatrix} \Delta \underline{y}_m \\ \Delta \underline{y}_a \end{bmatrix}$ , and  $\Delta \underline{u} = \Delta \underline{u}_a = \Delta \dot{\theta}_{com}$ .

We have also incorporated into this model the same wind disturbance state described in Section 4.1, representing the hub height component of uniform wind over the rotor disk.

If the system is controllable and observable, the controller can be designed to achieve desired performance. The final linear model we use to design this controller for implementation on the CART contains the following turbine states:

$$\begin{aligned}\Delta x_1 &= \text{perturbed drive train torsional deflection,} \\ \Delta x_2 &= \text{perturbed rotor first symmetric flap mode displacement,}\end{aligned}$$

- $\Delta x_3$  = perturbed generator rotational speed,
- $\Delta x_4$  = perturbed drive train torsional velocity,
- $\Delta x_5$  = perturbed rotor first symmetric flap mode velocity,
- $\Delta x_6$  = perturbed actuator pitch angle, and
- $\Delta x_7$  = perturbed actuator pitch rate.

The first five states are generated from the FAST linear model, as described in Section 4.1; the last two are added on in the MATLAB controller synthesis routine used to design this controller. This linear model is generated at a wind speed of 18 m/s with a pitch angle of 11 degrees and a rotor speed of 41.7 rpm. Eigenanalysis of the resulting  $A$  matrix shows that the open-loop poles are located at  $-0.02 \pm 22.6i$ ,  $-3.6 \pm 13.5i$ ,  $-0.1$ ,  $0$ , and  $-60$ . The first pole pair corresponds to the first drive train torsion mode; the second corresponds to the rotor first symmetric flap mode. The next pole corresponds to the generator speed state. The next two poles (at  $0$  and  $-60$ ) correspond to the actuator states. The first drive train torsion mode is very lightly damped, by noting the real part of this complex eigenvalue ( $-0.02$ ).

The advantage of using state-space control design is that the damping in the first drive train torsion mode can be increased by moving the poles further to the left in the complex plane. This is straightforward in state-space control design. It is much less clear how to accomplish this goal with classical control design methods.

Because of limited turbine measurements, state estimation must be used to estimate plant states. For the FAST model, successful state estimation was applied by using the following turbine measurements:

- $\Delta y_1$  = perturbed HSS rotational speed, and
- $\Delta y_2$  = perturbed pitch angle.

Previous studies using FAST included only a measurement of generator speed for state estimation, as illustrated in the control design example of Chapter 4. Now that actuator states are included in the model for control design, this additional pitch angle measurement is needed to ensure this composite system is observable.

The FAST-generated linear model is used to design the control system. The objective is to regulate the speed at 41.7 rpm in Region 3, enhance the damping of the first drive train torsion mode, and stabilize the blade first flap.

After checking controllability of the FAST generated state-space system, we use LQR to calculate the gains. The resulting closed-loop system increases the damping in the first drive train torsion mode and improves the transient response of the other states, with poles now located at  $-1.3 \pm 22.6i$ ,  $-3.9 \pm 13.5i$ ,  $-2$ ,  $-10$ , and  $-60$ .

One difficulty with including an actuator dynamics model of the form modeled in Equation (5.4) is that the disturbance gain can no longer be calculated as we did using the expression in Section 4.1.3:

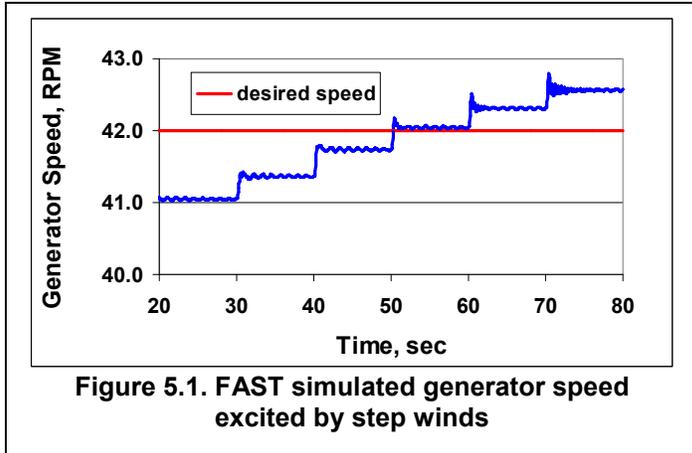
$$G_d = -B^{-1}B_d\Theta. \quad (5.6)$$

This is due to the zeros in each entry of B except the entry corresponding to the pitch rate state, because  $B = \begin{bmatrix} 0 \\ B_a \end{bmatrix}$ . Use of (5.6) to calculate  $G_d$  results in a value of zero. To resolve this issue we chose nonzero values of  $G_d$ , performed the control synthesis, then simulated the closed-loop system excited by step winds. We repeated this with different values of  $G_d$  until we achieved reasonable speed regulation performance. The final value we chose was 0.28. This way, selection of  $G_d$  becomes a trial-and-error approach, because a nonzero value cannot be calculated rigorously by Equation (5.6), as was done for the cases in Chapter 4.

State estimator gains are calculated by using pole placement that result in poles at  $-15 \pm 22.5i$ ,  $-4 \pm 13i$ ,  $-32$ ,  $-10$ ,  $-11$ , and  $-7$ , corresponding to the first drive train torsion mode, the first rotor symmetric flap mode, actuator deflection and rate, generator speed, and uniform wind disturbance.

## 5.2 Simulation Tests

Before implementing this controller in the CART, we tested it through FAST simulation. One implementation issue is steady-state speed errors. This issue is important to achieve good performance and to prevent triggering turbine overspeed in Region 3. Overspeed occurs in the CART when the rotor speed reaches 43 rpm. In this control design, the linearization point is at 18 m/s, a point midway between the highest and lowest wind speeds in Region 3. We can expect deviation from the desired 41.7 rpm set point for wind speeds above and below 18 m/s. To check this, test step winds are input to FAST, providing wind speeds below and above the 18 m/s linearization point. After implementing the controller into FAST and simulating closed-loop performance, Figure 5.1 shows that the generator speed varies from 41 rpm to 42.5 rpm. The simulated generator speed matches the 41.7 rpm set point only at a wind speed of 18 m/s (40 to 50 s). We can see that the controller maintains rotor speed below the 43 rpm overspeed point for this range of tested wind speeds from 16 m/s to 24 m/s, and with the choice of  $G_d = 0.28$ .



Additionally, we tested the closed-loop performance by using turbulent wind inputs to excite the turbine model. Figure 5.2 shows speed regulation for this case, again showing good speed regulation to 41.7 rpm and maintaining speed below the 43 rpm overspeed point. The controller should also reduce LSS torque loads because it was designed to increase damping of drive train torsion. We will show this capability when we compare preliminary controller test results to baseline controller results which does not include a drive train damping capability.

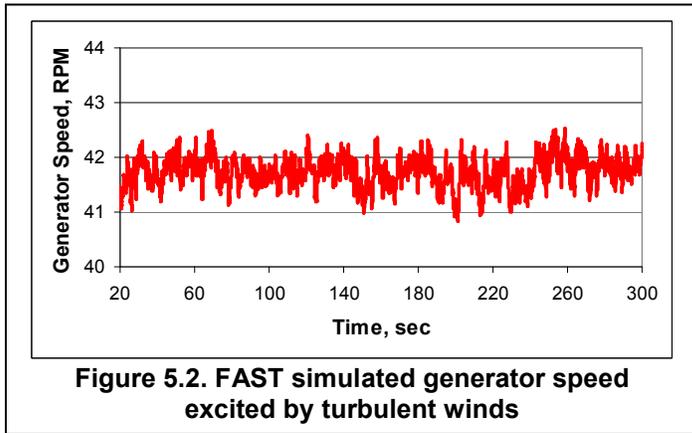
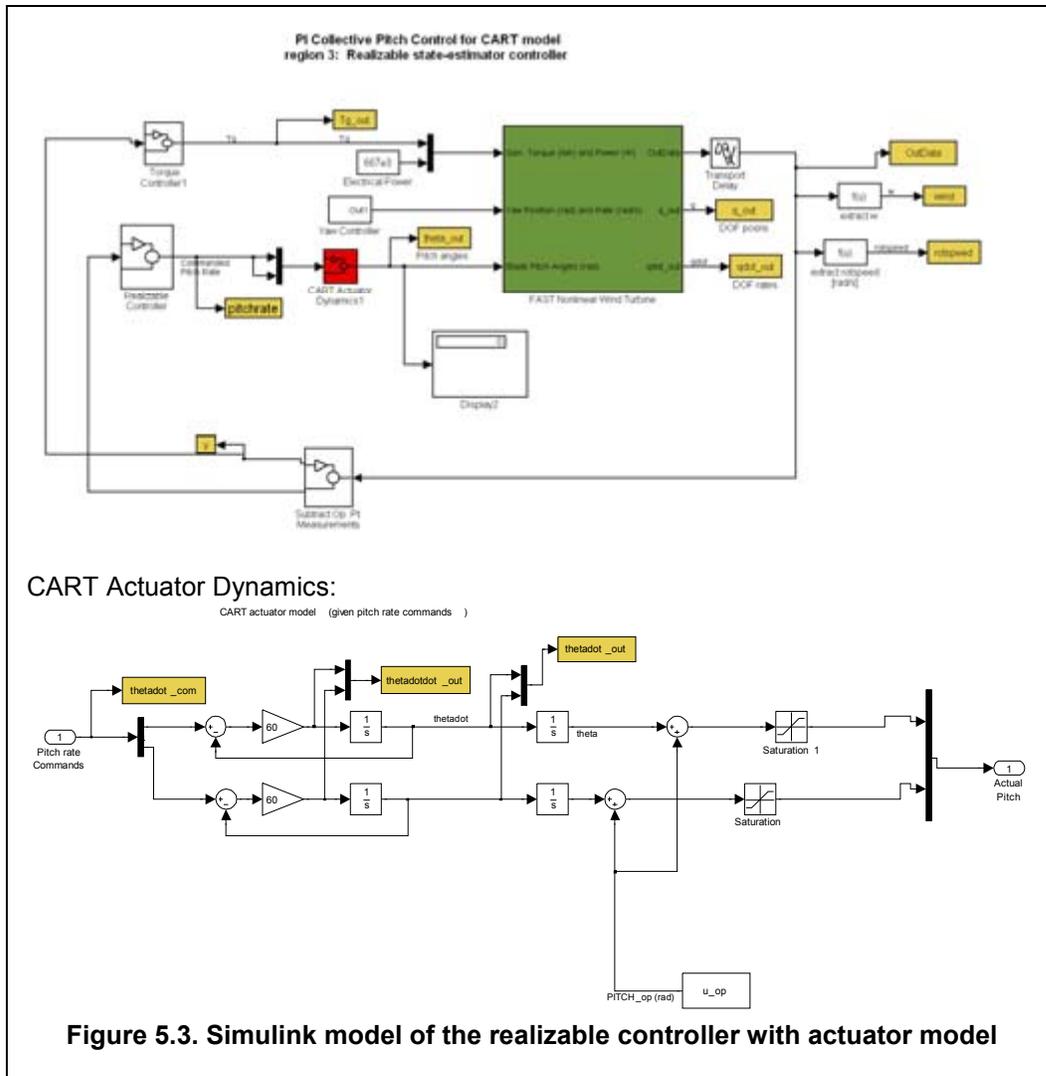


Figure 5.3 shows the Simulink model used to simulate this redesigned DAC controller. The new part to this model is the CART Actuator Dynamics1 block, modeling the pitch actuator. We show the details of this new block in the lower portion of the figure. We apply saturation and add the equilibrium pitch value inside this block. The input to this block is commanded pitch rate, passed from the realizable controller block. The output of this actuator block is then the pitch angle of each blade, which is then passed into the FAST S-function (green block). The realizable controller block is the same as shown for the controller of Section 4.1, except that the output from the realizable controller is pitch rate instead of pitch angle. Another difference is that we measure and pass blade pitch angle perturbations into the realizable controller block. This calculation is performed in the Subtract Op. Pt. Measurements block (not shown).

Adding the pitch actuator model into the model for control design and simulation was a significant implementation issue in implementing this controller on the CART. We now discuss further issues before discussing actual tests of this controller.



### 5.3 Further Implementation Issues

Another issue, already addressed with the FAST controller, is steady-state speed errors. An alternative method to address this issue is by feeding back the integral of speed error, as is achieved by the integral term in the PI baseline controller. This could be done by feedback of the generator azimuth error state, which is the natural integral of generator speed error. The consequence of this approach is that at low wind speeds, when the pitch angle can saturate at  $-1^\circ$ , the generator is running at just below rated speed and the azimuth error state increases in the negative direction. Another approach to resolve this problem is to incorporate an anti-windup term in the controller, similar to the anti-windup term added to the PI controller designed in Section 3.4. Simulation tests proved this to be a satisfactory approach, although we have not yet field tested this approach.

Another issue is the method in which HSS rotational speed is measured in the CART. The azimuth position of the HSS is first measured. HSS rotational speed is then derived from this measurement by taking its derivative and applying a low-pass filter to remove high-frequency quantization noise effects.

We have presented a state-space control methodology that assumes continuous time operation. However, the CART control algorithm must run on a digital computer that refreshes all measurements and command signals at the rate of 100 Hz, i.e. with a 0.01-s sampling period. To operate in discrete time, an equivalent control system is generated from the continuous time version. Once the continuous time state-space model or equivalent transfer function is calculated, MATLAB can be used to easily calculate an equivalent discrete-time state-space model.

Before implementing this controller into the CART, we investigated the effects of discretizing the continuous time controller into a discrete time controller. Using MATLAB to obtain the discrete time controller from the continuous time controller is straightforward. One simply invokes a MATLAB Control Toolbox (27) statement that transforms the continuous time system to discrete time such as:

$$[A_d, B_d, C_d, D_d] = \text{c2dm}(A, B, C, D, .01, 'zoh'),$$

where  $A_d$ ,  $B_d$ ,  $C_d$ , and  $D_d$  are the discrete time versions of the controller state space model  $A$ ,  $B$ ,  $C$ , and  $D$ . We specify the sample time of 0.01, as this is the sampling time in the controller software. The parameter 'zoh' reflects the method of converting to discrete time, assuming a zero-order hold on the inputs (27).

Another implementation issue was the effect of deriving HSS rotational speed from the HSS position measurement. We conducted FAST simulations with the HSS speed derived from the HSS azimuth position. In general, we found very little difference between this simulation and the simulation with HSS speed taken directly from FAST. After we conducted these thorough investigations, we proceeded to implement this controller in the CART.

Now we describe the last three steps involved with implementing and testing controls. The previously described controller is implemented into the CART control software in state-space form. Care is taken to be sure that all programming bugs are removed. Before this algorithm was field tested, a simple simulator built into the control software was used to test the control code. This step represents step 7 listed in Section 2.1: Bench test the controller hardware. This simulator integrates a single-state (rotor speed) and a lookup table for aerodynamic torque. These tests are useful for further catching C-code implementation bugs, highlighting region transition problems, and as an independent check of speed regulation performance. One weakness of this simulator is that it does not account for turbine flexible modes. In addition, because the simulator does not produce many of the measurement signals that the more complex state-space controllers assume, there are situations where the simulator predicts dynamic instability when the FAST

simulations do not. Still, this is a very useful test to conduct before the controller is tested on the real machine.

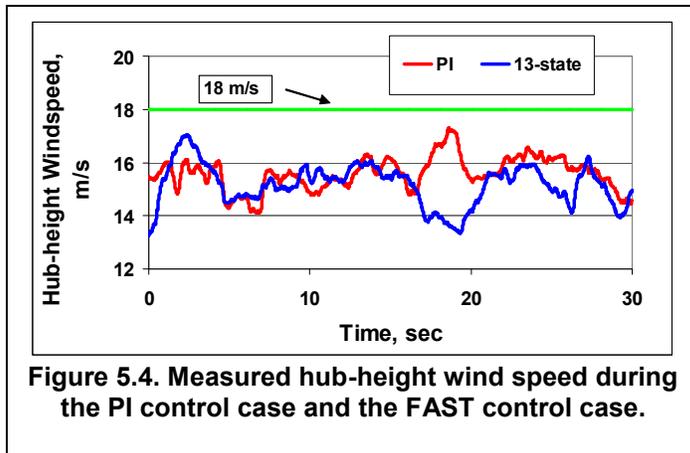
### 5.4 Field Test Results and Comparisons

We collected several 10-minute datasets on the CART while implementing the FAST controller, satisfying step 8 listed in Section 2.1. We then compared the test results to a case with simple PI baseline control to regulate speed in Region 3. Because these controllers were tested at different times, a direct comparison between results is difficult. Our objective was to show trends such as speed regulation and load mitigation. We attempted to examine both datasets and to extract smaller sections of data in which turbine operating parameters (such as wind speed, wind direction, and yaw error) are similar. The results of our comparison are preliminary, and we need many more hours of operation for realistic statistical comparisons between the controllers. In the next few paragraphs, data in the figures correspond to the same data summarized in Table 5.1.

**Table 5.1: Comparison between Baseline PI and FAST State-Space Controller**

Performance Measure	Baseline PI Controller	FAST State-Space Designed Controller
RMS speed error (RPM)	0.233	0.214
Max pitch rate (deg/s)	13.7	12.1
RMS pitch current (A)	28.8	20.7
Tower f-a fatigue DEL (kN-m)	578	525
LSS torque fatigue DEL (kN-m)	15.8	3.04
Blade-root flap fatigue DEL (kN-m)	126	135

Figure 5.4 shows the hub-height winds during turbine operation with these two controllers. Although the winds do not match, the deviations in wind speed for the two cases are similar; somewhat greater deviations occur during operation with the FAST controller.



**Figure 5.4. Measured hub-height wind speed during the PI control case and the FAST control case.**

Figure 5.5 shows speed regulation. Deviations in LSS rotational speed are seen using both controllers; the greatest deviations are seen for the PI controller. The FAST state-space controller seems to regulate speed to values below those of the PI controller. This is probably due to design of the controller at the 18 m/s and 41.7 rpm linearization point. We would expect regulated speed to be lower than 41.7 rpm for wind speeds lower than 18 m/s. Figures 5.4 and 5.5 show that as the wind speed increases to 18 m/s at about 23 s, the speed increases to about 41.7 rpm in that range.

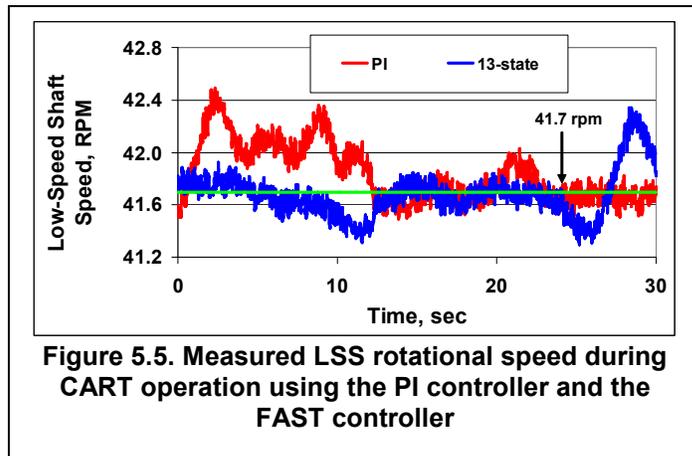


Figure 5.6 shows LSS torque during operation with each controller. The trend shows reductions in cyclic loads during operation with the state-space controller. This result is expected, because this controller was explicitly designed to increase damping in the first torsional mode of the drive train.

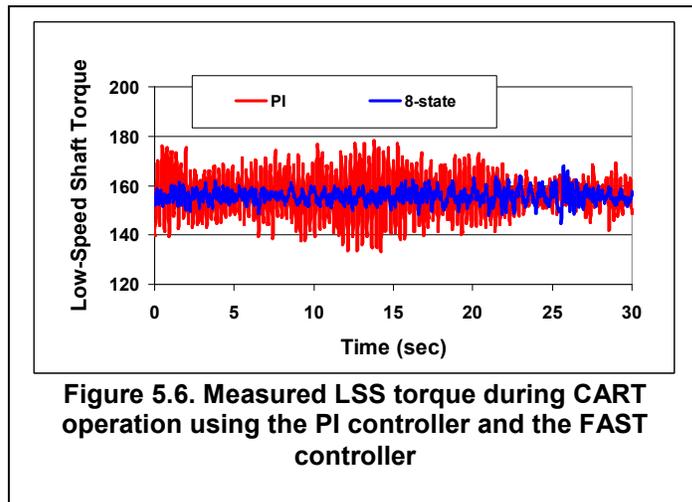


Figure 5.7 shows measured pitch rates from the CART during operation with each controller. The figure shows similar values of pitch rate for each controller. The trend indicates that the state-space controller, with its added drive train load mitigation, does not significantly increase pitch rates compared to the PI controller. Actual pitch rate

limits of  $\pm 18$  deg/s are implemented in the CART. As can be seen from the figure, these pitch rates are well within these limits.

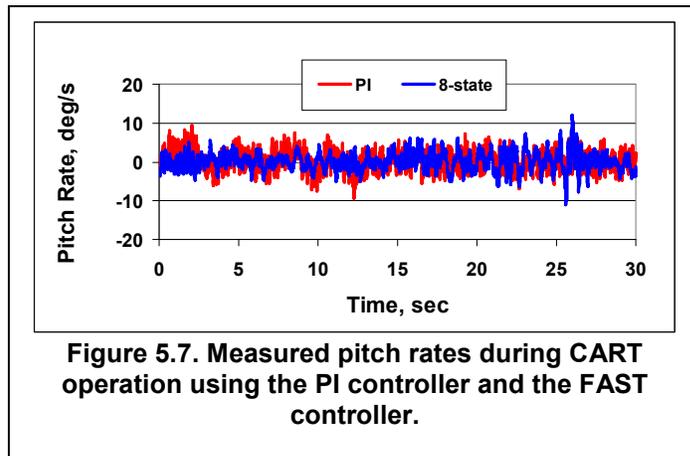
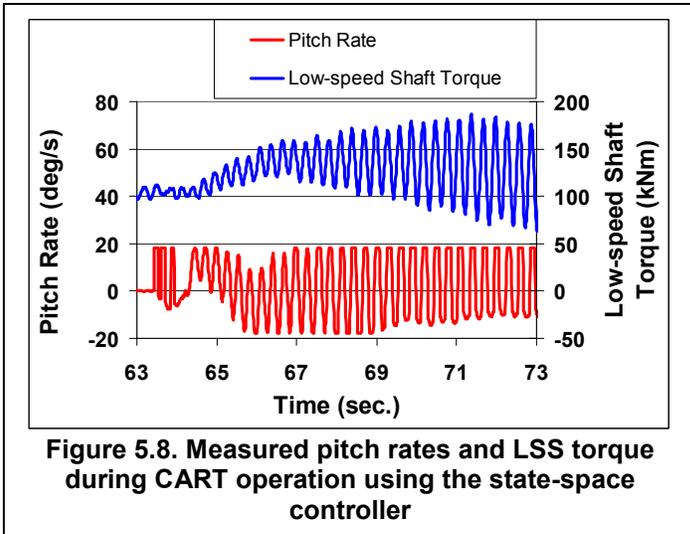


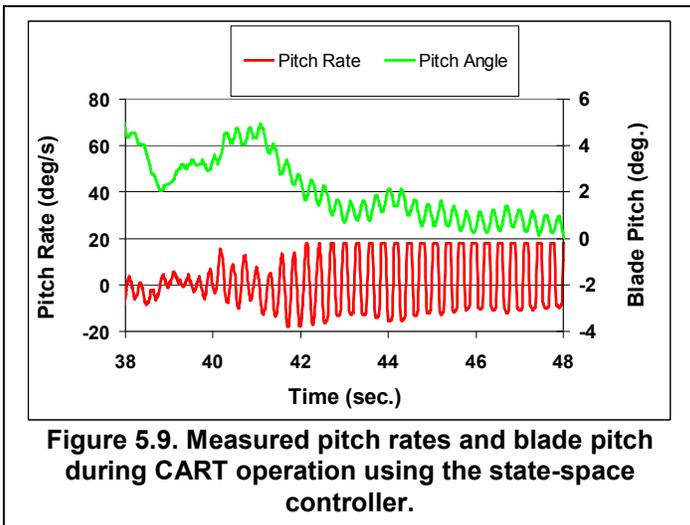
Table 5.1 compares statistics of the CART data during operation with the PI controller and the FAST controller. The FAST controller produces slightly lower RMS speed errors than the PI controller. The maximum pitch rate is slightly reduced with the FAST controller compared to the PI controller. The most notable difference between the two controllers is the reduction in LSS torque fatigue Damage Equivalent Loads (DEL) (28) using the FAST controller because of its design to add damping to the first drive train torsion mode. The blade root flap and tower f-a loads are comparable to the results from the PI controller.

## 5.5 Further Test Results

Further tests of this controller with increased gains showed undesirable behavior for some operating conditions of the CART. We tested the same controller but with increased weights on the states corresponding to the first drive train torsion mode. These increased weights increased the closed-loop damping so that the real part of the closed-loop poles had a value of -2 instead of -1.3 (for the controller implementation just described). When we tested this controller we observed undesirable behavior for wind speeds that resulted in the turbine operating at the low wind speed end of Region 3 (close to Region 2½). Figure 5.8 shows this behavior, showing pitch rates saturating at the upper and lower pitch rate limits ( $\pm 18$  deg/s) and LSS torque. The first drive train torsion is apparently being driven unstable by this control, just the opposite of what we intend.



We have already mentioned that the control gains vary with turbine operating point. We can expect different control performance for turbine operation in the low wind speed end of Region 3 (when the turbine passes from Region 2 into Region 3), compared to the upper end of Region 3. In this part of the operating region, the pitch angles are small, since in Region 2 the pitch is saturated at -1 degree and the pitch increases as we transition into Region 3. We designed this controller at a control design point which is in the upper wind speed part of Region 3. For such a control design point, the pitch angles are large, and we have good control authority because the control input gains are large. As the turbine operating point changes to the low wind speed end of Region 3, and the pitch angles decrease, we can expect a degradation in controller performance (we explained the reasons for this change in gains in Section 2.3.2 as shown in Figure 2.3). Indeed, this is what seems to happen (see Figure 5.9). This figure plots blade pitch rate and pitch angles. In the left part of the figure, the blade pitch angles are above 2 degrees, and the instability is less apparent. As the blade pitch angle decreases toward 0 degrees, this instability seems to grow. The control results shown in Table 5.1 and Figures 5.6 to 5.8 reflected turbine operation well into Region 3 for large pitch angles.



Causes of this behavior are the change in control input gains we have already described. The decrease in control input gains with decreasing blade pitch angle causes an increase in the required control effort to apply a given amount of damping to the first drive train torsion mode, causing the blade pitch actuator to reach the pitch rate limits, causing instability.

Another cause could be inappropriate transitioning between Region 2 and Region 3 controllers. If there is a discontinuity in blade pitch or generator torque in transitioning between Regions 2 and 3, flexible modes such as the first drive train torsion mode could be excited.

## 5.6 Lessons Learned

By implementing and testing this DAC controller we learned that:

1. Including a blade pitch actuator model in the model used for control design was very important.
2. The CART commands pitch rate. We designed our controller to output pitch rate instead of pitch angle. We then passed this commanded pitch rate to the actuator model, in which the input is pitch rate and the output is pitch angle.
3. Using MATLAB to convert the state-space control model to discrete time for inclusion in the turbine control software was straightforward.
4. Use of rotor collective pitch to actively damp the first drive train torsional mode worked well when the turbine operated deep in Region 3. For turbine operation closer to Region 2 (in the Region 2½ transition or slightly above) use of this algorithm resulted in high pitch rates, because of the variation of control gains with turbine operating point. At low pitch angles when the turbine is operating close to Region 2½, the pitch input gains are small, requiring large control efforts to actively damp drive train torsion. Performance of this controller could be improved through the use of gain scheduling. Gain scheduling for state-space controllers is an area of further study and was not presented here.
5. It is important to design controls to provide a smooth transition between Regions 2 and 3. Any sudden changes in blade pitch or generator torque caused by discontinuities could contribute to the excitation of the first drive train torsion mode (and other flexible modes) shown in Figures 5.8 and 5.9.

With these results and lessons learned we decided to switch from the use of blade pitch as the control actuator for drive train damping to generator torque. We decided to implement and test a Region 3 and Region 2 generator torque controller that actively damps the first drive train torsional mode. We provide a smooth transition between these two controllers so that flexible modes will not be excited by discontinuities in blade pitch or generator torque between operating regions. We now describe this control design, implementation, and test.

## 6. Designing, Implementing, and Testing a Region 2 and Region 3 Drive Train Damper Using Generator Torque

### 6.0 Introduction

In this chapter, we show the design of generator torque control to enhance the damping in the first drive train torsion mode in both Regions 2 and 3, thereby reducing drive train torque loads. We discuss the control objectives and control structure. We then discuss the state-space control design approach for this controller. We then discuss implementation issues and preliminary test results.

### 6.1 Control Objectives and Structure

As we saw in the last chapter, using rotor collective pitch control to actively damp the first drive train torsion mode results in high pitch rates for turbine operation at the low wind speed end of Region 3. We would like to use generator torque as the control actuator for this active damping, since it is more effective than pitch control. We design generator torque control to perform active damping of this mode in Regions 2 and 3. The primary control objectives are to maximize power in Region 2 and maintain constant rotor speed in Region 3.

We will no longer use the Region 2 baseline generator torque control described in Chapter 3. We will design one generator torque controller for Region 2. This controller will track wind speed disturbances in an optimum way to maintain operation at optimum  $C_p$ . It will also perform drive train damping.

We also design a Region 3 generator torque controller that allows small perturbations from the nominal constant torque to actively damp drive train torsion in this region. In Region 3, we also operate a rotor collective pitch controller, which performs the overall speed regulation but does not actively damp drive train torsion. This pitch controller is designed separately from the generator torque controller; it is a separate control loop from the generator torque controller. We must also design a smooth transition between Region 2 and Region 3 controllers to avoid exciting flexible turbine modes, which increase dynamic loads. We now discuss the state-space control design approach.

### 6.2 State-Space Control Design

In (21), a multi-controller approach to state-space control of variable-speed wind turbines over the entire operating envelope was shown. Separate state-space controllers were designed for Region 2 and Region 3. In Region 2, the state-space controller was designed using Disturbance Tracking Control (DTC) theory (21). This control design approach allows us to use full-state feedback to achieve desired transient response and stability, while tracking optimum  $C_p$  all in one state-space controller. In Region 3, the controller was designed using DAC, an approach that also allows full-state feedback. The DTC and DAC controllers were then combined in a multi-controller system that smoothly switched

between the two controllers when the variable-speed turbine operation transitioned between Regions 2 and 3.

We apply this technique to the design of a generator torque controller to enhance drive train torsion damping in Regions 2 and 3. This controller tracks wind disturbances to maintain constant  $C_p$  and adds active damping to drive train torsion in Region 2. In Region 3, its only objective is to add damping to drive train torsion, as designed in Section 4.2. In the transition region, we perform a linear interpolation of generator torque control between Regions 2 and 3 to provide a smooth transition. A pitch controller is also designed using DAC to regulate turbine speed in Region 3 and mitigate the effects of wind speed disturbances. In Region 2, this pitch controller is saturated at run pitch.

We now describe design of the generator torque controller.

### **6.2.1 Generator Torque Control Design**

For the generator torque control design in either Region 2 or 3, the linear model that we use for control design consists of the states:

$$\begin{aligned} \Delta x_1 &= \text{Drivetrain rotational-flexibility DOF} \\ \Delta x_2 &= \text{Variable speed generator DOF velocity (generator speed)} \\ \Delta x_3 &= \text{Drivetrain rotational-flexibility DOF velocity.} \end{aligned} \tag{6.1}$$

This is just like the control model of Section 4.2. We append this model with a uniform wind disturbance (in the control synthesis routine) to track wind speed disturbances in Region 2 in such a way as to maintain optimum  $C_p$ . In Region 3 we use a linear model with these same states as in (6.1) to design the generator torque controller. Even though for Region 3, we include a wind disturbance state, we do not want the controller to try to cancel wind speed disturbances. This role will be assigned to the pitch controller to be described later. To eliminate this role from the generator torque controller, we simply set the disturbance gain to zero. We wanted to maintain the same number of states (thus including the wind disturbance state in the Region 3 torque controller) in the Region 2 and Region 3 torque controllers to ensure we maintain a smooth transition between controllers.

We designed the Region 2 generator torque controller from a state-space model generated by FAST at the turbine operating point:

$$\begin{aligned} \text{Wind speed} &= 8 \text{ m/s} \\ \text{Rotor speed} &= 27.1 \text{ rpm} \\ \text{Pitch angle} &= -1 \text{ degree (normal Region 2 run pitch).} \end{aligned}$$

We refer to this operating point as the Region 2 control design point. It was chosen midway between the low and high wind speed ends of Region 2.

The goal of state-space control design is to use state feedback to place the poles of the plant to obtain desired stability and transient response. In open-loop, the eigenvalues of the A matrix at this operating point are  $-0.0004 \pm 3.57i$  (Hertz [Hz]),  $-0.0044$ . The pole

pair corresponds to the first drive train torsion mode, with a natural frequency of 3.57 Hz. We choose the gains in the feedback law so that the eigenvalues are  $-1 \pm 3.57i$  (Hz),  $-0.1$ . This results in greatly enhanced damping in the first drive train torsion mode. The gain corresponding to the uniform wind disturbance is chosen to track wind speed disturbances using the theory of DTC (21).

For the Region 3 generator torque controller, a linear model with the same plant states as in (6.1) is generated by FAST at the turbine operating point:

Wind speed = 18 m/s  
 Rotor speed = 41.7 rpm  
 Pitch angle = 11 degrees.

We will refer to this point as the Region 3 control design point.

The goal is again to use state full feedback to place the poles of the plant to obtain desired stability and transient response. In open-loop poles, the eigenvalues of the A matrix had the values  $-0.0016 \pm 3.5712i$  (Hz),  $-0.16$ . We choose the gains  $G$  in the feedback law so that the closed-loop poles will have values  $-0.6 \pm 3.57i$  (Hz),  $-0.15$ .

In Region 3 control, the generator torque is usually constant. Blade pitch control is then used to maintain a constant turbine rotational speed, thereby fixing power at rated power. Here we allow only small perturbations in generator torque to enhance damping in the first drive train torsion mode. The gain corresponding to the uniform wind disturbance ( $G_d$ ) is set to zero for the Region 3 torque controller to prevent tracking or canceling wind speed disturbances, which would cause large variations in generator torque.

State estimation is used to estimate unmeasured plant states by measuring only generator speed in both the Region 2 and Region 3 torque controllers.

### **6.2.2 Region 2 to 3 Generator Torque Control Transition**

In the Region 2 to 3 transition, we switch between the Region 2 and Region 3 generator torque controllers based on the equation:

$$Q_{gen} = (Q_2 - Q_1)(\Omega - \Omega_1)/(\Omega_2 - \Omega_1) + Q_1 \quad (6.2)$$

where  $Q_{gen}$  is the commanded generator torque,  $Q_1$  is the Region 2 torque at the Region 2 endpoint  $\Omega_1$ ,  $Q_2$  is the Region 3 torque at  $\Omega_2$ ,  $\Omega$  is rotor speed,  $\Omega_1$  is the rotor speed at the start of Region 2 to 3 transition (39.2 rpm), and  $\Omega_2$  the rotor speed at the start of Region 3 (41.3 rpm). This is similar to the approach used in (21) to switch between Region 2 and 3 controllers, as well as the expression used in Section 3.3 (Equation [3.3]) to interpolate between Region 2 and Region 3 torque controllers.

### 6.2.3 Region 3 Pitch Controller

Control of turbine speed in Region 3 is performed by the rotor collective pitch controller. Here we use DAC to design the controller to mitigate wind speed disturbances. The rotor collective pitch controller is designed from a state-space model with the following states:

$\Delta x_1$  = perturbed rotor symmetric flap displacement

$\Delta x_2$  = perturbed rotor symmetric flap velocity

$\Delta x_3$  = perturbed generator speed

$\Delta x_4$  = perturbed actuator pitch angle

$\Delta x_5$  = perturbed actuator pitch rate

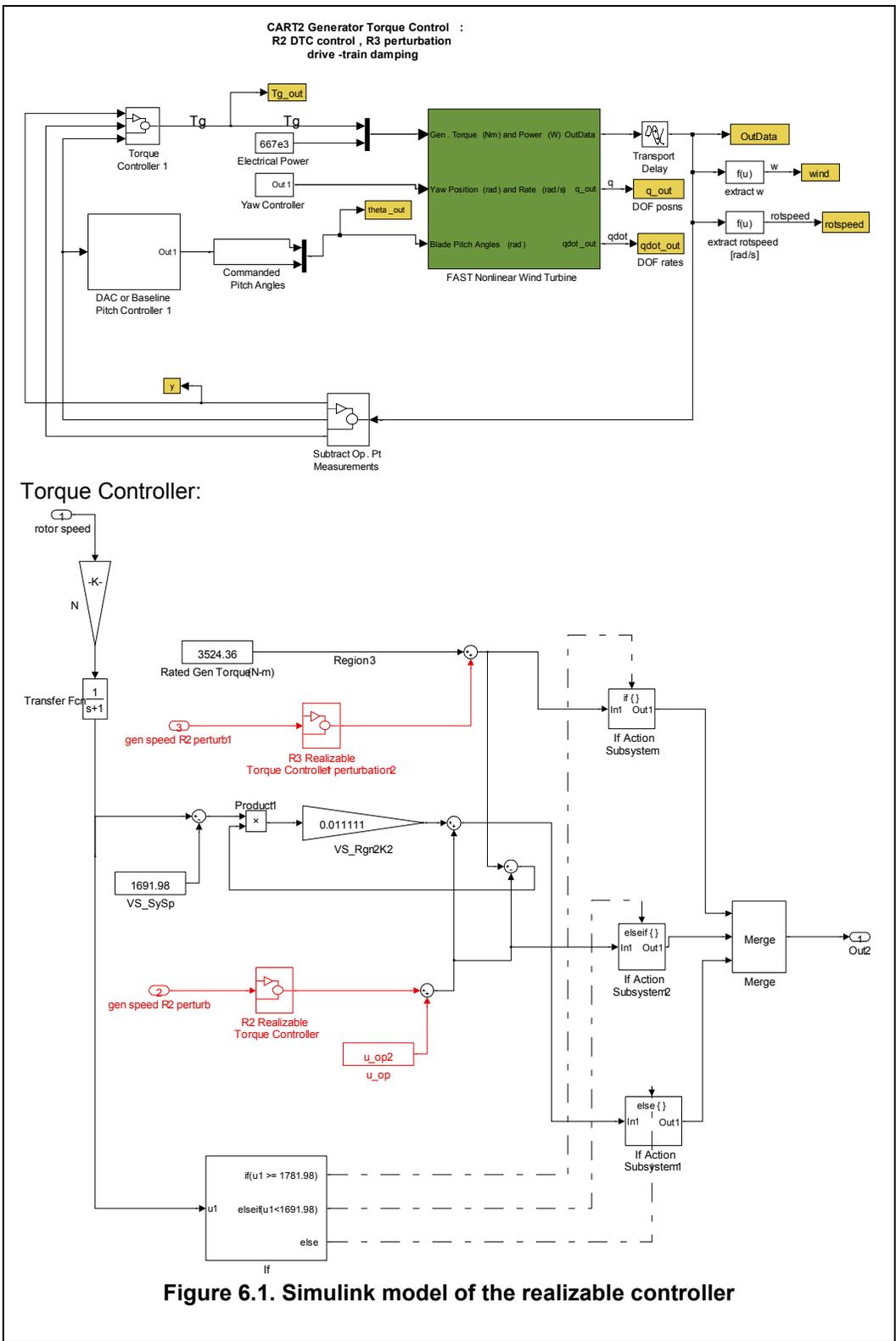
We again find it necessary to include actuator dynamics in the model for control design and turbine implementation as described in Chapter 5, and as reflected in selection of states  $x_4$  and  $x_5$ . In addition, a state to describe the uniform wind disturbance is appended to the model in the control synthesis routine.

For design of this controller, this state-space model is generated by FAST at a turbine operating point the same as the Region 3 control design point described in section 6.2.1.

We choose the gains  $G$  in the feedback law so the closed-loop poles will have the values  $-4.6 \pm 13.5i$  (Hz.),  $-2$ ,  $-10$ , and  $-59$ . The first pair of poles corresponds to the perturbed rotor first symmetric flap mode. The next pole corresponds to the perturbed rotor rotational speed state, and the next two states correspond to the perturbed actuator blade pitch angle and perturbed pitch rate. In this operating region, the control objective for the pitch controller is to regulate speed in the presence of persistent wind disturbances. The gain corresponding to the wind disturbance  $G_d$  is chosen to mitigate wind disturbances uniformly over the rotor disk using DAC (21). We do not include the first drive train torsion mode states in the linear model for design of the Region 3 pitch controller. We feel that this would minimize any tendency for the pitch controller and torque controller in Region 3 to interact with each other at frequencies at or close to the first drive train torsion mode.

## 6.3 Simulation Tests

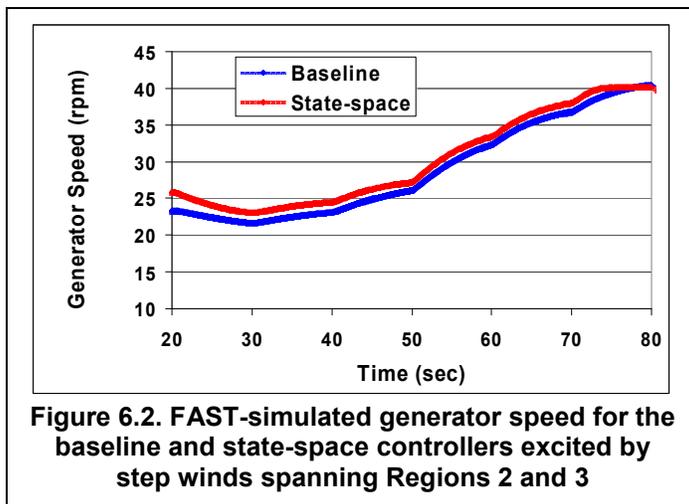
We run test simulations in FAST before implementing these controllers on the real machine. Figure 6.1 shows a Simulink model of the closed-loop system with pitch and generator torque controllers.



During one test, we want to compare the rotor speed predicted by FAST when using the state-space controllers to the baseline Region 2 and Region 3 controllers. These baseline Region 2 and 3 controllers are very similar to the baseline controllers illustrated in

Chapter 3. The baseline pitch controller is designed only to regulate speed in Region 3; it performs no mode damping. The Region 2 controller is almost identical to the torque controller designed in Chapter 3.

Figure 6.2 shows this comparison. In this simulation, we input step winds, beginning with a wind speed of 6 m/s and increasing to 12 m/s in steps during 60 s of simulation. Over the entire wind speed range, the state-space controller results in slightly higher rotor speeds than the baseline controller. The larger discrepancy at the low end of Region 2 may be a result of designing only one controller in Region 2, at one operating point (wind speed of 8 m/s). A solution may be to design several controllers at different Region 2 operating points and use gain scheduling to transition between controllers. In addition, it is shown in (29) that certain state-space models may result in imperfect optimum TSR tracking using DTC theory. We will investigate these issues in future work.

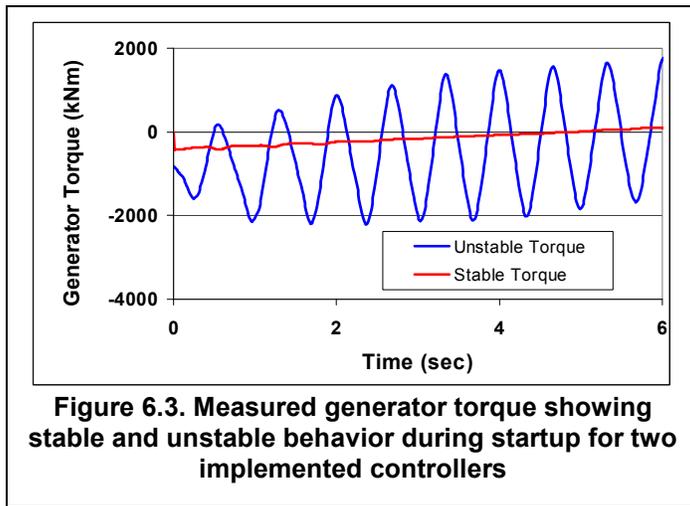


## 6.4 Implementation Issues

There are various issues in implementing and testing wind turbine controls. Sensor limitations, data sampling rates, and actuator dynamics are important, as reported in Chapter 5. The previously described controller is implemented into the CART control software. Care is taken to ensure that all programming bugs are removed. Before this algorithm was field tested, the control code was tested using the simple simulator built into the control software. These tests are useful for further catching C-code implementation bugs, highlighting region transition problems, and as an independent check of speed regulation performance.

There are also situations in which testing a new controller on the real machine results in unstable behavior, even though the simulations do not reveal this behavior. When first testing a new controller, turbine behavior must be closely monitored to ensure stable operation. The first Region 2 generator torque controller we implemented and tested in the CART resulted in unstable generator torque upon startup, as shown in Figure 6.3. None of our simulations predicted this behavior. This controller was designed from a linear model generated at a turbine operating point closer to Region 3 than the final

design. When the controller was redesigned at the operating point described in the control design section above, we obtained stable generator torque upon startup.



Another issue was correctly implementing the Region 2 to 3 transition. Early implementation of these generator torque controls resulted in a rough transition. Instead of adding damping to the first drive train torsion mode, we were actually destabilizing this mode. Figure 6.4a shows turbine response while being controlled by the first implementation of the transition Equation (6.2). In this first implementation, shown in Figure 6.4a, we used unfiltered rotor rotational speed in (6.2). This signal (LSS rpm) contains high-frequency noise, as seen in the figure. Figure 6.4a also shows commanded generator torque (torque demand) and LSS torque. Between 160 and 163 seconds, the turbine is entirely in transition, because the rotor speed is above the entrance to transition (39.2 rpm). The resulting demanded generator torque (torque demand) contains much of the same high-frequency content as the LSS rpm signal. This noise acts to excite the drive train torsion instead of adding damping, as seen in the LSS torque signal in Figure 6.3a. We then filtered rotor speed in Equation (6.2) with a first-order filter with a time constant of 1 s. The resulting measured demanded generator torque, LSS torque, and LSS rpm (unfiltered LSS rpm values shown in this plot) are shown in Figure 6.4b. Now, except for the point at which the turbine enters transition, the commanded generator torque is much smoother than in Figure 6.4a. The LSS torque signal shows much less response at the first drive train torsion mode.

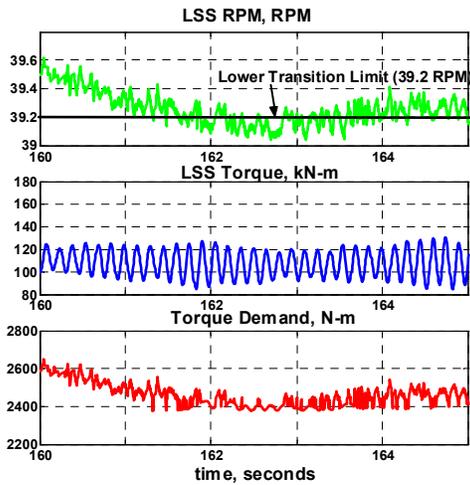


Figure 6.4a

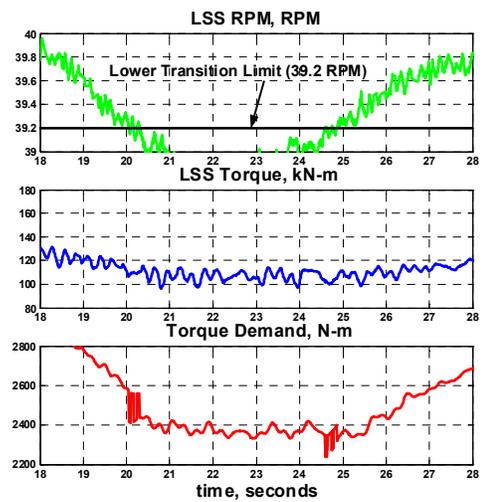


Figure 6.4b

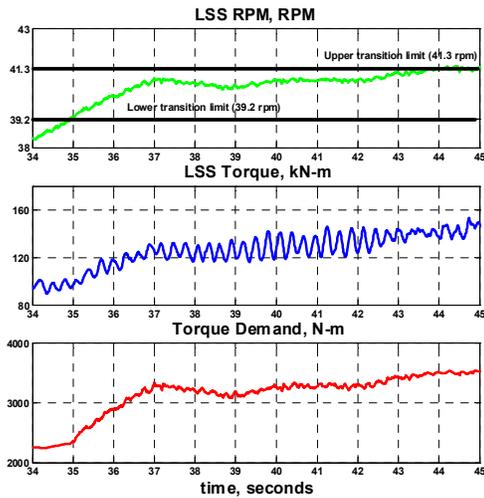


Figure 6.4c

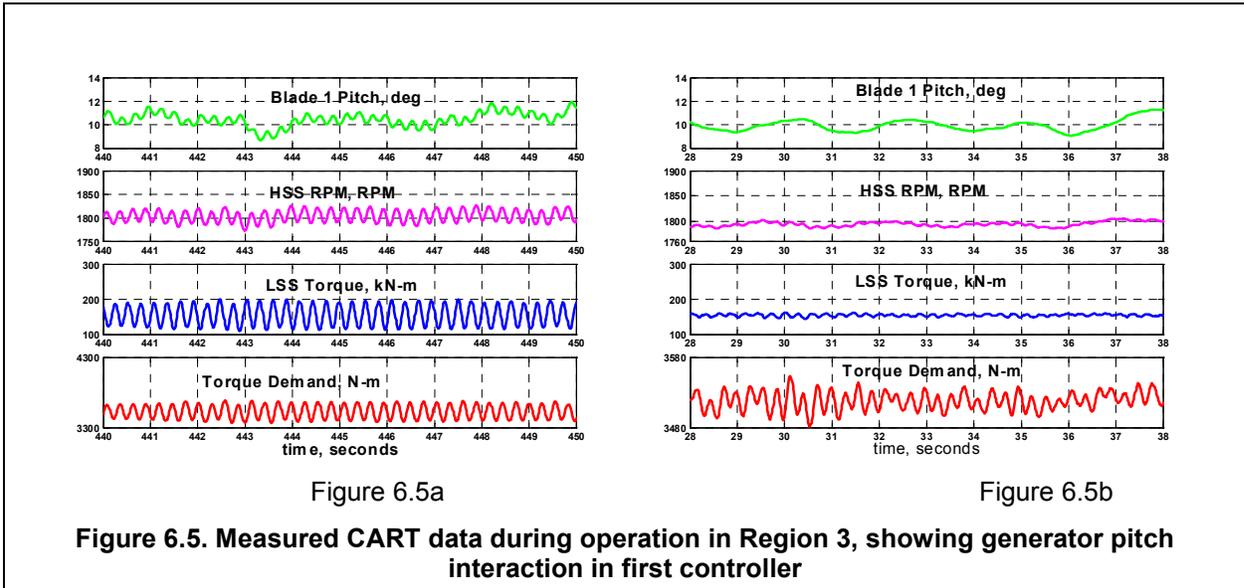
**Figure 6.4. Measured CART data during transition from Region 2 to Region 3**

A further problem occurred at the limit points of the Region 2 to 3 transition (lower limit 39.2 rpm, upper limit 41.3 rpm). Figure 6.4b shows discontinuities in demanded generator torque when the lower limit of 39.2 rpm is reached. These discontinuities are caused by a mismatch of rotor speed used in Equation (6.2) and rotor speed used in the decision to enter transition. At any time, the unfiltered and filtered values of rotor speed will be unequal. From Equation (6.2) we see that for a rotor speed of 39.2 RPM, the demanded generator torque should equal  $Q_1$  (Region 2 torque at 39.2 rpm). If there is a mismatch in the rotor speed used in Equation (6.2) and the rotor speed used in the decision to enter the transition test, the demanded generator torque will not equal  $Q_1$ , and there will be a discontinuity in demanded generator torque upon entering or exiting the transition region. Figure 6.4c shows the result of using filtered rotor speed in both

Equation (6.2) and the decision test. In this figure, the rotor speed passes through both the lower and upper limit points (39.2 and 41.3 rpm). Now the commanded generator torque is smooth without discontinuities.

Another problem concerned interaction between the Region 3 torque controller and the Region 3 pitch controller. Both the generator torque and blade pitch controllers were designed assuming generator speed (HSS rpm) as the control input. As shown in Figure 6.5a, the HSS rpm signal contains high cyclic amplitude at the first drive train torsion frequency. Figure 6.5a also shows that the LSS torque and the demanded generator torque all contain response at this frequency. Because HSS rotational speed is the input to the pitch controller as well, the pitch controller also responds at this natural frequency, as seen in the plot of blade 1 pitch in Figure 6.5a. The pitch controller and torque controller interact in an undesirable way.

We then changed the control input to the blade pitch controller to LSS rpm instead of HSS rpm. This signal contains less response at the first drive train torsion frequency because the large rotor inertia filters out response at this frequency (3.5 Hz). Now, the blade pitch controller no longer responds at the first drive train torsion frequency (plot of blade 1 pitch, Figure 6.5b), and the generator torque controller no longer interacts with the pitch controller (Figure 6.5b). Even though the pitch controller was not designed from a state-space model containing a description of the first drive train torsion mode, it interacts with the torque controller when high-speed rpm is used as the control input. Inputting rotorspeed (LSS rpm) instead of generator speed (HSS rpm) solves this problem. Another solution would be to use HSS RPM as input to the pitch controller but to apply a low-pass filter in order to filter out the effects of the first drive train torsion mode.



We now make some preliminary comparisons of the state-space controller to the baseline controller in Regions 2 and 3.

## 6.5 Field Test Results and Comparisons

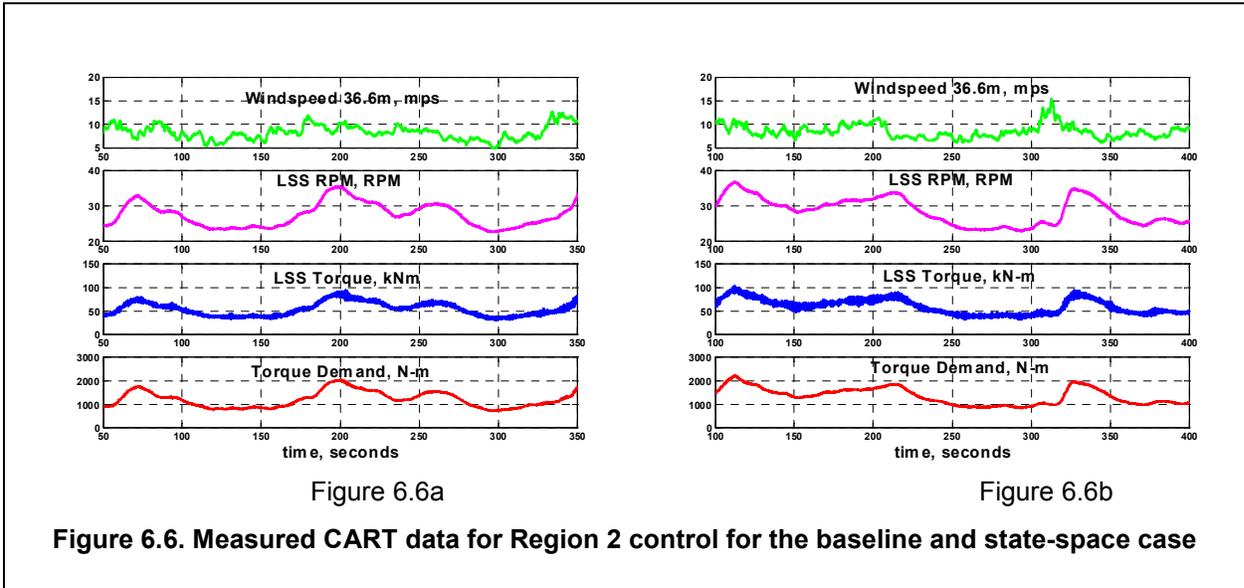
We collected data on the CART while testing the Region 2 state-space controller. We then compared the test results to a case from the baseline control in Region 2. Because these controllers were tested at different times, a direct results comparison is difficult. Our objective was to show trends such as load mitigation. We attempted to examine both datasets and extract smaller sections of data in which turbine operating parameters (such as wind speed, wind direction, and yaw error) were similar. The results of our comparison are preliminary, and we need many more hours of operation for realistic statistical comparisons between the controllers.

Table 6.1 shows the statistics of two datasets used for comparison. Each dataset consisted of 300 s of data. Listed in this table are the mean and standard deviation (std) of the windspeed (at a height of 36.6 m), rotor speed (LSS rpm), generator torque (torque demand), and LSS torque.

**Table 6.1: Preliminary Comparison of Baseline and FAST State-Space Controllers for Region 2 Operation**

<b>Statistics and Performance Measure</b>	<b>Baseline Control (05031750.DAT)</b>	<b>State-Space Control (07191751.DAT)</b>
Wind speed (m/s)	mean 8.45 std 1.44	mean 8.22 std 1.53
Rotor speed (rpm)	mean 28.53 std 3.86	mean 27.3 std 3.35
Demanded generator torque (kN-m)	mean 1340 std 361	mean 1190 std 341
Low-speed shaft torque (kN-m)	mean 59.4 std 15.9 fatigue DEL 7.63	mean 53.3 std 14.1 fatigue DEL 6.83

Figure 6.6 shows plots of wind speed, LSS rotational speed, LSS torque, and demanded generator torque for the two datasets. Figure 6.6a corresponds to the baseline controller and Figure 6.6b the state-space controller. Although the wind speeds do not match, the deviations in wind speed for the two cases are similar; somewhat greater deviations occur during operation with the state-space controller. We also calculated the DEL for the LSS torque for the two datasets (28). The state-space controller resulted in slightly lower DEL than the baseline controller.



We also calculated the power spectral density (PSD) of the LSS and demanded generator torque for the two cases, as shown in Figure 6.7. The peak in the PSD for the LSS torque (Figure 6.7a) corresponds to the first drive train torsion natural frequency. This peak is lower for the state-space control case because we have used full state feedback to place the poles corresponding to this mode to have higher damping, resulting in reduced loads at this frequency. The peak for the state-space control case is displaced slightly to the right of the baseline case because we have placed the pole to have a natural frequency at 3.57 Hz. The natural frequency of the first drive train torsion mode in open-loop is probably about 3.3 Hz, as seen by the peak response for the baseline case. This represents a slight discrepancy between the FAST-predicted natural frequency (3.57 Hz) and the open-loop natural frequency of this mode in the real machine, because of modeling uncertainty.

Figure 6.7b shows the PSD of the demanded generator torque. For the baseline control case, the peak at 3.57 Hz is absent because the controller is not adding damping to this mode. For the state-space case, the large peak at 3.57 Hz indicates the action of the generator controller to add damping to this mode as desired. Table 6.1 indicates (std of demanded generator torque) that the generator torque activity is lower for the state-space case compared to the baseline case. Further data collection and comparisons are needed before we can assess the effects of these additional control objectives on the actuator duty cycle of the generator.

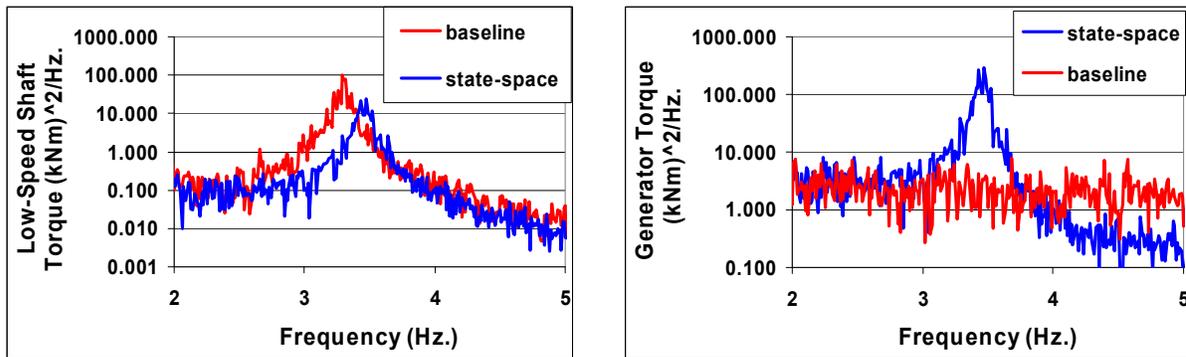


Figure 6.7a

Figure 6.7b

**Figure 6.7. Power spectral density of LSS and generator torque for the Region 2 baseline and state-space cases**

Next we compared the Region 3 state-space controller to the Region 3 baseline PI controller. Table 6.2 compares statistics of 300 s of CART data during operation with the PI controller and the state-space controller. Figure 6.8 shows plots of different data channels for these two cases. The mean and std of wind speed are slightly lower for the state-space controller. The mean rotor speed is also slightly lower, although the std is slightly higher than for the baseline case. A lack of turbine data for the turbine operating entirely in Region 3 meant that there was some operation in the transition region. The std of demanded generator torque is higher for the state-space controller than for the baseline case. This may be due to operation in the transition region. For the baseline controller operating entirely in Region 3, the std of demanded generator torque should be zero. A direct assessment of actuator duty for the state-space controller is difficult. We also calculated the DEL for the LSS torque, with a large reduction in load for the state-space controller compared to the baseline controller.

**Table 6.2. Comparison of Baseline and FAST State-Space Controller for Region 3 Operation**

Statistics and Performance Measure	(05212101.DAT)	State-Space Control (09092110.DAT)
Wind speed (m/s)	mean 15.85 std 2.31	mean 15.74 std 2.11
Rotor speed (rpm)	mean 41.67 std 0.30	mean 41.26 std 0.49
Demanded generator torque (kN-m)	mean 3501 std 112.9	mean 3409 std 160.2
Low-speed shaft torque (kN-m)	mean 154.10 std 8.08 fatigue DEL 18.27	mean 149.84 std 7.92 fatigue DEL 8.92

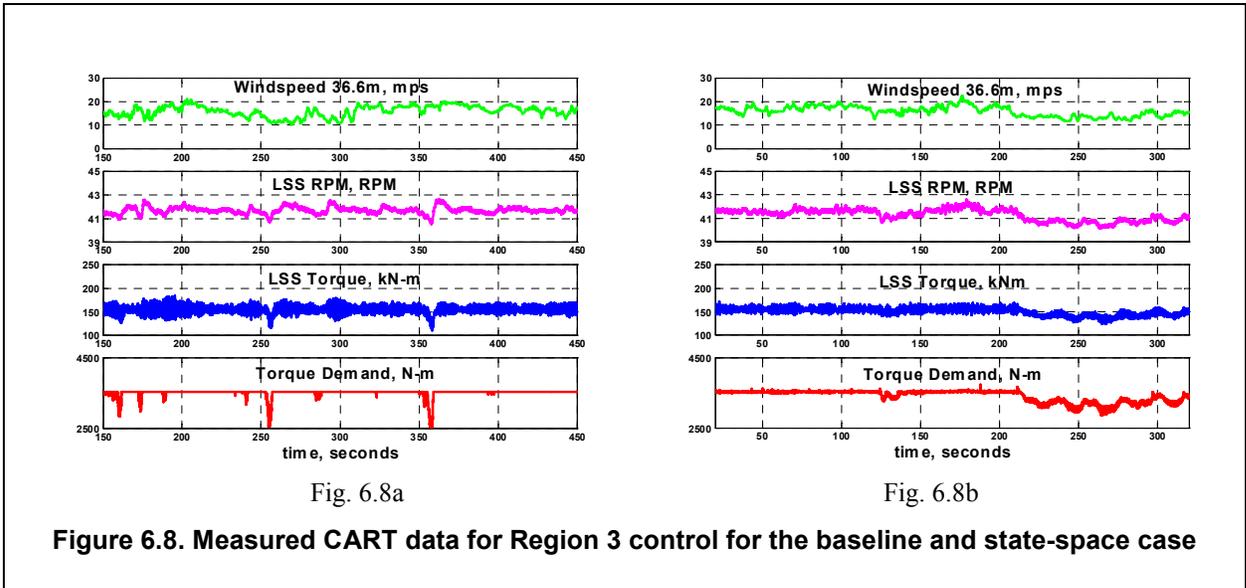
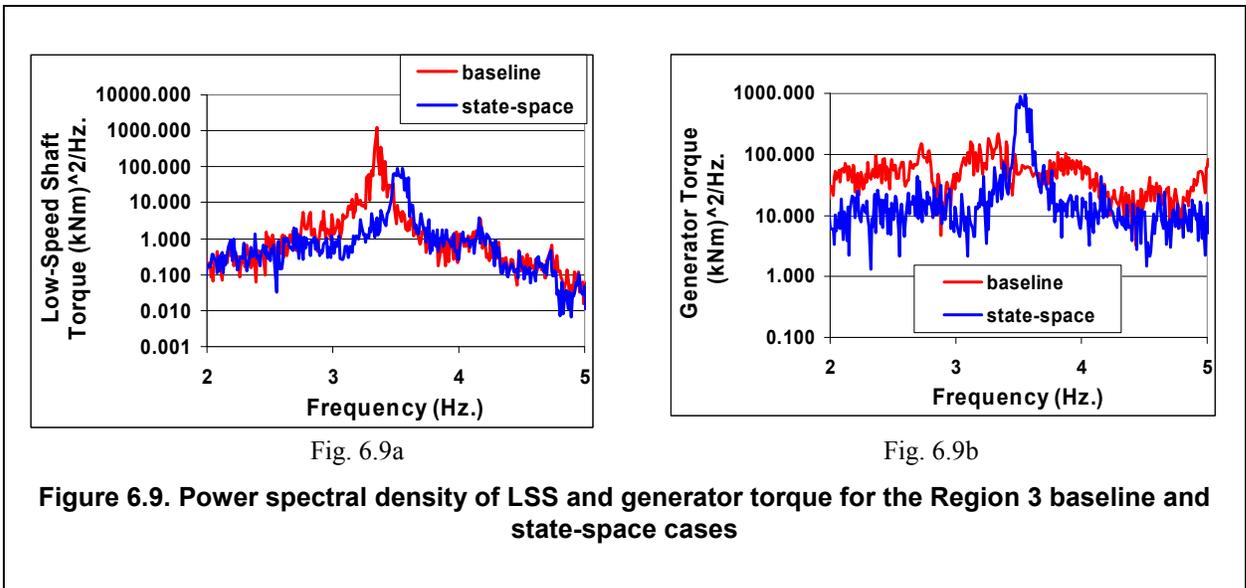


Figure 6.9 shows the PSDs of LSS and generator torque for the two control cases. Again, we see in Figure 6.9a a significant reduction in the peak at the first drive train torsion mode for the state-space case because of the damping enhancement provided by the generator controller. The PSD of generator torque shows a peak at 3.57 Hz for the state-space case, indicating that the generator is adding damping to the first drive train torsion mode.



Although these results are preliminary, Tables 6.1 and 6.2 seem to show the potential to reduce LSS torque loads with the state-space controllers. This is due to the enhanced damping of the first drive train torsion mode with these controllers. The load reduction seems to be greater in Region 3 than in Region 2. More data must be collected with the state-space controllers and comparisons made to the baseline controller before firm

conclusions can be reached. In this preliminary study, the trends indicate potential for load alleviation using state-space control designs.

## 6.6 Lessons Learned

By designing, implementing, and testing these Region 2 and Region 3 generator torque controllers for drive train damping we learned the following:

1. Use of generator torque as a control actuator for active drive train damping results in better performance than with blade pitch. The control input gains for generator torque are constant with turbine operating speed. We did not observe the problems in the low end of Region 3 that we observed when attempting to use blade pitch to perform this function (see Chapter 5).
2. Providing a smooth transition between the Region 2 torque controller and the Region 3 torque controller was crucial in preventing excessive drive train torque loads. The first attempt to tie these two controllers together resulted in discontinuities, which excited rather than damped this flexible mode. Correct use of Equation (6.2) was crucial in correctly tying these two controllers together, that is, both the controller and the transition test must use the same filtered signal.
3. There was some evidence that the pitch control system interacted with the generator torque control system in Region 3, because these two controllers are designed as two separate controllers, resulting in two separate control loops. This interaction was minimized by switching to rotor speed instead of generator speed as the control input for the pitch control system. This filters out action at the first drive train torsion frequency.
4. The disturbance tracking controller for Region 2 designed and implemented here may not track optimum  $C_p$  as well as the baseline Region 2 generator torque controller. This is because only one Region 2 controller was designed and used over the entire Region 2 operating envelope. This could be improved by designing several DTC Region 2 controllers and using gain scheduling for control over different parts of Region 2.

Having now described some state-space control implementations and lessons learned from the CART, we now state conclusions and future work.

## 7. Conclusions and Future Work

The purpose of this report is to provide wind turbine engineers information about designing, implementing, and testing advanced control systems for wind turbines. We wanted to illustrate the use of available control design tools as well as the steps involved with designing and implementing advanced controllers. We have illustrated design of controls for Regions 2 and 3 for a variable-speed turbine.

In Chapter 1 we described the challenges involved with designing controls for these complex nonlinear wind turbine structures. We described standard baseline wind turbine controllers commonly used in industry.

In Chapter 2 we described the control development process and the tools that were to be used in subsequent sections of this report.

In Chapter 3 we illustrated the design of baseline generator torque control as well as Region 3 baseline pitch control. The Region 2 torque control was designed using very simple equations for the generator torque as a function of generator speed. We saw that the Region 3 torque was constant and we described methods of interpolating to provide smooth transition between these regions.

We described the design of a Region 3 baseline pitch controller using simple PID control. We also saw how to implement anti-windup and gain scheduling to improve control performance. We also added tower damping control as an additional control loop onto this baseline pitch control.

In Chapter 4 we illustrated three state-space control design examples: (1) rotor collective pitch in Region 3; (2) generator torque to actively damp drive train torsion in Region 3; and (3) independent blade pitch control in Region 3 to mitigate the effects of asymmetric wind disturbances over the rotor disk. We saw how to produce a linear model for use in control design. We used MATLAB to illustrate use of the control design synthesis tools. We then illustrated how to simulate these state-space controllers with a Simulink model of the closed-loop system (simulation with FORTRAN subroutines instead of Simulink was illustrated in Appendix D).

In Chapter 5 we demonstrated the implementation and testing of the rotor collective pitch controller designed in Chapter 4. We implemented and tested this controller on the CART. We described implementation issues, preliminary test results, and lessons learned.

In Chapter 6 we described design, implementation, and testing of Region 2 and Region 3 generator torque controllers for active drive train damping. We described how to tie these two controllers together to provide smooth transition between Region 2 and Region 3 control. We described several implementation issues, presented some preliminary comparisons to baseline control results, and described lessons learned.

This report is Part I in a two-part series of reports detailing advanced control design, implementation, and test results. Part I (this report) has highlighted the control development process, from forming control objectives, designing the controller, testing the controller through analytical simulation, to field implementation and initial field testing. In this report, use of the control design and simulation tools has been demonstrated through design of simple industry standard baseline controls as well as more advanced state-space controls. We have also demonstrated some of the issues we have encountered in implementing and testing state-space controls in the CART. Hopefully these examples and descriptions have given wind turbine engineers needed information in the design and implementation of Region 2 and Region 3 advanced controls.

The lessons we have learned include:

1. Including a blade pitch actuator model in the model used for control design was very important when implementing a Region 3 DAC collective pitch controller in the CART.
2. Use of rotor collective pitch to actively damp the first drive train torsional mode worked well when the turbine operated deep in Region 3. For turbine operation closer to Region 2 (in the Region 2½ transition or slightly above) use of this algorithm resulted in high pitch rates, because the control gains varied with turbine operating point. At low pitch angles when the turbine is operating close to Region 2½, the pitch input gains are small, requiring large control efforts to actively damp drive train torsion.
3. It is important to design controls to provide a smooth transition between Regions 2 and 3. Any sudden changes in blade pitch or generator torque caused by discontinuities could contribute to the excitation of the first drive train torsion mode (and other flexible modes). The first attempt to tie the Region 2 and Region 3 generator torque controllers together resulted in discontinuities, which excited rather than damped this flexible mode.
4. Use of generator torque as a control actuator for active drive train damping results in better performance than with blade pitch. The control input gains for generator torque are constant with turbine operating speed. We did not observe the problems in the low end of Region 3 when using generator torque that we observed when using blade pitch to actively damp the drive train (as described in item 2) and in Chapter 5).
5. There was some evidence that the pitch control system interacted with the generator torque control system in Region 3, because these two controllers are designed as two separate controllers, resulting in two separate control loops. This interaction was minimized by switching to rotor speed instead of generator speed as the control input for the pitch control system. This filters out action at the first drive train torsion frequency.

This is not a complete list of lessons we have learned in testing advanced controls on the CART. For a further description of lessons learned, see (30).

Item 6 in the list of lessons learned highlights a problem with designing separate controllers in separate control loops. The possibility of these individual controllers interacting in an undesirable way increases unless they are very carefully designed. Therefore, the main focus of our advanced controls research will be to design advanced MIMO controllers that perform these multiple control objectives with a minimum number of control loops. An example MIMO state-space controller for tower load alleviation has already been designed and preliminary test results reported in (13).

Part II of this report (to be completed later) will give a detailed comparison of results from advanced load alleviating state-space controllers to test results from baseline controllers which do not perform load alleviation. The purpose of Part II is to demonstrate through rigorous testing the load mitigating potential of the advanced state-space controllers. Before this report can be published, extensive test data from these advanced state-space controllers must be collected.

Future work will include testing independent pitch control for load alleviation as well as tower damping controls. We plan to design one MIMO controller which performs these functions in one control loop. We also plan to begin developing controls and testing them on a 3-bladed version of the CART.

We also plan to begin advanced controls development for offshore wind turbine systems.

## References

1. Zhao, W. and Stol, K., "Individual Blade Pitch for Active Yaw Control of a Horizontal-Axis Wind Turbine," presented at the 45<sup>th</sup> AIAA Aerospace Sciences Meeting and Exhibit, 8-11 January 2007, Reno, NV.
2. Bossanyi, E. A., "Developments in Closed Loop Controller Design for Wind Turbines," *19<sup>th</sup> ASME Wind Energy Conference*, Reno, NV, 2000, pp. 64-74.
3. Fingersh, L. and Johnson, K., "Baseline Results and Future Plans for the NREL Controls Advanced Research Turbine," *23<sup>rd</sup> ASME Wind Energy Conference*, Reno, NV, 2004, pp. 87-93.
4. Johnson, K. E., Fingersh, L., Balas, M., and Pao, L. Y., "Methods for Increasing Region 2 Power Capture on a Variable-speed HAWT," *ASME J. Solar Energy Engineering*, Vol. 126, No. 4, November 2004, pp. 1092–1100.
5. Burton, T., Sharpe, D., Jenkins, N., and Bossanyi, E., 2001, *Wind Energy Handbook*, John Wiley & Sons, LTD, New York, pp. 488–489.
6. Bossanyi, E. A., "Individual Blade Pitch Control for Load Reduction," *Wind Energy*, 6: 119-128, 2003.
7. Bossanyi, E. A., "Developments in Individual Blade Pitch Control," EWEA conference "The Science of Making Torque from Wind," DUWIND, Delft University of Technology, The Netherlands, April 19-21, 2004.
8. Geyley, M. and Caselitz, P., "Individual Blade Pitch Control Design for Load Reduction on Wind Turbines," Presented at the 2007 European Wind Energy Conference and Exhibition, Milan, Italy, 7-10 May 2007.
9. Stol, K., Zhao, W., and Wright, A., "Individual Blade Pitch Control for the Controls Advanced Research Turbine (CART)," *ASME J. Solar Energy Engineering*, Vol. 128, No. 4, November 2006, pp. 498–505.
10. Stol, K., and Fingersh, L., "Wind Turbine Field Testing of State-Space Control Designs," Golden, CO: National Renewable Energy Laboratory, NREL/SR-500-35061, 2004.
11. Wright, A., Stol, K., and Fingersh, L., "Progress in Implementing and Testing State-Space Controls for the Controls Advanced Research Turbine," *24<sup>th</sup> ASME Wind Energy Conference*, Reno, NV, 2005, pp. 88-100.
12. Wright, A., Fingersh, L., and Balas, M., "Testing State-Space Controls for the Controls Advanced Research Turbine," *ASME J. Solar Energy Engineering*, Vol. 128, No. 4, November 2006, pp. 506–515.
13. Wright, A. D., Fingersh, L. J., and Stol, K. A., "Design and Testing Controls to Mitigate Tower Dynamic Loads in the Controls Advanced Research Turbine," Golden, CO: National Renewable Energy Laboratory, NREL/CP-500-40932, 2007 (<http://www.nrel.gov/docs/fy07osti/40932.pdf>).

14. Hansen, M. H., Hansen, A., Larsen, T. J., Oye, S., Sorensen, P., Fuglsang, P., "Control Design for a Pitch-regulated Variable-speed Wind Turbine," Report # Riso-R-1500(EN) Riso National Laboratory, Roskilde, Denmark, 2005.
15. Elliot, A. S. and Wright, A. D. "ADAMS/WT: An Industry-Specific Interactive Modeling Interface for Wind Turbine Analysis," in *Wind Energy 1994*, edited by W. D. Musial, S. M. Hock, and D. E. Berg, SED-Vol. 14, New York: American Society of Mechanical Engineers; pp. 111-122.
16. Jonkman, J. M. and Buhl, M. L., "FAST User's Guide," Golden, CO: National Renewable Energy Laboratory, NREL/EL-500-38230, 2005.
17. Wright, A. D., "Modern Control Design for Flexible Wind Turbines," NREL Report No. TP-500-35816, Golden, CO: National Renewable Energy Laboratory, 2004.
18. Stol, K. A. and Bir, G. S. "SymDyn User's Guide," NREL/EL-500-33845. Golden, CO: National Renewable Energy Laboratory, 2003.
19. Kwakernaak, H. and Sivan, R., 1972, *Linear Optimal Control Systems*, Wiley Interscience, New York, NY.
20. Stol, K. and Balas, M., "Periodic Disturbance Accommodating Control for Blade Load Mitigation in Wind Turbines," *ASME J. Solar Energy Engineering*, Vol. 125, No. 4, November 2003, pp. 379-385.
21. Balas, M.J., Lee, Y.J., and Kendall, L., "Disturbance Tracking Control Theory with Application to Horizontal Axis Wind Turbines," *17th ASME Wind Energy Conference*, Reno, NV, 12-15 January 1998.
22. Kane, T.R. and Levinson, D.A., 1985, *Dynamics: Theory and Applications*, McGraw-Hill Book Company, New York, 1985.
23. Laino, D.J. and Hansen, A.C., "User's Guide to the Computer Software Routines AeroDyn Interface for ADAMS®." Salt Lake City, UT: Windward Engineering, LC, 2001.
24. Buhl, M. L. Jr., "WT\_Perf User's Guide," NREL codes web-page: [http://nwtc/designcodes/simulators/wtperf/WT\\_Perf.pdf](http://nwtc/designcodes/simulators/wtperf/WT_Perf.pdf).
25. Ogata, K., 1990, *Modern Control Engineering*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
26. Jonkman, J. M., "Dynamics Modeling and Loads Analysis of an Offshore Floating Wind Turbine," Ph.D. Dissertation, Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 2007. (also NREL Technical Report NREL/TP-500-41958).
27. Grace, A., Laub, A. J., Little, J. N., and Thompson, C. M., "Control System TOOLBOX for Use with MATLAB," Natick, MA: The MATHWORKS, Inc., 1992.
28. Sutherland, H. J., "On the Fatigue Analysis of Wind Turbines," Albuquerque, NM: Sandia National Laboratories, SAND99-0089, 1999.

29. Stol, K. A., "Disturbance Tracking and Blade Load Control of Wind Turbines in Variable-Speed Operation," *22nd ASME Wind Energy Conference*, Reno, NV, 6-9 January 2003.
30. Johnson, K., Fingersh, L. J., and Wright, A., "Controls Advanced Research Turbine: Lessons Learned During Advanced Controls Testing," NREL Report No. TP-500-38130, Golden, CO: National Renewable Energy Laboratory, 2005.

# Appendix A: Input Files for Simulation and Linearization

## A.1 FAST Simulation Files for the CART (Chapters 3 and 4)

### FAST Input File Example (cartnewsim.fad)

```
----- FAST INPUT FILE for Turbine Simulation-----
CART Example.
Compatible with FAST v6.01.
----- SIMULATION CONTROL -----
False Echo - Echo input data to "echo.out" (switch)
1 ADAMSPrep - ADAMS preprocessor mode {1: Run FAST, 2: use FAST as a preprocessor to create an ADAMS model, 3: do both} (switch)
1 AnalMode - Analysis mode {1: Run a time-marching simulation, 2: create a periodic linearized model} (switch)
2 NumBl - Number of blades (-)
150.0 TMax - Total run time (s)
0.006 DT - Integration time step (s)
----- TURBINE CONTROL -----
0 YCMode - Yaw control mode {0: none, 1: user-defined from routine UserYawCont, 2: user-defined from Simulink} (switch)
9999.9 TYCON - Time to enable active yaw control (s)
2 PCMode - Pitch control mode {0: none, 1: user-defined from routine PitchCntrl, 2: user-defined from Simulink} (switch)
0. TPCOn - Time to enable active pitch control (s)
3 VSContrl - Variable-speed control mode {0: none, 1: simple VS, 2: user-defined from routine UserVSCont, 3: user-defined from Simulink} (switch)
1781.98 VS_RtGnSp - Rated generator speed for simple variable-speed generator control (HSS side) (rpm) [used only when VSContrl=1]
3524.36 VS_RtTq - Rated generator torque/constant generator torque in Region 3 for simple variable-speed generator control (HSS side) (N-m) [used only when VSContrl=1]
.0008992 VS_Rgn2K - Torque constant for simple variable-speed generator control in Region 2 (HSS side) (N-m/rpm^2) [used only when VSContrl=1]
23.05 VS_SlPc - Rated generator slip percentage in Region 2 1/2 for simple variable-speed generator control (%) [used only when VSContrl=1]
1 GenModel - Generator model {1: Simple, 2: Thevenin, 3: User Defined} (-)
True GenTiStr - Method to start the generator {T: timed using TimGenOn, F: generator speed using SpdGenOn} (switch)
True GenTiStp - Method to stop the generator {T: timed using TimGenOf, F: when generator power = 0} (switch)
900.0 SpdGenOn - Generator speed to turn on the generator for a startup (HSS speed) (rpm)
0.0 TimGenOn - Time to turn on the generator for a startup (s)
9999.9 TimGenOf - Time to turn off the generator (s)
1 HSSBrMode - HSS brake model {1: simple, 2: user-defined from routine UserHSSBr} (switch)
9999.9 THSSBrDp - Time to initiate deployment of the HSS brake (s)
9999.9 TiDynBrk - Time to initiate deployment of the dynamic generator brake [CURRENTLY IGNORED] (s)
9999.9 TTPBrDp(1) - Time to initiate deployment of tip brake 1 (s)
9999.9 TTPBrDp(2) - Time to initiate deployment of tip brake 2 (s)
9999.9 TTPBrDp(3) - Time to initiate deployment of tip brake 3 (s) [unused for 2 blades]
9999.9 TBDepIsp(1) - Deployment-initiation speed for the tip brake on blade 1 (rpm)
9999.9 TBDepIsp(2) - Deployment-initiation speed for the tip brake on blade 2 (rpm)
9999.9 TBDepIsp(3) - Deployment-initiation speed for the tip brake on blade 3 (rpm) [unused for 2 blades]
9999.9 TYawManS - Time to start override yaw maneuver and end standard yaw control (s)
9999.9 TYawManE - Time at which override yaw maneuver reaches final yaw angle (s)
0.0 NacYawF - Final yaw angle for yaw maneuvers (degrees)
9999.9 TPitManS(1) - Time to start override pitch maneuver for blade 1 and end standard pitch control (s)
9999.9 TPitManS(2) - Time to start override pitch maneuver for blade 2 and end standard pitch control (s)
9999.9 TPitManS(3) - Time to start override pitch maneuver for blade 3 and end standard pitch control (s) [unused for 2 blades]
9999.9 TPitManE(1) - Time at which override pitch maneuver for blade 1 reaches final pitch (s)
9999.9 TPitManE(2) - Time at which override pitch maneuver for blade 2 reaches final pitch (s)
9999.9 TPitManE(3) - Time at which override pitch maneuver for blade 3 reaches final pitch (s) [unused for 2 blades]
-1. B1Pitch(1) - Blade 1 initial pitch (degrees)
-1. B1Pitch(2) - Blade 2 initial pitch (degrees)
11. B1Pitch(3) - Blade 3 initial pitch (degrees) [unused for 2 blades]
5.3 B1PitchF(1) - Blade 1 final pitch for pitch maneuvers (degrees)
5.3 B1PitchF(2) - Blade 2 final pitch for pitch maneuvers (degrees)
5.3 B1PitchF(3) - Blade 3 final pitch for pitch maneuvers (degrees) [unused for 2 blades]
----- ENVIRONMENTAL CONDITIONS -----
9.80665 !JASON: 9.80665 Gravity - Gravitational acceleration (m/s^2)
----- FEATURE SWITCHES -----
False FlapDOF1 - First flapwise blade mode DOF (switch)
False FlapDOF2 - Second flapwise blade mode DOF (switch)
False EdgeDOF - First edgewise blade mode DOF (switch)
False TeetDOF - Rotor-teeter DOF (switch) [unused for 3 blades]
```

False DrTrDOF - Drivetrain rotational-flexibility DOF (switch)  
 True GenDOF - Generator DOF (switch)  
 False YawDOF - Yaw DOF (switch)  
 False TwFADOF1 - First fore-aft tower bending-mode DOF (switch)  
 False TwFADOF2 - Second fore-aft tower bending-mode DOF (switch)  
 False TwSSDOF1 - First side-to-side tower bending-mode DOF (switch)  
 False TwSSDOF2 - Second side-to-side tower bending-mode DOF (switch)  
 True CompAero - Compute aerodynamic forces (switch)  
 False CompNoise - Compute aerodynamic noise (switch)

----- INITIAL CONDITIONS -----

0.0 OoPDefl - Initial out-of-plane blade-tip displacement, (meters)  
 0.0 IPDefl - Initial in-plane blade-tip deflection, (meters)  
 0.0 TeetDefl - Initial or fixed teeter angle (degrees) [unused for 3 blades]  
 0.0 Azimuth - Initial azimuth angle for blade 1 (degrees)  
 41.7 RotSpeed - Initial or fixed rotor speed (rpm)  
 -0.0 NacYaw - Initial or fixed nacelle-yaw angle (degrees)  
 0. TTDspFA - Initial fore-aft tower-top displacement (meters)  
 0.0 TTDspSS - Initial side-to-side tower-top displacement (meters)

----- TURBINE CONFIGURATION -----

21.336 TipRad - The distance from the rotor apex to the blade tip (meters)  
 1.381 HubRad - The distance from the rotor apex to the blade root (meters)  
 1 PSpnEIN - Number of the innermost blade element which is still part of the pitchable portion of the blade for partial-span pitch control [1 to BldNodes] [CURRENTLY IGNORED] (-)  
 0.000 UndSling - Undersling length [distance from teeter pin to the rotor apex] (meters) [unused for 3 blades]  
 0.210 HubCM - Distance from rotor apex to hub mass [positive downwind] (meters)  
 -3.858 OverHang - Distance from yaw axis to rotor apex [3 blades] or teeter pin [2 blades] (meters)  
 -1.1 NacCMxn - Downwind distance from the tower-top to the nacelle CM (meters)  
 0.0 NacCMyn - Lateral distance from the tower-top to the nacelle CM (meters)  
 1.734 NacCMzn - Vertical distance from the tower-top to the nacelle CM (meters)  
 34.862 TowerHt - Height of tower above ground level (meters)  
 1.734 Twr2Shft - Vertical distance from the tower top to the yaw/shaft intersection (meters)  
 0.0 TwrRBHt - Tower rigid base height (meters)  
 -3.77 ShftTilt - Rotor shaft tilt angle (degrees)  
 0.0 Delta3 - Delta-3 angle for teetering rotors (degrees) [unused for 3 blades]  
 0.0 PreCone(1) - Blade 1 cone angle (degrees)  
 0.0 PreCone(2) - Blade 2 cone angle (degrees)  
 0.0 PreCone(3) - Blade 3 cone angle (degrees) [unused for 2 blades]  
 270.0 AzimB1Up - Azimuth value to use for I/O when blade 1 points up (degrees)

----- MASS AND INERTIA -----

0.0 YawBrMass - Yaw bearing mass (kg)  
 29113. NacMass - Nacelle mass (kg)  
 5852. HubMass - Hub mass (kg)  
 0. TipMass(1) - Tip-brake mass, blade 1 (kg)  
 0. TipMass(2) - Tip-brake mass, blade 2 (kg)  
 0. TipMass(3) - Tip-brake mass, blade 3 (kg) [unused for 2 blades]  
 71750. NacYIner - Nacelle inertia about yaw axis (kg m<sup>2</sup>)  
 34.4 ! 64100. GenIner - Generator inertia about HSS (kg m<sup>2</sup>)  
 15000. HubIner - Hub inertia about teeter axis (kg m<sup>2</sup>) [unused for 3 blades]

----- DRIVETRAIN -----

100.0 GBoxEff - Gearbox efficiency (%)  
 100.0 GenEff - Generator efficiency [ignored by the Thevenin and user-defined generator models] (%)  
 43.165 !43.165 GBRatio - Gearbox ratio (-)  
 False GBRevers - Gearbox reversal {T: if rotor and generator rotate in opposite directions} (switch)  
 6000.0 HSSBrTqF - Fully deployed HSS-brake torque (N-m)  
 0.5 HSSBrDt - Time for HSS-brake to reach full deployment once initiated (sec)  
 "DynBrk.dat"DynBrkFi - File containing a mech-gen-torque vs HSS-speed curve for a dynamic brake [CURRENTLY IGNORED] (quoted string)  
 2.691e7 DTTorSpr - Drivetrain torsional spring (N-m/rad)  
 0.e0 DTTorDmp - Drivetrain torsional damper (N-m/s)

----- SIMPLE INDUCTION GENERATOR -----

0.001 SIG\_SIPc - Rated generator slip percentage [>0] (%) Now HSS side!  
 1799.98 SIG\_SySp - Synchronous (zero-torque) generator speed [>0] (rpm) Now HSS side!  
 1799.98 SIG\_RtTq - Rated torque [>0] (N-m) Now HSS side!  
 2 SIG\_PORT - Pull-out ratio (Tpullout/Trated) [>1] (-)

----- THEVENIN-EQUIVALENT INDUCTION GENERATOR -----

60.0 TEC\_Freq - Line frequency [50 or 60] (Hz)  
 6 TEC\_NP0l - Number of poles [even integer > 0] (-)  
 0.0185 TEC\_SRes - Stator resistance [>0] (ohms)  
 0.017 TEC\_RRes - Rotor resistance [>0] (ohms)  
 480.0 TEC\_VLL - Line-to-line RMS voltage (volts)  
 0.0340 TEC\_SLR - Stator leakage reactance (ohms)

0.0050 TEC\_RLR - Rotor leakage reactance (ohms)  
0.7750 TEC\_MR - Magnetizing reactance (ohms)  
----- PLATFORM MODEL -----  
0 PtfmModel - Platform model {0: none, 1: onshore, 2: fixed bottom offshore, 3: floating offshore} (switch)  
PtfmFile - Name of file containing platform properties (quoted string) [unused when PtfmModel=0]  
----- TOWER -----  
15 TwrNodes - Number of tower nodes used for analysis (-)  
"CART\_towersoft.dat" TwrFile - Name of file containing tower properties (quoted string)  
----- NACELLE-YAW -----  
0.0 YawSpr - Nacelle-yaw spring constant (N-m/rad)  
0.0 YawDamp - Nacelle-yaw constant (N-m/rad/s)  
0.0 YawNeut - Neutral yaw position--yaw spring force is zero at this yaw (degrees)  
----- FURLING -----  
False Furling - Read in additional model properties for furling turbine (flag)  
FurlFile - Name of file containing furling properties (quoted string)  
----- ROTOR-TEETER -----  
1 TeetDMod - Rotor-teeter damper model (0: none, 1: linear, 2: user-defined) (switch) [unused for 3 blades]  
0.0 TeetDmpP - Rotor-teeter damper position (degrees) [unused for 3 blades]  
0.0e4 TeetDmp - Rotor-teeter damping constant (N-m/rad/s) [unused for 3 blades]  
0.0 TeetCDmp - Rotor-teeter rate-independent Coulomb-damping moment (N-m) [unused for 3 blades]  
0.0 TeetSStP - Rotor-teeter soft-stop position (degrees) [unused for 3 blades]  
180.0 TeetHSStP - Rotor-teeter hard-stop position (degrees) [unused for 3 blades]  
0.0e4 TeetSSSp - Rotor-teeter soft-stop linear-spring constant (N-m/rad) [unused for 3 blades]  
5.0e6 TeetHSSp - Rotor-teeter hard-stop linear-spring constant (N-m/rad) [unused for 3 blades]  
----- TIP-BRAKE -----  
0.0 TBDrConN - Tip-brake drag constant during normal operation, Cd\*Area (m<sup>2</sup>)  
0.0 TBDrConD - Tip-brake drag constant during fully-deployed operation, Cd\*Area (m<sup>2</sup>)  
0.5 TpBrDT - Time for tip-brake to reach full deployment once released (sec)  
----- BLADE -----  
"CART\_blades.dat" BldFile(1) - Name of file containing properties for blade 1 (quoted string)  
"CART\_blades.dat" BldFile(2) - Name of file containing properties for blade 2 (quoted string)  
"CART\_blades.dat" BldFile(3) - Name of file containing properties for blade 3 (quoted string) [unused for 2 blades]  
----- AERODYN -----  
"AeroDyn01sim.ip" ADFile - Name of file containing AeroDyn input parameters (quoted string)  
----- NOISE -----  
"Noise.dat" NoiseFile - Name of file containing aerodynamic noise input parameters (quoted string)  
----- ADAMS -----  
"ADAMS.dat" ADAMSFile - Name of file containing ADAMS-specific input parameters (quoted string)  
----- LINEARIZATION CONTROL -----  
"CART\_Linear.dat" LinFile - Name of file containing FAST linearization parameters (quoted string)  
----- OUTPUT -----  
True SumPrint - Print summary data to "<RootName>.fsm" (switch)  
True TabDelim - Generate a tab-delimited tabular output file. (switch)  
"ES10.3E2" OutFmt - Format used for tabular output except time. Resulting field should be 10 characters. (quoted string) [not checked for validity!]  
0 TStart - Time to begin tabular output (s)  
10 DecFact - Decimation factor for tabular output [1: output every time step] (-)  
1.0 SttsTime - Amount of time between screen status messages (sec)  
0.0 NcIMUxn - Downwind distance from the tower-top to the nacelle IMU (meters)  
0.0 NcIMUyn - Lateral distance from the tower-top to the nacelle IMU (meters)  
0.0 NcIMUzn - Vertical distance from the tower-top to the nacelle IMU (meters)  
0.99 ShftGagL - Distance from rotor apex [3 blades] or teeter pin [2 blades] to shaft strain gages [positive for upwind rotors] (meters)  
2 NTwGages - Number of tower nodes that have strain gages for output [0 to 5] (-)  
4,7 TwrGagNd - List of tower nodes that have strain gages [1 to TwrNodes] (-) [unused if NTwGages=0]  
3 NBIGages - Number of blade nodes that have strain gages for output [0 to 5] (-)  
7,12,15 BldGagNd - List of blade nodes that have strain gages [1 to BldNodes] (-)  
OutList - The next line(s) contains a list of output parameters. See OutList.txt for a listing of available output channels, (-)  
"Azimuth,LSSGagP" - Rotor and Gen Azimuth Angles  
"WindVxi" - Hub height windspeed  
"LSSGagV,HSShftV,LSSTipVxa"  
"LSSGagAxa,HSShftA" - Low-speed shaft vel. and generator vel.  
"BldPitch1,BldPitch2" - Blade 1 and 2 pitch angles  
"YawBrTDxt,YawBrTDyt" - Tower-top fore-aft and side-side displ  
"TwHt1MLxt,TwHt1MLyt"  
"TipDxb1,TipDxb2"  
"TeetPya"  
"TipDxc1,TipDyc1" - Blade 1 tip out-plane and in-plane defl  
"TipDxc2,TipDyc2" - Blade 2 tip out-plane and in-plane defl  
"RotTorq,LSShftTq,HSShftTq" - Rotor and shaft torque  
"GenTq,RotThrust" - Generator torque and rotor thrust  
"RotPwr,GenPwr,HSShftPwr" - rotor power

```

"TipSpdRat,RotCp"
"YawBrTAyp"
"YawBrTAxp"
"YawBrTDyp"
"LSSTipPxa"
"YawBrMzn"
"NcIMUTVys"
"RootMyb1"
"RootMyb2"
"RootMxb1"
"RootMxb2"
"RootFxb1"
"RootFxc1"
"LSSTipAxa"
"HSShftA"
"TwrBsMxt"
"YawBrFyp"
"YawBrMxp"
"LSShftFys"
END of FAST input file (the word "END" must appear in the first 3 columns of this last line).

```

---

## Aerodyn Input File Example (Aerodyn01sim.ipt)

```

CART aerodynamic parameters for FAST.
SI SysUnits - System of units for used for input and output [must be SI for FAST] (unquoted string)
STEADY StallMod - Dynamic stall included [BEDDOES or STEADY] (unquoted string)
NO_CM UseCm - Use aerodynamic pitching moment model? [USE_CM or NO_CM] (unquoted string)
EQUIL !JASON:DYNIN InfModel - Inflow model [DYNIN or EQUIL] (unquoted string)
WAKE IndModel - Induction-factor model [NONE or WAKE or SWIRL] (unquoted string)
0.001 !JASON: 0.001 AToler - Induction-factor tolerance (convergence criteria) (-)
PRANDTL TLModel - Tip-loss model (EQUIL only) [PRANDtl, GTECH, or NONE] (unquoted string)
NONE HLMModel - Hub-loss model (EQUIL only) [PRANDtl or NONE] (unquoted string)
"Wind/StepUp.wnd" Name of file containing wind data (quoted string)
36.850 HH - Wind reference (hub) height [TowerHt+Twr2Shft+OverHang*SIN(NacTilt)] (m)
0.05 !JASON: 0.3 TwrShad - Tower-shadow velocity deficit (-)
3.0 ShadHWid - Tower-shadow half width (m)
4.0 T_Shad_Refpt - Tower-shadow reference point (m)
1.03 Rho - Air density (kg/m^3)
1.4639e-5 KinVisc - Kinematic air viscosity [CURRENTLY IGNORED] (m^2/sec)
0.002 DTAero - Time interval for aerodynamic calculations (sec)
11 NumFoil - Number of airfoil files (-)
"AeroData\art15.air" FoilNm - Names of the airfoil files [NumFoil lines] (quoted strings)
"AeroData\art25.air"
"AeroData\art35.air"
"AeroData\art45.air"
"AeroData\art55.air"
"AeroData\art65.air"
"AeroData\art75.air"
"AeroData\art75-5.air"
"AeroData\art85.air"
"AeroData\art85-5.air"
"AeroData\art95.air"
20 BldNodes - Number of blade nodes used for analysis (-)
RNodes AeroTwst DRNodes Chord NFoil PrnElm
1.8799 3.3740 0.998 1.1929 1 PRINT
2.8777 3.1895 0.998 1.3286 1 PRINT
3.8754 3.0569 0.998 1.4276 1 PRINT
4.8731 2.8685 0.998 1.5637 1 PRINT
5.8709 2.7371 0.998 1.6633 2 PRINT
6.8686 2.5294 0.998 1.6575 2 PRINT
7.8663 2.3700 0.998 1.6163 3 PRINT
8.8641 2.1379 0.998 1.5555 3 PRINT
9.8618 1.9386 0.998 1.5017 4 PRINT
10.8595 1.6665 0.998 1.4274 4 PRINT
11.8573 1.4339 0.998 1.3735 5 PRINT
12.8550 1.0945 0.998 1.3000 5 PRINT
13.8528 0.8374 0.998 1.2461 6 PRINT
14.8506 0.4020 0.998 1.1718 6 PRINT
15.8483 0.0770 0.998 1.1179 7 PRINT

```

```

16.8460 -0.4568 0.998 1.0444 7      PRINT
17.8438 -0.8951 0.998 0.9906 8      PRINT
18.8416 -1.5209 0.998 0.9171 9      PRINT
19.8393 -2.1452 0.998 0.8626 10     PRINT
20.8371 -2.9979 0.998 0.7889 11     PRINT

```

## Wind Input File Example 1 (StepUp.wnd)

```

! Wind file for CART turbine (StepUp.wnd).
! TimeWind      Wind      Vert.      Horiz.      Vert.      LinV      Gust
!   Speed      Dir      Speed      Shear      Shear      Shear      Speed
0.0  5.0  0.0  0.0  0.0  0.0  0.0  0.0
19.9 5.0  0.0  0.0  0.0  0.0  0.0  0.0
20.0 5.0  0.0  0.0  0.0  0.0  0.0  0.0
39.9 5.0  0.0  0.0  0.0  0.0  0.0  0.0
40.0 5.0  0.0  0.0  0.0  0.0  0.0  0.0
59.9 5.5  0.0  0.0  0.0  0.0  0.0  0.0
60.0 6.0  0.0  0.0  0.0  0.0  0.0  0.0
79.9 6.0  0.0  0.0  0.0  0.0  0.0  0.0
80.0 7.0  0.0  0.0  0.0  0.0  0.0  0.0
99.9 7.0  0.0  0.0  0.0  0.0  0.0  0.0
100.0 8.0  0.0  0.0  0.0  0.0  0.0  0.0
119.9 8.0  0.0  0.0  0.0  0.0  0.0  0.0
120.0 9.0  0.0  0.0  0.0  0.0  0.0  0.0
139.9 9.0  0.0  0.0  0.0  0.0  0.0  0.0
140.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0
159.9 10.0  0.0  0.0  0.0  0.0  0.0  0.0
160.0 11.0  0.0  0.0  0.0  0.0  0.0  0.0
179.9 11.0  0.0  0.0  0.0  0.0  0.0  0.0
180.0 12.0  0.0  0.0  0.0  0.0  0.0  0.0
199.9 12.0  0.0  0.0  0.0  0.0  0.0  0.0
200.0 12.0  0.0  0.0  0.0  0.0  0.0  0.0
219.9 12.0  0.0  0.0  0.0  0.0  0.0  0.0
220.0 13.0  0.0  0.0  0.0  0.0  0.0  0.0
239.9 13.0  0.0  0.0  0.0  0.0  0.0  0.0
240.0 10.0  0.0  0.0  0.0  0.0  0.0  0.0
259.9 10.0  0.0  0.0  0.0  0.0  0.0  0.0
260.0 10.5  0.0  0.0  0.0  0.0  0.0  0.0
279.9 10.5  0.0  0.0  0.0  0.0  0.0  0.0
280.0 11.0  0.0  0.0  0.0  0.0  0.0  0.0
299.9 11.0  0.0  0.0  0.0  0.0  0.0  0.0
300.0 11.5  0.0  0.0  0.0  0.0  0.0  0.0
319.9 11.5  0.0  0.0  0.0  0.0  0.0  0.0
320.0 12.0  0.0  0.0  0.0  0.0  0.0  0.0
339.9 12.0  0.0  0.0  0.0  0.0  0.0  0.0
340.0 12.0  0.0  0.0  0.0  0.0  0.0  0.0
359.9 20.0  0.0  0.0  0.0  0.0  0.0  0.0
360.0 21.0  0.0  0.0  0.0  0.0  0.0  0.0
379.9 21.0  0.0  0.0  0.0  0.0  0.0  0.0
380.0 22.0  0.0  0.0  0.0  0.0  0.0  0.0
399.9 22.0  0.0  0.0  0.0  0.0  0.0  0.0
400.0 23.0  0.0  0.0  0.0  0.0  0.0  0.0
419.9 23.0  0.0  0.0  0.0  0.0  0.0  0.0
420.0 24.0  0.0  0.0  0.0  0.0  0.0  0.0
439.9 24.0  0.0  0.0  0.0  0.0  0.0  0.0
440.0 25.0  0.0  0.0  0.0  0.0  0.0  0.0
459.9 25.0  0.0  0.0  0.0  0.0  0.0  0.0
9999.9 25.0  0.0  0.0  0.0  0.0  0.0  0.0

```

## Wind Input File Example 2 (step18mps-5.wnd)

! Wind file for CART turbine.

! Time	Wind Speed	Wind Dir	Vert. Speed	Horiz. Shear	Vert. Shear	LinV Shear	Gust Speed
0.0	17.0	0.0	0.0	0.0	0.0	0.0	0.0
0.1	17.0	0.0	0.0	0.0	0.0	0.0	0.0
10.0	17.0	0.0	0.0	0.0	0.0	0.0	0.0
15.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
15.1	17.0	0.0	0.0	0.0	0.0	0.0	0.0
20.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
20.1	17.0	0.0	0.0	0.0	0.0	0.0	0.0
25.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
25.1	17.0	0.0	0.0	0.0	0.0	0.0	0.0
30.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
30.10	17.0	0.0	0.0	0.0	0.0	0.0	0.0
35.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
35.1	17.0	0.0	0.0	0.0	0.0	0.0	0.0
40.05	17.0	0.0	0.0	0.0	0.0	0.0	0.0
40.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
50.05	18.0	0.0	0.0	0.0	0.0	0.0	0.0
50.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
60.05	18.0	0.0	0.0	0.0	0.0	0.0	0.0
60.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
70.05	18.0	0.0	0.0	0.0	0.0	0.0	0.0
70.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
80.05	18.0	0.0	0.0	0.0	0.0	0.0	0.0
80.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
100.1	18.0	0.0	0.0	0.0	0.0	0.0	0.0
930.05	18.0	0.0	0.0	0.0	0.0	0.0	0.0

## A.2 FAST Linearization Files for the CART (Chapters 3 and 4)

### FAST Input File Example (cartnewlin.fad)

```

----- FAST INPUT FILE for Turbine Linearization-----
CART Example.
Compatible with FAST v6.01.
----- SIMULATION CONTROL -----
False Echo - Echo input data to "echo.out" (switch)
1 ADAMSPrep - ADAMS preprocessor mode {1: Run FAST, 2: use FAST as a preprocessor to create an ADAMS model, 3: do
both} (switch)
2 AnalMode - Analysis mode {1: Run a time-marching simulation, 2: create a periodic linearized model} (switch)
2 NumBl - Number of blades (-)
1200.0 TMax - Total run time (s)
0.006 DT - Integration time step (s)
----- TURBINE CONTROL -----
0 YCMode - Yaw control mode {0: none, 1: user-defined from routine UserYawCont, 2: user-defined from Simulink} (switch)
9999.9 TYCOn - Time to enable active yaw control (s)
0 PCMode - Pitch control mode {0: none, 1: user-defined from routine PitchCntrl, 2: user-defined from Simulink} (switch)
10. TPCOn - Time to enable active pitch control (s)
1 VSContrl - Variable-speed control {0: none, 1: simple VS, 2: user-defined VS} (switch)
1800.0 RatGenSp - Rated generator speed for simple variable-speed generator control (HSS side) (rpm) [used only when
VSContrl=1]
3526. VS_RtTq - Rated generator torque/constant generator torque in Region 3 for simple variable-speed generator control (HSS
side) (N-m) [used only when VSContrl=1]
.001088 VS_Rgn2K - Torque constant for simple variable-speed generator control in Region 2 (HSS side) (N-m/rpm^2) [used
only when VSContrl=1]
10.0 VS_SIPc - Rated generator slip percentage in Region 2 1/2 for simple variable-speed generator control (%) [used only when
VSContrl=1]
3 GenModel - Generator model {1: Simple, 2: Thevenin, 3: User Defined} (-)
True GenTiStr - Method to start the generator {T: timed using TimGenOn, F: generator speed using SpdGenOn} (switch)
True GenTiStp - Method to stop the generator {T: timed using TimGenOf, F: when generator power = 0} (switch)
900.0 SpdGenOn - Generator speed to turn on the generator for a startup (HSS speed) (rpm)
0.0 TimGenOn - Time to turn on the generator for a startup (s)
9999.9 TimGenOf - Time to turn off the generator (s)

```

1 HSSBrMode - HSS brake model {1: simple, 2: user-defined from routine UserHSSBr} (switch)  
 9999.9 THSSBrDp - Time to initiate deployment of the HSS brake (s)  
 9999.9 TiDynBrk - Time to initiate deployment of the dynamic generator brake [CURRENTLY IGNORED] (s)  
 9999.9 TTpBrDp(1) - Time to initiate deployment of tip brake 1 (s)  
 9999.9 TTpBrDp(2) - Time to initiate deployment of tip brake 2 (s)  
 9999.9 TTpBrDp(3) - Time to initiate deployment of tip brake 3 (s) [unused for 2 blades]  
 9999.9 TBDeplSp(1) - Deployment-initiation speed for the tip brake on blade 1 (rpm)  
 9999.9 TBDeplSp(2) - Deployment-initiation speed for the tip brake on blade 2 (rpm)  
 9999.9 TBDeplSp(3) - Deployment-initiation speed for the tip brake on blade 3 (rpm) [unused for 2 blades]  
 9999.9 TYawManS - Time to start override yaw maneuver and end standard yaw control (s)  
 9999.9 TYawManE - Time at which override yaw maneuver reaches final yaw angle (s)  
 0.0 NacYawF - Final yaw angle for yaw maneuvers (degrees)  
 9999.9 TPitManS(1) - Time to start override pitch maneuver for blade 1 and end standard pitch control (s)  
 9999.9 TPitManS(2) - Time to start override pitch maneuver for blade 2 and end standard pitch control (s)  
 9999.9 TPitManS(3) - Time to start override pitch maneuver for blade 3 and end standard pitch control (s) [unused for 2 blades]  
 9999.9 TPitManE(1) - Time at which override pitch maneuver for blade 1 reaches final pitch (s)  
 9999.9 TPitManE(2) - Time at which override pitch maneuver for blade 2 reaches final pitch (s)  
 9999.9 TPitManE(3) - Time at which override pitch maneuver for blade 3 reaches final pitch (s) [unused for 2 blades]  
 11 B1Pitch(1) - Blade 1 initial pitch (degrees)  
 11 B1Pitch(2) - Blade 2 initial pitch (degrees)  
 11 B1Pitch(3) - Blade 3 initial pitch (degrees) [unused for 2 blades]  
 5.3 B1PitchF(1) - Blade 1 final pitch for pitch maneuvers (degrees)  
 5.3 B1PitchF(2) - Blade 2 final pitch for pitch maneuvers (degrees)  
 5.3 B1PitchF(3) - Blade 3 final pitch for pitch maneuvers (degrees) [unused for 2 blades]  
 ----- ENVIRONMENTAL CONDITIONS -----  
 9.80665 !JASON: 9.80665 Gravity - Gravitational acceleration (m/s^2)  
 ----- FEATURE SWITCHES -----  
 True FlapDOF1 - First flapwise blade mode DOF (switch)  
 False FlapDOF2 - Second flapwise blade mode DOF (switch)  
 False EdgeDOF - First edgewise blade mode DOF (switch)  
 False TeetDOF - Rotor-teeter DOF (switch) [unused for 3 blades]  
 True DrTrDOF - Drivetrain rotational-flexibility DOF (switch)  
 True GenDOF - Generator DOF (switch)  
 False YawDOF - Yaw DOF (switch)  
 False TwFADOF1 - First fore-aft tower bending-mode DOF (switch)  
 False TwFADOF2 - Second fore-aft tower bending-mode DOF (switch)  
 False TwSSDOF1 - First side-to-side tower bending-mode DOF (switch)  
 False TwSSDOF2 - Second side-to-side tower bending-mode DOF (switch)  
 True CompAero - Compute aerodynamic forces (switch)  
 False CompNoise - Compute aerodynamic noise (switch)  
 ----- INITIAL CONDITIONS -----  
 0.0 OoPDefl - Initial out-of-plane blade-tip displacement, (meters)  
 0.0 IPDefl - Initial in-plane blade-tip deflection, (meters)  
 0.0 TeetDefl - Initial or fixed teeter angle (degrees) [unused for 3 blades]  
 0.0 Azimuth - Initial azimuth angle for blade 1 (degrees)  
 41.7 RotSpeed - Initial or fixed rotor speed (rpm)  
 -0.0 NacYaw - Initial or fixed nacelle-yaw angle (degrees)  
 0.0 TTDspFA - Initial fore-aft tower-top displacement (meters)  
 0.0 TTDspSS - Initial side-to-side tower-top displacement (meters)  
 ----- TURBINE CONFIGURATION -----  
 21.336 TipRad - The distance from the rotor apex to the blade tip (meters)  
 1.381 HubRad - The distance from the rotor apex to the blade root (meters)  
 1 PSpnEIN - Number of the innermost blade element which is still part of the pitchable portion of the blade for partial-span pitch control [1 to BldNodes] [CURRENTLY IGNORED] (-)  
 0.000 UndSling - Undersling length [distance from teeter pin to the rotor apex] (meters) [unused for 3 blades]  
 0.210 HubCM - Distance from rotor apex to hub mass [positive downwind] (meters)  
 -3.858 OverHang - Distance from yaw axis to rotor apex [3 blades] or teeter pin [2 blades] (meters)  
 -1.1 NacCMxn - Downwind distance from the tower-top to the nacelle CM (meters)  
 0.0 NacCMyn - Lateral distance from the tower-top to the nacelle CM (meters)  
 1.734 NacCMzn - Vertical distance from the tower-top to the nacelle CM (meters)  
 34.862 TowerHt - Height of tower above ground level (meters)  
 1.734 Twr2Shft - Vertical distance from the tower top to the yaw/shaft intersection (meters)  
 0.0 TwrRBHt - Tower rigid base height (meters)  
 -3.77 ShftTilt - Rotor shaft tilt angle (degrees)  
 0.0 Delta3 - Delta-3 angle for teetering rotors (degrees) [unused for 3 blades]  
 0.0 PreCone(1) - Blade 1 cone angle (degrees)  
 0.0 PreCone(2) - Blade 2 cone angle (degrees)  
 0.0 PreCone(3) - Blade 3 cone angle (degrees) [unused for 2 blades]  
 0.0 AzimB1Up - Azimuth value to use for I/O when blade 1 points up (degrees)  
 ----- MASS AND INERTIA -----

0.0 YawBrMass - Yaw bearing mass (kg)  
 29113. NacMass - Nacelle mass (kg)  
 5852. HubMass - Hub mass (kg)  
 0. TipMass(1) - Tip-brake mass, blade 1 (kg)  
 0. TipMass(2) - Tip-brake mass, blade 2 (kg)  
 0. TipMass(3) - Tip-brake mass, blade 3 (kg) [unused for 2 blades]  
 71750. NacYIner - Nacelle inertia about yaw axis (kg m<sup>2</sup>)  
 34.4 !64100. GenIner - Generator inertia about HSS (kg m<sup>2</sup>)  
 15000. HubIner - Hub inertia about teeter axis (kg m<sup>2</sup>) [unused for 3 blades]  
 ----- DRIVETRAIN -----  
 100.0 GBoxEff - Gearbox efficiency (%)  
 100.0 GenEff - Generator efficiency [ignored by the Thevenin and user-defined generator models] (%)  
 43.165 !43.165 GBRatio - Gearbox ratio (-)  
 False GBRevers - Gearbox reversal {T: if rotor and generator rotate in opposite directions} (switch)  
 6000.0 HSSBrTqF - Fully deployed HSS-brake torque (N-m)  
 0.5 HSSBrDt - Time for HSS-brake to reach full deployment once initiated (sec)  
 "DynBrk.dat" DynBrkFi - File containing a mech-gen-torque vs HSS-speed curve for a dynamic brake [CURRENTLY  
 IGNORED] (quoted string)  
 2.691e7 DTTorSpr - Drivetrain torsional spring (N-m/rad)  
 2.691e0 DTTorDmp - Drivetrain torsional damper (N-m/s)  
 ----- SIMPLE INDUCTION GENERATOR -----  
 0.001 SIG\_SIPc - Rated generator slip percentage [>0] (%) Now HSS side!  
 1799.98 SIG\_SySp - Synchronous (zero-torque) generator speed [>0] (rpm) Now HSS side!  
 1799.98 SIG\_RtTq - Rated torque [>0] (N-m) Now HSS side!  
 2 SIG\_PORT - Pull-out ratio (Tpullout/Trated) [>1] (-)  
 ----- THEVENIN-EQUIVALENT INDUCTION GENERATOR -----  
 60.0 TEC\_Freq - Line frequency [50 or 60] (Hz)  
 6 TEC\_NPol - Number of poles [even integer > 0] (-)  
 0.0185 TEC\_SRes - Stator resistance [>0] (ohms)  
 0.017 TEC\_RRes - Rotor resistance [>0] (ohms)  
 480.0 TEC\_VLL - Line-to-line RMS voltage (volts)  
 0.0340 TEC\_SLR - Stator leakage reactance (ohms)  
 0.0050 TEC\_RLR - Rotor leakage reactance (ohms)  
 0.7750 TEC\_MR - Magnetizing reactance (ohms)  
 ----- PLATFORM MODEL -----  
 0 PtfmModel - Platform model {0: none, 1: onshore, 2: fixed bottom offshore, 3: floating offshore} (switch)  
 PtfmFile - Name of file containing platform properties (quoted string) [unused when PtfmModel=0]  
 ----- TOWER -----  
 15 TwrNodes - Number of tower nodes used for analysis (-)  
 "CART\_Towersoft.dat" TwrFile - Name of file containing tower properties (quoted string)  
 ----- NACELLE-YAW -----  
 0.0 YawSpr - Nacelle-yaw spring constant (N-m/rad)  
 0.0 YawDamp - Nacelle-yaw constant (N-m/rad/s)  
 0.0 YawNeut - Neutral yaw position--yaw spring force is zero at this yaw (degrees)  
 ----- FURLING -----  
 False Furling - Read in additional model properties for furling turbine (flag)  
 FurlFile - Name of file containing furling properties (quoted string)  
 ----- ROTOR-TEETER -----  
 1 TeetDMod - Rotor-teeter damper model (0: none, 1: linear, 2: user-defined) (switch) [unused for 3 blades]  
 0.0 TeetDmpP - Rotor-teeter damper position (degrees) [unused for 3 blades]  
 4.0e4 TeetDmp - Rotor-teeter damping constant (N-m/rad/s) [unused for 3 blades]  
 0.0 TeetCDmp - Rotor-teeter rate-independent Coulomb-damping moment (N-m) [unused for 3 blades]  
 0.0 TeetSStP - Rotor-teeter soft-stop position (degrees) [unused for 3 blades]  
 180.0 TeetHStP - Rotor-teeter hard-stop position (degrees) [unused for 3 blades]  
 1.0 TeetSSSp - Rotor-teeter soft-stop linear-spring constant (N-m/rad) [unused for 3 blades]  
 5.0e6 TeetHSSp - Rotor-teeter hard-stop linear-spring constant (N-m/rad) [unused for 3 blades]  
 ----- TIP-BRAKE -----  
 0.0 TBDrConN - Tip-brake drag constant during normal operation, Cd\*Area (m<sup>2</sup>)  
 0.0 TBDrConD - Tip-brake drag constant during fully-deployed operation, Cd\*Area (m<sup>2</sup>)  
 0.5 TpBrDT - Time for tip-brake to reach full deployment once released (sec)  
 ----- BLADE -----  
 "CART\_blades.dat" BldFile(1) - Name of file containing properties for blade 1 (quoted string)  
 "CART\_blades.dat" BldFile(2) - Name of file containing properties for blade 2 (quoted string)  
 "CART\_blades.dat" BldFile(3) - Name of file containing properties for blade 3 (quoted string) [unused for 2 blades]  
 ----- AERODYN -----  
 "AeroDyn01lin.ipt" ADFile - Name of file containing AeroDyn input parameters (quoted string)  
 ----- NOISE -----  
 "Noise.dat" NoiseFile - Name of file containing aerodynamic noise input parameters (quoted string)  
 ----- ADAMS -----  
 "ADAMS.dat" ADAMSFile - Name of file containing ADAMS-specific input parameters (quoted string)

```

----- LINEARIZATION CONTROL -----
"CART_Linear.dat"      LinFile - Name of file containing FAST linearization parameters (quoted string)
----- OUTPUT -----
True SumPrint - Print summary data to "<RootName>.fsm" (switch)
True TabDelim - Generate a tab-delimited tabular output file. (switch)
"ES10.3E2" OutFmt - Format used for tabular output except time. Resulting field should be 10 characters. (quoted string) [not
checked for validity!]
20 TStart - Time to begin tabular output (s)
10 DecFact - Decimation factor for tabular output [1: output every time step] (-)
1.0 SttsTime - Amount of time between screen status messages (sec)
0.0 NeIMUxn - Downwind distance from the tower-top to the nacelle IMU (meters)
0.0 NeIMUyn - Lateral distance from the tower-top to the nacelle IMU (meters)
0.0 NeIMUzn - Vertical distance from the tower-top to the nacelle IMU (meters)
0.99 ShftGagL - Distance from rotor apex [3 blades] or teeter pin [2 blades] to shaft strain gages [positive for upwind rotors]
(meters)
2 NTwGages - Number of tower nodes that have strain gages for output [0 to 5] (-)
4,7 TwrGagNd - List of tower nodes that have strain gages [1 to TwrNodes] (-) [unused if NTwGages=0]
3 NBIGages - Number of blade nodes that have strain gages for output [0 to 5] (-)
7,12,15 BldGagNd - List of blade nodes that have strain gages [1 to BldNodes] (-)
OutList - The next line(s) contains a list of output parameters. See OutList.txt for a listing of available output channels, (-)
"HSShftV"
END of FAST input file (the word "END" must appear in the first 3 columns of this last line).
-----

```

## Aerodyn Input File Example (Aerodynlin.ipt)

```

CART aerodynamic parameters for FAST.
SI SysUnits - System of units for used for input and output [must be SI for FAST] (unquoted string)
STEADY StallMod - Dynamic stall included [BEDDOES or STEADY] (unquoted string)
NO_CM UseCm - Use aerodynamic pitching moment model? [USE_CM or NO_CM] (unquoted string)
EQUIL          !JASON:DYNIN InfModel - Inflow model [DYNIN or EQUIL] (unquoted string)
WAKE IndModel - Induction-factor model [NONE or WAKE or SWIRL] (unquoted string)
0.001          !JASON: 0.001 AToler - Induction-factor tolerance (convergence criteria) (-)
PRANDTL TLModel - Tip-loss model (EQUIL only) [PRANDtl, GTECH, or NONE] (unquoted string)
NONE          HLModel - Hub-loss model (EQUIL only) [PRANDtl or NONE] (unquoted string)
"Wind/CONST18mps.wnd" Name of file containing wind data (quoted string)
36.850 HH - Wind reference (hub) height [TowerHt+Twr2Shft+OverHang*SIN(NacTilt)] (m)
0.3          !JASON: 0.3 TwrShad - Tower-shadow velocity deficit (-)
3.0 ShadHWid - Tower-shadow half width (m)
4.0 T_Shad_Refpt - Tower-shadow reference point (m)
1.03 Rho - Air density (kg/m^3)
1.4639e-5 KinVisc - Kinematic air viscosity [CURRENTLY IGNORED] (m^2/sec)
0.002 DTAero - Time interval for aerodynamic calculations (sec)
11 NumFoil - Number of airfoil files (-)
"AeroData\art15.air" FoilNm - Names of the airfoil files [NumFoil lines] (quoted strings)
"AeroData\art25.air"
"AeroData\art35.air"
"AeroData\art45.air"
"AeroData\art55.air"
"AeroData\art65.air"
"AeroData\art75.air"
"AeroData\art75-5.air"
"AeroData\art85.air"
"AeroData\art85-5.air"
"AeroData\art95.air"
20 BldNodes - Number of blade nodes used for analysis (-)
RNodes AeroTwst DRNodes Chord NFoil PmElm
1.8799 3.3740 0.998 1.1929 1 NOPRINT
2.8777 3.1895 0.998 1.3286 1      NOPRINT
3.8754 3.0569 0.998 1.4276 1 NOPRINT
4.8731 2.8685 0.998 1.5637 1 NOPRINT
5.8709 2.7371 0.998 1.6633 2      NOPRINT
6.8686 2.5294 0.998 1.6575 2 NOPRINT
7.8663 2.3700 0.998 1.6163 3 NOPRINT
8.8641 2.1379 0.998 1.5555 3 NOPRINT
9.8618 1.9386 0.998 1.5017 4      NOPRINT
10.8595 1.6665 0.998 1.4274 4 NOPRINT
11.8573 1.4339 0.998 1.3735 5      NOPRINT
12.8550 1.0945 0.998 1.3000 5 NOPRINT

```

```

13.8528 0.8374 0.998 1.2461 6 NOPRINT
14.8506 0.4020 0.998 1.1718 6 NOPRINT
15.8483 0.0770 0.998 1.1179 7      NOPRINT
16.8460 -0.4568 0.998 1.0444 7      PRINT
17.8438 -0.8951 0.998 0.9906 8      NOPRINT
18.8416 -1.5209 0.998 0.9171 9      PRINT
19.8393 -2.1452 0.998 0.8626 10     NOPRINT
20.8371 -2.9979 0.998 0.7889 11     PRINT

```

## FAST Linearization Parameter File Example (Cart\_linear.dat)

```

----- FAST LINEARIZATION CONTROL FILE -----
CART linearization input properties.
----- PERIODIC STEADY STATE SOLUTION -----
True CalcStdy - Calculate periodic steady state condition (False: linearize about zero) (switch)
3 TrimCase - Trim case {1: find nacelle yaw, 2: find generator torque, 3: find collective blade pitch} (switch) [used only when
CalcStdy=True and GenDOF=True]
0.0001 DispTol - Convergence tolerance for the 2-norm of displacements in the periodic steady state calculation (rad ) [used only
when CalcStdy=True]
0.00001 VelTol - Convergence tolerance for the 2-norm of velocities in the periodic steady state calculation (rad/s) [used only
when CalcStdy=True]
----- MODEL LINEARIZATION -----
24 NAzimStep - Number of equally-spaced azimuth steps in periodic linearized model (-)
1 MdlOrder - Order of output linearized model {1: 1st order A, B, Bd, C, D, Dd; 2: 2nd order M, C, K, F, Fd, VelC, DspC, D,
Dd} (switch)
----- INPUTS AND DISTURBANCES -----
1 NInputs - Number of control inputs [0 (none) or 1 to 4+NumBl] (-)
4 CntrlInpt - CntrlInpt - List of control inputs [1 to NInputs] {1: nacelle yaw angle, 2: nacelle yaw rate, 3: generator torque, 4:
collective blade pitch, 5: individual pitch of blade 1, 6: individual pitch of blade 2, 7: individual pitch of blade 3 [unavailable for 2-
bladed turbines]} (-) [unused if NInputs=0]
1 NDisturbs - Number of wind disturbances [0 (none) or 1 to 7] (-)
1 Disturbnc - List of input wind disturbances [1 to NDisturbs] {1: horizontal hub-height wind speed, 2: horizontal wind direction,
3: vertical wind speed, 4: horizontal wind shear, 5: vertical power law wind shear, 6: linear vertical wind shear, 7: horizontal hub-
height wind gust} (-) [unused if NDisturbs=0]

```

## Appendix B: MATLAB Control Design Scripts

### B.1 Files for the Controllers Designed in Section 4.2 (Rotor Collective Pitch Control):

#### B.1.1 MATLAB Script for Control Design Described in Section 4.1.3: (“LQR\_design\_DAC6states.m”):

```
% LQR_design.m: Control design using Linear Quadratic Regulation (LQR)
% Run a FAST linearization first (.lin file must be present)
% Pitch control inputs
%
% A. Wright
% 7/07
%

disp('Performing a time-invariant LQR design');

%=====
% Initialization
%=====

clear az_prev counter AMat BMat CMat

% Read the FAST linear output file (.lin) containing state-space model data
Read_linfile

az = [0: 2*pi/NAzimStep: 2*pi]; % vector of azimuth angles
omega_op = Avgxop(az_stanenum + NActvDOF); % mean operating pt rotor speed

% Initialize global variables for 'azimuth_unwind' function
global az_prev counter
az_prev = 0;
counter = 0;

%=====
% Add pitch actuator dynamics to plant model
%=====

% 6 states:
% drive-train torsional defl
% rotor 1st symmetric flap disp
% gen speed
% drive-train torsional vel
% rotor 1st symmetric flap vel.
% uniform wind component

Amod = [[0,0,0,1,0
0,0,0,0,1
AvgAMat(5,2),2*AvgAMat(5,3),AvgAMat(5,5),AvgAMat(5,6),2*AvgAMat(5,7)
AvgAMat(6,2),2*AvgAMat(6,3),AvgAMat(6,5),AvgAMat(6,6),2*AvgAMat(6,7)
AvgAMat(7,2),(AvgAMat(7,3)+AvgAMat(7,4)),AvgAMat(7,5),AvgAMat(7,6),(AvgAMat(7,7)+AvgAMat(7,8))]]

NStates = size(Amod,1);

Bmod = [[AvgBMat(1,1),
.5*(AvgBMat(3,1)+AvgBMat(4,1)),
AvgBMat(5,1),
AvgBMat(6,1),
.5*(AvgBMat(7,1)+AvgBMat(8,1))]]

Bdmod = [[AvgBdMat(1,1),
.5*(AvgBdMat(3,1)+AvgBdMat(4,1)),
AvgBdMat(5,1),
AvgBdMat(6,1),
```

```

.5*(AvgBdMat(7,1)+AvgBdMat(8,1))]

Cmod = [AvgCMat(1,2),.5*(AvgCMat(1,3)+AvgCMat(1,4)),AvgCMat(1,5),AvgCMat(1,6),.5*(AvgCMat(1,7)+AvgCMat(1,8))]

A_c = Amod;
Bd_c = Bdmod;
B_c = Bmod;
C_c = Cmod;
D_c = AvgDMat;
Dd_c = AvgDdMat;

[nump,denp]=ss2tf(A_c,B_c,C_c,D_c)

eig(A_c)

%=====
% Wind disturbance generator
%=====

F_dac = 0; % for step disturbance generator
theta_dac = 1; %
ndstates = size(F_dac,1); % number of disturbance states
ndinp = size(theta_dac,1); % number of disturbance inputs

%=====
% FSFB design
%=====

disp('__FSFB design__');

% Define state and input weightings
%Q = diag([0,0,0,0, 1e2,1e2,0,0, 0,0,1e3]);
%R = diag([1000,1]); %eye(NCtrlInpts); % input weightings
%disp(['State weightings: Q = diag([' ,num2str(diag(Q)),''])]);
%disp(['Input weightings: R = diag([' ,num2str(diag(R)),''])]);

% check controllability
rnk = rank(ctrb(A_c,B_c), 0); % rank with tol = 0
if rnk == size(A_c,1)
    disp(' (A_c, B_c) controllable');
else
    disp([' (A_c, B_c) uncontrollable, rank = ', num2str(rnk), ' < ', num2str(size(A_c,1))]);
    error(' Cannot continue!');
end

% LQR design using pitch and torque
R=1
Q=[.00001,0,0,0,0
0,0,0,0,0
0,0,6,0,0
0,0,0,8,0
0,0,0,0,0]
[K,S,E]=lqr(A_c, B_c, Q, R);
E
Gx = -K;

%pnum = [-2+22.6i,-2-22.6i,-3.5+15i,-3.5-15i,-2,-10,-60]
%Gx = -place(A_c, B_c,pnum)

```

```

%=====
% State Estimator design
%=====

disp(' ');
prompt = input('State Estimator design (default: n)? ', 's');
if isempty(prompt)
    prompt = 'n';
end

if prompt == 'y'
    disp(' __ State Estimator design __ ');

    % Augment with disturbance estimate

    Abar = [A_c, Bd_c*theta_dac; zeros(ndstates,NStates), F_dac];
    Bbar = [B_c; zeros(ndstates,NCtrlInpts)];

    C_c
    zeros(NCtrlOuts,ndstates)
    Cbar = [C_c, zeros(NCtrlOuts,ndstates)];

    eig(Abar)

    Dbar = 0.

    % Check observability
    rnk = rank(observ(Abar,Cbar), 0); % rank with tol = 0
    if rnk == size(Abar,1)
        disp(' (Abar, Cbar) observable');
    else
        disp([' (Abar, Cbar) unobservable, rank = ', num2str(rnk), ' < ', num2str(size(Abar,1))]);
        error(' Cannot continue!');
    end

    % Define estimated state and measurement weightings
    % QE = diag([1,1,1,1, 1,1,1e5,1, 1,1,1, 1e5]);
    % RE = eye(size(Cbar,1));

    % disp(['State weightings: QE = diag([' ,num2str(diag(QE)) ,'])]);
    % disp(['Output weightings: RE = diag([' ,num2str(diag(RE)) ,'])]);

    % Calculate estimator gain
    % [Kbar, P, K_poles] = lqr(Abar', Cbar', QE, RE);

    pbar = [-15+22.5i,-15-22.5i,-9+13i,-9-13i,-10,-11]

    Kbar = place(Abar',Cbar',pbar);
    Kbar = Kbar'
    Kx = Kbar(1:end-1,:);
    Kd = Kbar(end,:);

    % Control arrays used in Simulink
    Acon = Abar - Kbar*Cbar;
    Bcon = [Bbar, Kbar];

    % Disturbance gain

    Bseudoin = inv((B_c')*B_c)*(B_c')
    Gd = -Bseudoin*(Bdmod)
    % Augment Gx with disturbance gain for total Gbar
    Gbar = [Gx, Gd]

    % Control arrays used in Simulink
    Acon = Abar - Kbar*Cbar;
    Bcon = [Bbar, Kbar]
    Lbar = Abar + Bbar * Gbar - Kbar * Cbar
    eig(Lbar)
end % prompt

```

```

[num1,den]=ss2tf(Lbar,Kbar,Gbar,[0],1);

%=====
% Simulation constants for Simulink
%=====

% Averaged operating point for states
q_op = Avgxop([1:NActvDOF]);
q_op(az_statenum) = 0;
qdot_op = zeros(NActvDOF,1);
qdot_op(az_statenum) = omega_op;

% Operating point for output
theta_op = u_op(1) % collective pitch op. pt

q_op_mod = [q_op(2),q_op(3),q_op(4)]

% Other constants
extract = eye(NActvDOF); % Boolean vector identifying the azimuth DOF
extract = extract(:,az_statenum);

NumBl=2
theta_init = theta_op*ones(NumBl,1) % initial conditions for pitch actuator

%=====
% Clear temporary variables
%=====

clear nfqkeep debugflag prompt prompt2
clear i i2
clear Kfast Aact Bact Cact

```

### ***B.1.2 MATLAB Script for Control Design Described in Section 4.2.4: ("LQR\_design\_GenTorq\_3states.m"):***

```

% LQR_design.m: Control design using Linear Quadratic Regulation (LQR)
% Run a FAST linearization first (.lin file must be present)
% Torque control inputs
%
% A. Wright
% 6/07
%

disp('Performing a time-invariant LQR design');

%=====
% Initialization
%=====

clear az_prev counter AMat BMat CMat

% Read the FAST linear output file (.lin) containing state-space model data
Read_linfile

az = [0: 2*pi/NAzimStep: 2*pi]; % vector of azimuth angles
omega_op = Avgxop(az_statenum + NActvDOF); % mean operating pt rotor speed

% Initialize global variables for 'azimuth_unwind' function
global az_prev counter
az_prev = 0;
counter = 0;

%=====
% Add pitch actuator dynamics to plant model

```

```

%=====

% 3 states:
% drive-train torsional defl
% gen speed
% drive-train torsional vel

Amod = [[0,0,1
AvgAMat(3,2),AvgAMat(3,3),AvgAMat(3,4)
AvgAMat(4,2),AvgAMat(4,3),AvgAMat(4,4)]]

eig(Amod)

NStates = size(Amod,1);

Bmod = [[AvgBMat(2,1),
AvgBMat(3,1),
AvgBMat(4,1)]]

Cmod = [AvgCMat(1,2),AvgCMat(1,3),AvgCMat(1,4)]

A_c = Amod;
B_c = Bmod;
C_c = Cmod;
D_c = AvgDMat;

[nump,denp]=ss2tf(A_c,B_c,C_c,D_c)

eig(A_c)

%=====
% FSFB design
%=====

disp('__FSFB design__');

% Define state and input weightings
%Q = diag([0,0,0,0, 1e2,1e2,0,0, 0,0,1e3]);
%R = diag([1000,1]); %eye(NCtrlInpts); % input weightings
%disp(['State weightings: Q = diag([' ,num2str(diag(Q)'),''])]);
%disp(['Input weightings: R = diag([' ,num2str(diag(R)'),''])]);

% check controllability
rnk = rank(ctrb(A_c,B_c), 0); % rank with tol = 0
if rnk == size(A_c,1)
    disp(' (A_c, B_c) controllable');
else
    disp([' (A_c, B_c) uncontrollable, rank = ', num2str(rnk), ' < ',num2str(size(A_c,1))]);
    error(' Cannot continue!');
end

% LQR design using pitch and torque
R=1
Q=[.01,0,0
0,2,0
0,0,16000000]

[K,S,E]=lqr(A_c, B_c, Q, R);
E
Gx = -K;

%pnun = [-2+22.6i,-2-22.6i,-3.5+15i,-3.5-15i,-2,-10,-60]
%Gx = -place(A_c, B_c,pnun)

```

```

%=====
% State Estimator design
%=====

disp(' ');
prompt = input('State Estimator design (default: n)? ', 's');
if isempty(prompt)
    prompt = 'n';
end

if prompt == 'y'
    disp('__State Estimator design__');

    % Augment with disturbance estimate

    Abar = [A_c];
    Bbar = [B_c];

    C_c
    Cbar = [C_c];

    eig(Abar)

    Dbar = 0.

    % Check observability
    rnk = rank(observ(Abar,Cbar), 0); % rank with tol = 0
    if rnk == size(Abar,1)
        disp(' (Abar, Cbar) observable');
    else
        disp([' (Abar, Cbar) unobservable, rank = ', num2str(rnk), ' <
',num2str(size(Abar,1))]);
        error(' Cannot continue!');
    end

    % Define estimated state and measurement weightings
    % QE = diag([1,1,1,1, 1,1,1e5,1, 1,1,1, 1e5]);
    % RE = eye(size(Cbar,1));

    % disp(['State weightings: QE = diag([' ,num2str(diag(QE)'),''])]);
    % disp(['Output weightings: RE = diag([' ,num2str(diag(RE)'),''])]);

    % Calculate estimator gain
    % [Kbar, P, K_poles] = lqr(Abar', Cbar', QE, RE);

    pbar = [-15+22.5i,-15-22.5i,-11]

    Kbar = place(Abar',Cbar',pbar);
    Kbar = Kbar'

    % Control arrays used in Simulink
    Acon = Abar - Kbar*Cbar;
    Bcon = [Bbar, Kbar];

    % Disturbance gain

    Gbar = [Gx]

    % Control arrays used in Simulink
    Acon = Abar - Kbar*Cbar;
    Bcon = [Bbar, Kbar]
    Lbar = Abar + Bbar * Gbar - Kbar * Cbar
    eig(Lbar)
end % prompt

Dbar = [0 ]
[num1,den]=ss2tf(Lbar,Kbar,Gbar,Dbar,1);

%=====

```

```

% Simulation constants for Simulink
%=====

% Averaged operating point for states
q_op = Avgxop([1:NActvDOF]);
q_op(az_statenum) = 0;
qdot_op = zeros(NActvDOF,1);
qdot_op(az_statenum) = omega_op;

% Operating point for output
theta_op = u_op(1) % collective pitch op. pt

q_op_mod = [q_op(2)]

% Other constants
extract = eye(NActvDOF); % Boolean vector identifying the azimuth DOF
extract = extract(:,az_statenum);

NumBl=2
theta_init = theta_op*ones(NumBl,1) % initial conditions for pitch actuator

%=====
% Clear temporary variables
%=====

clear nfqkeep debugflag prompt prompt2
clear i i2
clear Kfast Aact Bact Cact

```

### ***B.1.3 MATLAB Script for Control Design Described in Section 4.3: ("LQR\_design\_IndepPitch&states.m"):***

```

% LQR_design.m: Control design using Linear Quadratic Regulation (LQR)
% Run a FAST linearization first (.lin file must be present)
% Independent Pitch control inputs
%
% A. Wright
% 8/07
%

disp('Performing a time-invariant LQR design');

%=====
% Initialization
%=====

clear az_prev counter AMat BMat CMat

% Read the FAST linear output file (.lin) containing state-space model data
Read_linfile

az = [0: 2*pi/NAzimStep: 2*pi]; % vector of azimuth angles
omega_op = Avgxop(az_statenum + NActvDOF); % mean operating pt rotor speed

% Initialize global variables for 'azimuth_unwind' function
global az_prev counter
az_prev = 0;
counter = 0;

%=====
% Add pitch actuator dynamics to plant model
%=====

% 5 plant states:
% blade1 1st flap disp
% blade2 1st flap disp

```

```

% gen speed
% blade1 1st flap vel
% blade2 1st flap vel

% 3 disturbance states:
% 2 states: linear shear disturbance
% 1 state: uniform wind component

Amod = [[AvgAMat(2,2),AvgAMat(2,3),AvgAMat(2,4),AvgAMat(2,5),AvgAMat(2,6)
AvgAMat(3,2),AvgAMat(3,3),AvgAMat(3,4),AvgAMat(3,5),AvgAMat(3,6)
AvgAMat(4,2),AvgAMat(4,3),AvgAMat(4,4),AvgAMat(4,5),AvgAMat(4,6)
AvgAMat(5,2),AvgAMat(5,3),AvgAMat(5,4),AvgAMat(5,5),AvgAMat(5,6)
AvgAMat(6,2),AvgAMat(6,3),AvgAMat(6,4),AvgAMat(6,5),AvgAMat(6,6)]]

NStates = size(Amod,1);

Bmod = [[AvgBMat(2,1),AvgBMat(2,2)
AvgBMat(3,1),AvgBMat(3,2)
AvgBMat(4,1),AvgBMat(4,2)
AvgBMat(5,1),AvgBMat(5,2)
AvgBMat(6,1),AvgBMat(6,2)]]

Bdmod = [[AvgBdMat(2,1), AvgBdMat(2,1)
AvgBdMat(3,1), AvgBdMat(3,1)
0., AvgBdMat(4,1)
AvgBdMat(5,1), AvgBdMat(5,1)
-AvgBdMat(6,1), AvgBdMat(6,1)]]

Cmod = [AvgCMat(1,2), AvgCMat(1,3), AvgCMat(1,4), AvgCMat(1,5), AvgCMat(1,6)
0.5*(AvgCMat(2,2)-AvgCMat(3,2)), 0.5*(AvgCMat(2,3)-AvgCMat(3,3)), 0.5*(AvgCMat(2,4)-
AvgCMat(3,4)), 0.5*(AvgCMat(2,5)-AvgCMat(3,5)), 0.5*(AvgCMat(2,6)-AvgCMat(3,6))]

A_c = Amod;
Bd_c = Bdmod;
B_c = Bmod;
C_c = Cmod;
D_c = AvgDMat;
Dd_c = AvgDdMat;

eig(A_c)

%=====
% Wind disturbance generator
%=====

Omega=4.3668

F_dac = [[0,1,0
-Omega*Omega,0,0
0,0,0]]; % for step disturbance generator
theta_dac = [[1,0,0
0,0,1]]; %
ndstates = size(F_dac,1); % number of disturbance states
ndinp = size(theta_dac,1); % number of disturbance inputs

%=====
% FSFB design
%=====

disp('__FSFB design__');

```

```

% Define state and input weightings
%Q = diag([0,0,0,0, 1e2,1e2,0,0, 0,0,1e3]);
%R = diag([1000,1]); %eye(NCtrlInpts); % input weightings
%disp(['State weightings: Q = diag(['',num2str(diag(Q)'),'])']);
%disp(['Input weightings: R = diag(['',num2str(diag(R)'),'])']);

% check controllability
rnk = rank(ctrb(A_c,B_c), 0); % rank with tol = 0
if rnk == size(A_c,1)
    disp(' (A_c, B_c) controllable');
else
    disp([' (A_c, B_c) uncontrollable, rank = ', num2str(rnk), ' < ',num2str(size(A_c,1))]);
    error(' Cannot continue!');
end

% blade1 1st flap disp
% blade2 1st flap disp
% gen speed
% blade1 1st flap vel
% blade2 1st flap vel
% LQR design using pitch and torque
R=[1,0
    0,1];
Q=[0,0,0,0,0
    0,0,0,0,0
    0,0,1,0,0
    0,0,0,.00001,0
    0,0,0,0,.00001]
[K,S,E]=lqr(A_c, B_c, Q, R);
E
Gx = -K

%pnm = [-2+22.6i,-2-22.6i,-3.5+15i,-3.5-15i,-2,-10,-60]
%Gx = -place(A_c, B_c,pnm)

%=====
% State Estimator design
%=====

disp(' ');
prompt = input('State Estimator design (default: n)? ', 's');
if isempty(prompt)
    prompt = 'n';
end

if prompt == 'y'
    disp('__State Estimator design__');

    % Augment with disturbance estimate
    A_c
    Bd_c
    theta_dac
    Bd_c*theta_dac
    zeros(ndstates,NStates)
    F_dac
    Abar = [A_c, Bd_c*theta_dac; zeros(ndstates,NStates), F_dac];
    Bbar = [B_c; zeros(ndstates,NCtrlInpts)]

    C_c
    zeros(NCtrlOuts,ndstates)
    Cbar = [C_c, zeros(NCtrlOuts-1,ndstates)]

    eig(Abar)

    Dbar = 0.

    % Check observability
    rnk = rank(obsv(Abar,Cbar), 0); % rank with tol = 0

```

```

if rnk == size(Abar,1)
disp(' (Abar, Cbar) observable');
else
disp([' (Abar, Cbar) unobservable, rank = ', num2str(rnk), ' <
',num2str(size(Abar,1))]);
error(' Cannot continue!');
end

% Define estimated state and measurement weightings
% QE = diag([1,1,1,1, 1,1,1e5,1, 1,1,1, 1e5]);
% RE = eye(size(Cbar,1));

% disp(['State weightings: QE = diag([' ,num2str(diag(QE)'),'])']);
% disp(['Output weightings: RE = diag([' ,num2str(diag(RE)'),'])']);

% Calculate estimator gain

RE=[1,0
0,.1]

QE=[0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0
0,0,.0006,0,0,0,0,0
0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0
0,0,0,0,0,1000,0,0
0,0,0,0,0,0,10000,0
0,0,0,0,0,0,0,0,10]
[Kbar, P, K_poles] = lqr(Abar', Cbar', QE, RE);

K_poles

%Kbar = place(Abar',Cbar',pbar);

Kbar = Kbar'
Kx = Kbar(1:end-1,:);
Kd = Kbar(end,:);

% Control arrays used in Simulink
Acon = Abar - Kbar*Cbar;
Bcon = [Bbar, Kbar];

% Disturbance gain
B_c
B_pseudoin = inv((B_c')*B_c)*(B_c')
%Gd = [.0,0,.013
% -.0,0,.013]
Gd = [.01,0,.013
.01,0,.013]
% Augment Gx with disturbance gain for total Gbar
Gbar = [Gx, Gd]

% Control arrays used in Simulink
Acon = Abar - Kbar*Cbar;
Bcon = [Bbar, Kbar]
Lbar = Abar + Bbar * Gbar - Kbar * Cbar
eig(Lbar)
end % prompt

Dbar = [0,0
0,0]
[num1,den]=ss2tf(Lbar,Kbar,Gbar,Dbar,1);
[num2,den]=ss2tf(Lbar,Kbar,Gbar,Dbar,2);

%=====
% Simulation constants for Simulink
%=====

```

```

% Averaged operating point for states
q_op = Avgxop([1:NActvDOF]);
q_op(az_statenum) = 0;
qdot_op = zeros(NActvDOF,1);
qdot_op(az_statenum) = omega_op;

% Operating point for output
theta_op = u_op(1) % collective pitch op. pt

q_op_mod = [q_op(1),q_op(2),q_op(3)]

% Other constants
extract = eye(NActvDOF); % Boolean vector identifying the azimuth DOF
extract = extract(:,az_statenum);

NumBl=2
theta_init = theta_op*ones(NumBl,1) % initial conditions for pitch actuator

%=====
% Clear temporary variables
%=====

clear nfgkeep debugflag prompt prompt2
clear i i2
clear Kfast Aact Bact Cact

```

## Appendix C. Simulating Control with Fortran Subroutines Linked with FAST

### C.1 Description

Another way to simulate the closed-loop system with FAST is by using the pitch control subroutine compiled and linked with the code. This method is useful for engineers who choose not to use the FAST-Simulink simulation capability. This method, along with an example Fortran subroutine for pitch control (to be compiled and linked with FAST) is described here.

The input PCMode in the FAST input file should be set to 1 in order to simulate control using a pitch control subroutine instead of Simulink.

Dr. A. Craig Hansen wrote a special pitch control subroutine (PitchCntrl\_ACH.f90) to link with FAST that is supplied with the FAST code (see the FAST User's Manual, page 25). To simulate this baseline controller, we have slightly modified this subroutine. The main modification has been to change the method of scheduling gains to reflect the method described in Section 3.4.2. This modified pitch control subroutine (PitchCntrl\_ACH\_mod1.f90) is listed in this appendix, starting on page C2.

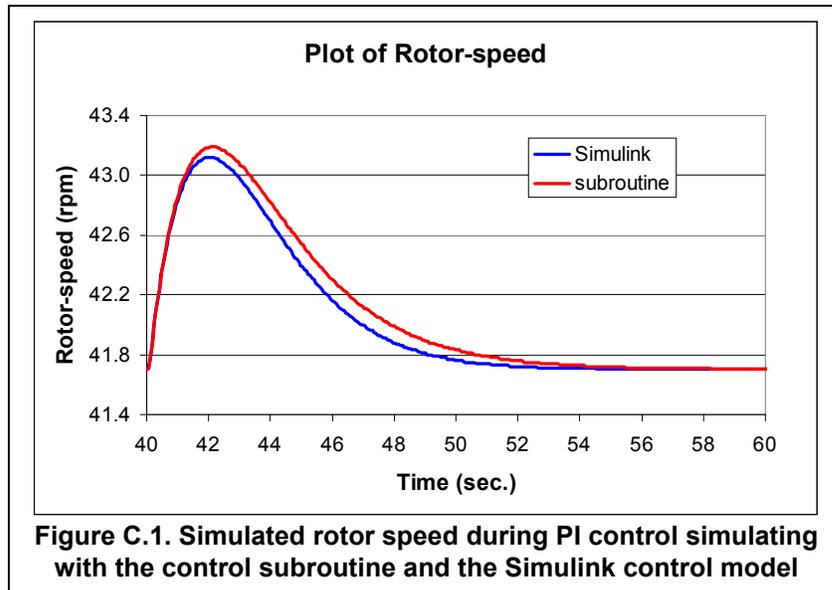
This subroutine depends on a special input file used to define various parameters needed in the pitch control subroutine (pitch.ipt). These input parameters include such items as the rotor speed set point in Region 3, integrator anti-windup gain, minimum and maximum allowable pitch angle (used to apply pitch angle limits), pitch angle at start of gain scheduling, pitch angle at end of gain scheduling, etc. This input file is also used to input the transfer functions used to define the pitch controller. It is listed starting on page C13.

In this subroutine, the input to the pitch controller is generator speed subtracted from the set point divided by the gear ratio. When simulating with only the generator DOF, this will give exactly the same result as using rotor speed as the pitch control input (we used rotor speed in the Simulink models described above). One difference between this subroutine and the Simulink models is that the generator speed input has the units rpm instead of r/s. In addition, the pitch angle output from these transfer functions has the units of degree instead of radian. In the pitch.ipt file, the first transfer function models the integrator portion of the pitch controller. The numerator coefficient reflects the integrator gain ( $K_I$ ). The next transfer function models the proportional part of the controller ( $K_p$ ). The contributions from these two transfer functions are then summed. The steady-state pitch angle obtained from a linearization is then added to this sum. Before the final commanded pitch angle is passed to FAST, it is converted to radians.

The subroutine listed in the appendix performs all of the same functions as the Simulink controller models described in Chapter 3. In this subroutine, we apply the same gain scheduling technique as described in Section 3.3.2 and anti-windup described in Section

3.3.3. Tower feedback damping can also be modeled. The transfer function for tower feedback is shown in the pitch.ipt file as transfer function #3. For this example input file, the gains for tower damping were set to zero.

We simulated control for a step change in wind speed using the pitch control subroutine. The only DOF switched on in FAST was the generator DOF. Figure C1 shows a comparison between the results using this subroutine and the results using the Simulink model of Section 3.3. There are minor differences, which may be caused by differences in methods of numerical integration, tolerances, etc. Overall the results from the two methods are in very close agreement.



## C.2 Pitch Control Subroutine and Related Files:

### C.2.1 Subroutine:

```
! NOTE: This source file contains an example PitchCntrl() user-specified
! routine for computing blade pitch control commands based on transfer
! function descriptions specified in a pitch.ipt input file. This
! routine was written by Craig Hansen (ACH) of Windward Engineering
! LLC. Questions related to the use of this routine should be
! addressed to Craig Hansen.
```

```
=====
SUBROUTINE PitchCntrl ( BIPitch, ElecPwr, HSS_Spd, GBRatio, TwrAccel, NB, ZTime, DT, DirRoot, TFOOutput )
```

```
! This routine reads a data file containing user specified transfer
! function information to allow the use of control systems in FAST
! and ADAMS. The parameters read in, and array dimensions may be
! adjusted to suit the users need.
! The transfer function coefficients are read in, then converted to
! state space form for integration using a fourth order Runge-Kutta
! integration scheme.
! This routine was originally written by C. Hansen in Fortran 77 for
! use with FAST and ADAMS. It was converted to Modern Fortran by
! J. Jonkman.
```

USE Precision

IMPLICIT NONE

! Passed variables:

INTEGER(4), INTENT(IN) :: NB ! Number of blades.  
REAL(ReKi), INTENT(IN) :: BIPitch (NB) ! Current values of the blade pitch angles (rad)  
REAL(ReKi), INTENT(IN) :: DT ! Integration time step (sec)  
REAL(ReKi), INTENT(IN) :: ElecPwr ! Electrical power (watts)  
REAL(ReKi), INTENT(IN) :: GBRatio ! Gearbox ratio (-)  
REAL(ReKi), INTENT(IN) :: HSS\_Spd ! HSS speed (rad/s)  
REAL(ReKi), INTENT(OUT) :: TFOutput(NB) ! Desired pitch angles returned by this subroutine (rad)  
REAL(ReKi), INTENT(IN) :: TwrAccel ! Tower top acceleration (m/s^2)  
REAL(ReKi), INTENT(IN) :: ZTime ! Current simulation time (sec)

CHARACTER(1024), INTENT(IN) :: DirRoot ! The name of the root file including the full path to the current working directory. This may be useful if you want this routine to write a permanent record of what it does to be stored with the simulation results: the results should be stored in a file whose name (including path) is generated by appending any suitable extension to DirRoot.

! Local Variables:

! NOTE: If the values of MSZ OR NSZ are changed, the PARAMETER  
! statements in TFSISO must also be changed to match.

INTEGER(4), PARAMETER :: MSZ = 12 ! Larger than highest order of transfer function; also used to  
size number of constants CNST  
INTEGER(4), PARAMETER :: NSZ = 4 ! Number of transfer functions we will use

REAL(ReKi) :: A0  
REAL(ReKi), SAVE :: AC (MSZ,NSZ)  
REAL(ReKi), SAVE :: BC (0:MSZ,NSZ)  
REAL(ReKi), SAVE :: CNST (MSZ) ! Maximum of MSZ constants  
REAL(ReKi) :: TFInput ! Input to the transfer function  
REAL(ReKi), SAVE :: TPCOn ! Time to enable active pitch control.  
REAL(ReKi), PARAMETER :: RPS2RPM = 9.5492966 ! Factor to convert radians per second to revolutions  
per minute.  
REAL(ReKi) :: SUM

INTEGER(4), SAVE :: CntrlRgn ! Control region (CntrlRgn = 2 = power control, CntrlRgn = 3 =  
speed control)  
INTEGER(4) :: I  
INTEGER(4) :: J  
INTEGER(4) :: NCNST  
INTEGER(4), SAVE :: NORDER (NSZ)  
INTEGER(4) :: NR  
INTEGER(4) :: NTEMP

LOGICAL(1), SAVE :: INITFLAG = .TRUE.

CHARACTER(80) :: DESCRIP  
CHARACTER(3) :: FmtText = '(A)' ! Format for outputting pure text.

IF ( INITFLAG ) THEN

! Save the value of time in which pitch control is first activated:

TPCOn = ZTime

! Read control parameters from 'pitch.ipt' if control is employed

```

CALL OpenInFile ( 86, 'pitch.ipt' )

READ(86,FmtText) DESCRIP

WRITE(*,*) ''
WRITE(*,*) '*****'
WRITE(*,*) 'Running with control option using data from!'
WRITE(*,FmtText) DESCRIP
WRITE(*,*) '*****'
WRITE(*,*) ''

READ(86,*) CntrlRgn

READ(86,*) NCNST

DO I = 1,NCNST
  READ(86,*,END=20) CNST(I)
ENDDO

NR = 1

10 READ(86,*,END=40) NTEMP ! Use NTEMP to avoid array overflow at end of file
  NORDER(NR) = NTEMP
  READ(86,*,END=30) (BC(I,NR), I = NORDER(NR),0,-1)
  READ(86,*,END=30) (AC(I,NR), I = NORDER(NR),1,-1), A0

  IF ( ABS(A0) > 1.E-7 ) THEN
    DO I = 0,NORDER(NR)
      BC(I,NR) = BC(I,NR)/A0
    ENDDO
    DO I = 1,NORDER(NR)
      AC(I,NR) = AC(I,NR)/A0
    ENDDO
  ELSE
    CALL Abort ( 'Coefficient of largest power of s in the denominator must not be zero.' )
  ENDIF

! Calculate coefficients for state space model

DO I = 1,NORDER(NR)
  SUM = 0.0
  DO J = 1,I
    SUM = SUM - AC(J,NR)*BC(I-J,NR)
  ENDDO
  BC(I,NR) = BC(I,NR) + SUM
ENDDO

NR = NR + 1

GOTO 10

20 WRITE(*,*) 'Error while reading constants from pitch.ipt file'
  WRITE(*,*) 'Encountered end of file while reading constants'
  WRITE(*,*) 'Number of constants expected:', NCNST
  WRITE(*,*) 'Number of constants found: ', I-1
  CALL Abort( 'Check your pitch.ipt file.' )

30 WRITE(*,*) 'Error in specification of transfer function #',NR
  CALL Abort( 'Check your pitch.ipt file.' )

```

40 CONTINUE

! Always require that NSZ transfer functions are input to help  
! ensure correct input file for this version of code

```
IF( NR-1 /= NSZ ) THEN
  WRITE(*,*) 'Error in pitch.ipt file'
  WRITE(*,*) 'Incorrect number of transfer functions '
  WRITE(*,*) 'Number that were read:      ', NR-1
  WRITE(*,*) 'Number that were expected: ', NSZ
  CALL Abort( 'Check your pitch.ipt file.' )
ENDIF
```

INITFLAG = .FALSE.

CLOSE(86)

RETURN

ENDIF ! Initialization

! Determine the Transfer Function Input, TFInput, based on specified control  
! region:

SELECT CASE ( CntrlRgn ) ! Which control region are we in?

CASE ( 2 ) ! Region 2 control = power control

TFInput = 0.001\*ElecPwr ! Electric power, kW

CASE ( 3 ) ! Region 3 control = speed control

TFInput = HSS\_Spd/GBRatio\*RPS2RPM ! LSS speed at gearbox entrance, rpm

CASE DEFAULT ! None of the above

CALL Abort( ' CntrlRgn must be 2 or 3.' )

ENDSELECT

! Use the control routine

CALL CTRL4 ( CNST, AC, BC, NORDER, MSZ, NSZ, &  
TFOutput, BiPitch, TFInput, TwrAccel, NB, ZTime, TPCOn )

RETURN

END SUBROUTINE PitchCntrl

---

SUBROUTINE CTRL4 ( CNST, AC, BC, NORDER, MSZ, NSZ, &  
TFOutput, BiPitch, TFInput, TwrAccel, NB, ZTime, TPCOn )

! In this subroutine dependencies between transfer functions, as well  
! as inputs and outputs of transfer functions are defined. For the  
! call to TFSISO the user need only be concerned with the first  
! three arguments. The first argument is the transfer function  
! input, the second is the transfer function output, the third  
! specifies the transfer function number, with the number  
! corresponding to the order in which the transfer functions were

! read from the input file.

USE Constant  
USE Precision  
USE Output

IMPLICIT NONE

! Passed variables:

INTEGER(4), INTENT(IN) :: MSZ  
INTEGER(4), INTENT(IN) :: NB ! Number of blades.  
INTEGER(4), INTENT(IN) :: NSZ  
  
REAL(ReKi), INTENT(IN) :: AC (MSZ,NSZ)  
REAL(ReKi), INTENT(IN) :: BC (0:MSZ,NSZ)  
REAL(ReKi), INTENT(IN) :: BIPitch (NB) ! Current blade pitch.  
REAL(ReKi), INTENT(IN) :: CNST (MSZ)  
REAL(ReKi), INTENT(IN) :: TFInput  
REAL(ReKi), INTENT(OUT) :: TFOutput(NB)  
REAL(ReKi), INTENT(IN) :: TPCOn ! Time to enable active pitch control.  
REAL(ReKi), INTENT(IN) :: TwrAccel  
REAL(ReKi), INTENT(IN) :: ZTime ! Current simulation time.  
  
INTEGER(4), INTENT(IN) :: NORDER (NSZ)

! Local variables:

REAL(ReKi) :: AWIND = 0.0  
REAL(ReKi) :: DTSTRT  
REAL(ReKi) :: DTCNTRL  
REAL(ReKi) :: GAINSCHEM  
REAL(ReKi) :: GSCoef  
REAL(ReKi) :: GSExp  
REAL(ReKi) :: GSPit1  
REAL(ReKi) :: GSPit2  
REAL(ReKi) :: OLDTIME = 0.0 ! Previous time we changed the pitch angle, sec  
REAL(ReKi) :: OLDTFOUTPUT  
REAL(ReKi) :: PHI0  
REAL(ReKi) :: PHI1  
REAL(ReKi) :: PHI2  
REAL(ReKi) :: PITMAX  
REAL(ReKi) :: PITMIN

! Mod made by A. Wright on 6/5/07:

REAL(ReKi) :: TWRFAAccel  
!  
REAL(ReKi) :: TWROUTPUT  
REAL(ReKi) :: U1  
REAL(ReKi) :: U2  
REAL(ReKi) :: X

INTEGER(4) :: DEBUGFLAG  
INTEGER(4) :: K ! Blade number

LOGICAL(1) :: TRIMFLAG = .TRUE. ! Initialization flag

CHARACTER( 8) :: Frmt1 = '(20(:A))'  
CHARACTER(14) :: Frmt2 = '(20(:G12.5,A))'

! Global functions:

REAL(ReKi), EXTERNAL :: SAT2

```

SAVE ! mlb - Do we need to save everything?

! Enter variables which need to be initialized after trim solution here

IF ( TRIMFLAG ) THEN

PHI0    = BIPitch(1)*RtoD ! Initial pitch angle (deg)
OLDTFOUTPUT = BIPitch(1)
TRIMFLAG = .FALSE.

! Assign variable values from the pitch.ipt file

PITMIN   = CNST( 4) ! Minimum pitch angle, deg
PITMAX   = CNST( 5) ! Maximum pitch angle, deg
DTCNTRL  = CNST( 6) ! Time interval for pitch control, sec
GSPit1   = CNST( 7) ! Pitch angle for start of gain scheduling (>0), rad
GSPit2   = CNST( 8) ! Pitch angle for end of gain scheduling, rad
GSCcoef  = CNST( 9) ! constant 'a' in gain schedule power law ( GS = a * x**p)
GSExp    = CNST(10) ! exponent 'p' in gain schedule power law ( GS = a * x**p)
DEBUGFLAG = CNST(11) ! Debug file output (0=no, 1=yes)

! Open file to receive control variable output for debug (if desired)

IF( DEBUGFLAG == 1 ) THEN

CALL OpenOutFile ( 40, 'pitcntrl.plt' )

WRITE(40,*) 'Output of PITCH control control subroutine'
WRITE(40,*) 'Gain schedule coeffs = ', GSCcoef, GSExp

!  WRITE(40,Frmt1)  &
!  'Time',      TAB, &
!  'Pitch',    TAB, &
!  'TFInput',  TAB, &
!  'TwrAccel', TAB, &
!  'TWROUTPUT', TAB, &
!  'Input1UI', TAB, &
!  'OutputPHI0', TAB, &
!  'OutputPHI1', TAB, &
!  'OutputPHI2', TAB, &
!  'PitchOutput', TAB, &
!  'GainSched', TAB, &
!  'Awindup'

ENDIF

ENDIF

IF( ZTime - OLDTIME < DTCNTRL ) THEN ! Time check needed for FAST
DO K = 1,NB
    TFOutput(K) = OLDTFOUTPUT
ENDDO ! K
RETURN
ENDIF

OLDTIME = ZTime

! Apply gain scheduling based on measured pitch angle (rad)

X = SAT2( BIPitch(1), GSPit1, GSPit2 ) ! Use endpoints if outside the endpoints
GAINSCHEDED = 1./( 1.+ X/GSCcoef )

! Ramp the gains up from zero to the desired values during startup
! The intent is to avoid controller problems due to startup transients

```

```

DTSTRT = ZTime - TPCOn + 0.01 ! Avoid zero gain by using offset
IF( DTSTRT < 5.0 ) GAINSCHEDED = GAINSCHEDED * DTSTRT / 5.0

! First TF input is error * gain

U1 = GAINSCHEDED * CNST(1) * ( CNST(2) - TFInput )
U2 = GAINSCHEDED * CNST(3) * AWIND

! U1 = Input to transfer function (rotor RPM or kW error, depending upon PCHMODE)
! PHI0 = Output of transfer function = pitch in degrees
! Third argument = number or ID of transfer function
! First transfer function is integral term of PID pitch demand control

CALL TFSISO( U1-U2, PHI0, 1, AC, BC, DTCNTRL, NORDER, MSZ, NSZ )

! Second transfer function is PD terms of PID pitch demand control
! NOTE: The antiwindup term is not included here

CALL TFSISO( PHI0 , PHI1, 2, AC, BC, DTCNTRL, NORDER, MSZ, NSZ )

! Add these two to get pitch demand control output

!PHI1 = PHI0 + PHI1

! Now apply third transfer function to represent tower acceleration
! feedback. If order of 3rd transfer function is zero, this
! transfer function is not applied.

IF ( NORDER(3) == 0 ) THEN
  TWROUTPUT = 0.0
ELSE
! Mod made by A. Wright on 6/5/07: change to TWRFAAccel
  CALL TFSISO(TWRFAAccel, TWROUTPUT, 3, AC, BC, DTCNTRL, NORDER, MSZ, NSZ )
ENDIF

! Add pitch demand from tower accel. to that from speed/power error
! Add equilibrium pitch value

PHI1 = PHI1 + TWROUTPUT + 11.

! SAT2 function ensures that the pitch angle demand does not go
! beyond limits (2nd and 3rd arguments in degrees)

PHI2 = SAT2( PHI1, PITMIN, PITMAX ) ! Pitch angle, deg

AWIND = PHI1 - PHI2 ! Anti windup term when pitch demand saturates, deg

! Now apply fourth transfer function to represent the actuator
! (Do not use this actuator in ADAMS. If order of 4th
! transfer function is zero, this transfer function is not applied)

IF ( NORDER(4) == 0 ) THEN
  DO K=1,NB
    TFOutput(K) = PHI2
  ENDDO ! K
ELSE
  CALL TFSISO( PHI2, TFOutput(1), 4, AC, BC, DTCNTRL, NORDER, MSZ, NSZ )
  TFOutput = TFOutput(1) ! All blades use same pitch in this version
ENDIF

DO K=1,NB
  TFOutput(K) = TFOutput(K)*Dtor ! Pitch angle returned by subroutine, rad

```

ENDDO ! K

OLDTFOUTPUT = TFOutput(1) ! Save for use until next control time

! Write to controller output file if desired.

```
!IF ( DEBUGFLAG == 1 ) THEN
! WRITE(40,Frmt2) &
! ZTime, TAB, &
! BIPitch(1)*RtoD, TAB, &
! TFInput, TAB, &
! TwrAccel, TAB, &
! TWROUTPUT, TAB, &
! U1, TAB, &
! PHI0, TAB, &
! PHI1, TAB, &
! PHI2, TAB, &
! TFOutput(1)*RtoD, TAB, &
! GAINSCHEM, TAB, &
! AWINDD
!ENDIF
```

RETURN  
END SUBROUTINE CTRL4

=====

SUBROUTINE TFSISO ( U, Y, NR, AC, BC, DT, NORDER, MSZ, NSZ )

! This routine integrates the transfer functions using a fourth order  
! Runge-Kutta method.

USE Precision

IMPLICIT NONE

! Passed variables:

```
INTEGER(4), INTENT(IN) :: MSZ
INTEGER(4), INTENT(IN) :: NSZ

REAL(ReKi), INTENT(IN) :: AC (MSZ,NSZ)
REAL(ReKi), INTENT(IN) :: BC (0:MSZ,NSZ)
REAL(ReKi), INTENT(IN) :: DT
REAL(ReKi), INTENT(IN) :: U
REAL(ReKi), INTENT(OUT) :: Y

INTEGER(4), INTENT(IN) :: NORDER (NSZ)
INTEGER(4), INTENT(IN) :: NR
```

! Local variables:

! NOTE: If the values of M OR N are changed, the PARAMETER  
! statements in PitchCntrl must also be changed to match.

```
INTEGER(4), PARAMETER :: M = 12 ! Larger than highest order of transfer function; also used to size
number of constants CNST
INTEGER(4), PARAMETER :: N = 4 ! Number of transfer functions we will use
```

```
REAL(ReKi) :: DT6
REAL(ReKi) :: DXDT (MSZ)
REAL(ReKi) :: DXM (MSZ)
REAL(ReKi) :: DXT (MSZ)
REAL(ReKi) :: HDT
REAL(ReKi) :: X (M,N) = 0.0
```

```

REAL(ReKi)      :: XT (MSZ)
INTEGER(4)      :: I
LOGICAL(1)      :: INITFLAG(N) = .TRUE.

IF ( INITFLAG(NR) ) THEN
  CALL TFINIT( U, Y, X, AC, BC, NORDER, NSZ, MSZ, NR )
  IF( ( NSZ /= N ) .OR. ( MSZ /= M ) ) THEN
    PRINT*, 'ERROR IN PARAMETERS M AND/OR N IN TFSISO'
  ENDIF
  INITFLAG(NR) = .FALSE.
ENDIF

HDT = 0.5*DT
DT6 = DT/6.0

DO I = 1,NORDER(NR)
  XT(I) = X(I,NR)
ENDDO

CALL XDOT( U, XT, AC, BC, DXDT, NORDER, NSZ, MSZ, NR )
DO I = 1,NORDER(NR)
  XT(I) = X(I,NR) + HDT*DXDT(I)
ENDDO

CALL XDOT( U, XT, AC, BC, DXT, NORDER, NSZ, MSZ, NR )
DO I = 1,NORDER(NR)
  XT(I) = X(I,NR) + HDT*DXT (I)
ENDDO

CALL XDOT( U, XT, AC, BC, DXM, NORDER, NSZ, MSZ, NR )
DO I = 1,NORDER(NR)
  XT(I) = X(I,NR) + DT*DXM (I)
  DXM(I) = DXT(I) + DXM(I)
ENDDO

CALL XDOT( U, XT, AC, BC, DXT, NORDER, NSZ, MSZ, NR )
DO I = 1,NORDER(NR)
  X(I,NR) = X(I,NR) + DT6*( DXDT(I) + DXT(I) + 2.0*DXM(I) )
ENDDO

Y = X(1,NR) + BC(0,NR)*U

RETURN
END SUBROUTINE TFSISO
=====
SUBROUTINE XDOT ( U, X, AC, BC, DXDT, NORDER, NSZ, MSZ, NR )

! This routine calculates derivatives for fourth order Runge-Kutta.

USE Precision

IMPLICIT NONE

```

```

! Passed variables:

INTEGER(4), INTENT(IN ) :: MSZ
INTEGER(4), INTENT(IN ) :: NSZ

REAL(ReKi), INTENT(IN ) :: AC (MSZ,NSZ)
REAL(ReKi), INTENT(IN ) :: BC (0:MSZ,NSZ)
REAL(ReKi), INTENT(OUT) :: DXDT (MSZ)
REAL(ReKi), INTENT(IN ) :: U
REAL(ReKi), INTENT(IN ) :: X (MSZ)

INTEGER(4), INTENT(IN ) :: NORDER (NSZ)
INTEGER(4), INTENT(IN ) :: NR

! Local variables:

REAL(ReKi)          :: SUM

INTEGER(4)          :: I

! Derivatives:

DO I = 1,NORDER(NR)-1
  DXDT(I) = X(I+1) + BC(I,NR)*U
ENDDO

SUM = 0.0
DO I = 1,NORDER(NR)
  SUM = SUM - X(I)*AC( NORDER(NR) + 1 - I, NR )
ENDDO

DXDT(NORDER(NR)) = SUM + BC(NORDER(NR),NR)*U

RETURN
END SUBROUTINE XDOT
=====
SUBROUTINE TFINIT ( U, Y, X, AC, BC, NORDER, NSZ, MSZ, NR )

! This routine initialize states for fourth order Runge-Kutta.

USE          Precision

IMPLICIT     NONE

! Passed variables:

INTEGER(4), INTENT(IN ) :: MSZ
INTEGER(4), INTENT(IN ) :: NSZ

REAL(ReKi), INTENT(IN ) :: AC (MSZ,NSZ)
REAL(ReKi), INTENT(IN ) :: BC (0:MSZ,NSZ)
REAL(ReKi), INTENT(IN ) :: U
REAL(ReKi), INTENT(OUT) :: X (MSZ,NSZ)
REAL(ReKi), INTENT(IN ) :: Y

INTEGER(4), INTENT(IN ) :: NORDER (NSZ)
INTEGER(4), INTENT(IN ) :: NR

```

```

! Local variables:
REAL(ReKi)      :: SUM
INTEGER(4)      :: I

X(1,NR) = Y - BC(0,NR)*U
DO I = 1,NORDER(NR)-1
  X(I+1,NR) = -BC(I,NR)*U
ENDDO

SUM = 0.0
DO I = 1,NORDER(NR)-1
  SUM = SUM - X(I,NR)*AC( NORDER(NR) + 1 - I, NR )
ENDDO

! Watch out for zero values of AC
IF( AC(1,NR) /= 0.0 ) THEN
  X(NORDER(NR),NR) = ( -SUM - BC(NORDER(NR),NR)*U )/AC(1,NR)
ELSE
  X(NORDER(NR),NR) = ( -SUM - BC(NORDER(NR),NR)*U )/0.001
ENDIF

RETURN
END SUBROUTINE TFINIT
!=====
FUNCTION SAT2 ( X, XMIN, XMAX )

! Saturation function.

USE          Precision

IMPLICIT    NONE

! Passed variables:
REAL(ReKi)  :: SAT2
REAL(ReKi), INTENT(IN )  :: X
REAL(ReKi), INTENT(IN )  :: XMAX
REAL(ReKi), INTENT(IN )  :: XMIN

IF ( X > XMAX ) THEN
  SAT2 = XMAX
ELSEIF ( X < XMIN ) THEN
  SAT2 = XMIN
ELSE
  SAT2 = X
ENDIF

RETURN
END FUNCTION SAT2
!=====
FUNCTION DEADBAND ( X, XMIN, XMAX )

```

! Deadband function.

USE Precision

IMPLICIT NONE

! Passed variables:

```
REAL(ReKi)          :: DEADBAND
REAL(ReKi), INTENT(IN) :: X
REAL(ReKi), INTENT(IN) :: XMAX
REAL(ReKi), INTENT(IN) :: XMIN
```

```
IF ( X > XMAX ) THEN
  DEADBAND = X - XMAX
ELSEIF ( X < XMIN ) THEN
  DEADBAND = X - XMIN
ELSE
  DEADBAND = 0.0
ENDIF
```

```
RETURN
END FUNCTION DEADBAND
```

!=====

### **C.2.2 Pitch.ipt file:**

Speed controller for CART turbine.

```
3   CntrlRgn - Control region {2 = power control, 3 = speed control}
11  NCSNT   - Number of constants used in controls
-1.  CNST(1) - Gain on TF 1
41.7 CNST(2) - Rotor speed set point, RPM
60.  CNST(3) - Integrator anti-windup gain
-1   CNST(4) - Minimum pitch angle, deg
90.  CNST(5) - Maximum pitch angle, deg
0.006 CNST(6) - Time interval for pitch control, sec ( $\geq$  DT)
.0052 CNST(7) - Pitch angle at start of gain scheduling, (must be  $> 0$ ) rad
.4363 CNST(8) - Pitch angle at end of gain scheduling (GSPit2>GSPit1), rad
.0454 CNST(9) - Power law gain sched. coefficient
-0.000 CNST(10) - Power law gain sched. exponent
0     CNST(11) - Write pitcntrl.plt file? 0=no, 1=yes

1     Order of tf #1 (highest power of s), input=Rotor speed (RPM), output=Integral Pitch demand (deg)
4.6794 0.000 Numerator coefficients (Order+1 values, ascending powers of s)
0.0 1.0 Denominator coefficients (Order+1 values, ascending powers of s)

1     Order of tf #2 (highest power of s), input=Rotor speed (RPM), output=PD Pitch demand (deg)
0 14.089 Numerator coefficients (Order+1 values, ascending powers of s)
0.0 1.0 Denominator coefficients (Order+1 values, ascending powers of s)

1     Order of tf #3, input=Tower accel, output=pitch (deg) (Use 0 to not include tower feedback cntrl)
0. 0. Numerator coefficients (Order+1 values, ascending powers of s)
0.0 1.0 Denominator coefficients (Order+1 values, ascending powers of s)

0     Order of tf #4, input=Pitch demand(deg), output=pitch (deg) (Use 0 for ADAMS to not include actuator)
85.696 0.0 0.0 Numerator coefficients (Order+1 values, ascending powers of s)
85.696 14.812 1.0 Denominator coefficients (Order+1 values, ascending powers of s)
```

# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Executive Services and Communications Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

**PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.**

<b>1. REPORT DATE (DD-MM-YYYY)</b> March 2008		<b>2. REPORT TYPE</b> Technical report		<b>3. DATES COVERED (From - To)</b>	
<b>4. TITLE AND SUBTITLE</b> Advanced Control Design for Wind Turbines; Part I: Control Design, Implementation, and Initial Tests				<b>5a. CONTRACT NUMBER</b> DE-AC36-99-GO10337	
				<b>5b. GRANT NUMBER</b>	
				<b>5c. PROGRAM ELEMENT NUMBER</b>	
<b>6. AUTHOR(S)</b> A.D. Wright and L.J. Fingersh				<b>5d. PROJECT NUMBER</b> NREL/TP-500-42437	
				<b>5e. TASK NUMBER</b> WER8.2101	
				<b>5f. WORK UNIT NUMBER</b>	
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> National Renewable Energy Laboratory 1617 Cole Blvd. Golden, CO 80401-3393				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b> NREL/TP-500-42437	
<b>9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b>				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b> NREL	
				<b>11. SPONSORING/MONITORING AGENCY REPORT NUMBER</b>	
<b>12. DISTRIBUTION AVAILABILITY STATEMENT</b> National Technical Information Service U.S. Department of Commerce 5285 Port Royal Road Springfield, VA 22161					
<b>13. SUPPLEMENTARY NOTES</b>					
<b>14. ABSTRACT (Maximum 200 Words)</b> The purpose of this report is to give wind turbine engineers information and examples of the design, testing through simulation, field implementation, and field testing of advanced wind turbine controls. This report will be Part I in a two-part series of reports that detail advanced control design, implementation, and test results. Part I (this report) will highlight the control development process, from forming control objectives, to designing the controller, to testing the controller through analytical simulation, to field implementation and initial field testing.					
<b>15. SUBJECT TERMS</b> wind turbine control systems; multi-input multi-ouput multivariable control design methods; MIMO; wind energy					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b> UL	<b>18. NUMBER OF PAGES</b>	<b>19a. NAME OF RESPONSIBLE PERSON</b>
<b>a. REPORT</b> Unclassified	<b>b. ABSTRACT</b> Unclassified	<b>c. THIS PAGE</b> Unclassified			<b>19b. TELEPHONE NUMBER (Include area code)</b>

Standard Form 298 (Rev. 8/98)  
Prescribed by ANSI Std. Z39.18