MODIFIED SIMULTANEOUS PERTURBATION STOCHASTIC APPROXIMATION METHOD FOR POWER CAPTURE MAXIMIZATION OF WIND TURBINES

by

YANG WANG

B.A., University of Science and Technology of China, 2004

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical and Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2013

Approved by:

Warren N. White
Major Professor

Abstract

As traditional resources are becoming scarce, renewable energy is a recent topic receiving greater concern. Among the renewable energies, wind power is a very popular type of energy extracted from wind which is readily available in the environment. The use of wind power all over the world is receiving increased attention. Horizontal axis wind turbines are the most popular equipment for extracting power form the wind. One of the problems of using wind turbines is how to maximize the wind power capture. In this paper, a method for maximizing the rotor power coefficient of a wind turbine is proposed.

Simultaneous Perturbation Stochastic Approximation (SPSA) is an efficient way for extremum seeking. It is different from the classical gradient based extremum seeking algorithms. For maximizing the rotor power coefficient, it only needs two objective function measurements to take a step toward the next extremum approximation. The one measurement SPSA is a modification of SPSA method developed in this work. Instead of using measurements of two positions occurring at random directions away from the current position, it uses the measurement of one position in a random direction and the measurement of the current position to estimate the gradient.

Usually, the rotor power coefficient is not easily measurable. For speed regulation, a nonlinear robust speed controller is used in this work. The controller produces an estimate of the aerodynamic torque of wind turbine. The quality of this estimate improves with time. From that, a good estimate of power coefficient can be obtained.

Simulations in MATLAB are executed with a model of a wind turbine based on its dynamic equations. From simulations, it can be seen that the one measurement SPSA method works very well for the wind turbine. It changes the tip speed ratio and blade pitch simultaneously, and the power coefficient reaches its maximum value quickly in a reliable manner. The power capture optimization is then implemented in FAST, a turbine simulation model created by NREL which is used to test the 5MW NREL reference turbine. From the results, it is evident that the wind turbine reaches the maximum power coefficient rapidly.

Table of contents

Table of contents iii
List of Figures iv
List of Tables vi
Acknowledgements vii
Chapter 1 - Introduction and literature review 1
Introduction of wind turbines 1
Wind energy 1
Wind turbine 4
Types of wind turbines 4
Wind Turbine locations 7
Literature review of wind turbine control 8
Chapter 2 - Wind turbine system 12
Important components 12
Dynamic model of wind turbine 15
Chapter 3 - Control method 17
Inner loop control strategy 17
SPSA method 20
Chapter 4 - Simulations in MATLAB 27
Inner loop simulation 28
Outer loop simulation 31
Simulation combining inner and outer loop 36
Chapter 5 - SPSA in FAST 41
FAST 41
Simulations using SPSA with FAST 44
Chapter 6 - Conclusion and recommendation 54
References 56
Appendix A - MATLAB codes of SPSA algorithm in Chapter 3 59
Appendix B - Simulink Embedded Code and Subsystem Blocks in Chapter 4 61
Appendix C – Simulink Embedded Code and Subsystem Blocks in Chapter 5 69

List of Figures

Figure 1.1 Greenland's melting icebergs 2
Figure 1.2 Worldwide installed wind power capacity forecast 3
Figure 1.3 On-shore wind turbines 4
Figure 1.4 HAWT and VAWT wind turbines 5
Figure 1.5 Components of a HAWT 6
Figure 1.6 Off shore wind turbines 7
Figure 1.7 Operations regions of a wind turbine 9
Figure 2.1 Signal flow of the wind turbine control system 12
Figure 2.2 3D plot of $\boldsymbol{Cp}(\boldsymbol{\lambda}, \boldsymbol{\beta})$ 14
Figure 2.3 Diagram of wind power system 15
Figure 3.1 3D plot of the path of the SPSA method 21
Figure 3.2 The plot of tip speed ratio 24
Figure 3.3 The plot of blade pitch 25
Figure 3.4 The plot of Cp 25
Figure 4.1 3D plot of $\boldsymbol{0.6e^{-0.3(x-4)^2}e^{-0.3(y-2)^2}}$ 28
Figure 4.2 Simulation diagram of inner loop 29
Figure 4.3 Plot of actual angular velocity comparing with desired one 30
Figure 4.4 Diagram of outer loop 31
Figure 4.5 Plot of tip speed ratio and blade pitch of outer loop simulation 33
Figure 4.6 Cp of outer loop simulation 33
Figure 4.7 Simulation diagram of outer loop 34
Figure 4.8 tip speed ratio 34
Figure 4.9 blade pitch 35
Figure 4.10 Cp 35
Figure 4.11 Simulation diagram with both inner loop and outer loop 37
Figure 4.12 Tip Speed Ratio 37
Figure 4.13 Blade Pitch 38
Figure 4.14 Cp 38
Figure 5.1 FAST Wind Turbine Block 42

Figure 5.2 Configuration of FAST in MATLAB 42
Figure 5.3 Dynamic performance of 5MW turbine model in FAST 43
Figure 5.4 Simulation diagram 45
Figure 5.5 Cp and Cphat 46
Figure 5.6 Rotor speed 47
Figure 5.7 Tip speed ratio 48
Figure 5.8 Rotor torque 48
Figure 5.9 Simulation diagram using classical gradient based algorithm 49
Figure 5.10 Cp and Cp estimate 50
Figure 5.11 Rotor speed 50
Figure 5.12 Tip speed ratio 51
Figure 5.13 Result of modified SPSA method with $\boldsymbol{c = 40, \gamma = 3}$ 52
Figure 5.14 Result of modified SPSA method with $\boldsymbol{c = 10, \gamma = 1.5}$ 53
Figure B.1 Block diagram of controller subsystem in Figure 4.2 61
Figure B.2 Block diagram of variables subsystem in Figure 4.2 63
Figure B.3 Block diagram of the plant subsystem in Figure 4.2 63
Figure B.4 Block diagram of plant subsystem in Figure 4.4 64
Figure B.5 Block diagram of the SPSA subsystem in Figure 4.4 64
Figure B.6 Block diagram of the Cpestimator subsystem in Figure 4.11 68
Figure C.1 Block diagram of torque controller subsystem in Figure 5.4 76
Figure C.2 Block diagram of the outer_loop_control subsystem in Figure 5.4 77

List of Tables

Table 1.1 Use of traditional resources 1
Table 3.1 Parameters of m-file of SPSA method..... 24
Table 4.1 Parameters of the dynamic model..... 27
Table 4.2 Parameters of inner loop controller 29
Table 4.3 Parameters of SPSA method for simulation with both inner loop and outer loop..... 39
Table 5.1 Parameters of the speed regulation 44
Table 5.2 Parameters of the SPSA algorithm 44

Acknowledgements

I would like to express sincere thanks to my major professor Dr. White for his great help and advice through the thesis process and all my master study in the program. It would not have been possible to complete this work without his guidance and encouragement. I learned a lot from him.

I would also like to give many thanks to my thesis committee members, Dr. Dale Schinstock and Dr. Ruth Douglas Miller for their time and help.

I am very grateful to all the faculties of Mechanical Engineering in Kansas State University as well.

My family members and friends have substantively contributed to this work. I would like to thank my wife for her support during my research and study. And I would like to thank my parents for their encouragement during my graduate education. Finally, I would like to thank my lovely daughter for making me happy every day.

Chapter 1 - Introduction and literature review

Introduction of wind turbines

In this Chapter, basic knowledge about wind power and wind turbines such as their definitions and types are introduced. Literature on wind turbine control including both torque control and blade pitch control are reviewed. Control methods which will be presented in later chapters are briefly mentioned at the end of this chapter.

Wind energy

The major energy sources today in human society are still conventional ones from which we can produce large amount of energy. However, that creates many difficulties. Firstly, as we all know, the amount of conventional energy all over the world is limited and becoming scarce after centuries of exploitation. Most of these energy supplies will be depleted in the future if the current use of these resources continues at the present rate. Some can even run out in tens of years. Table 1.1 shows the current usage of four main traditional energy resources and the years left at the current using rate. According to this table, three of these four resources, oil, gas, and uranium, will be depleted in a century.

Table 1.1 Use of traditional resources [36]

Resource	% Annual Total Energy Use	Years Left at this % Usage
Oil	35%	40 yrs
Gas	22%	60 yrs
Coal	23%	250 yrs
Uranium	7%	80 yrs

Also, due to increased demand for energy, using conventional technology to produce power causes pressing environmental issues which are harmful to human and non-human life. Most of the dangerous gases coming from burning fossil fuels end up in the atmosphere. Human beings and animals are breathing this polluted air. Also, it makes the earth global average temperature increase. As a result, icebergs are melting at the south and north poles which raises the ocean levels. Figure 1.1 is of a melting iceberg in Greenland.

Figure 1.1 Greenland's melting icebergs [40]

Comparatively, using renewable energies has many advantages. First, because renewable energy is usually produced by harnessing the natural elements in the world around us, which are continually replenished, it is unlimited. We can use it as long as the earth and sun exist. Second, it usually doesn't create any pollution to the environment or only produces very small amounts.

Wind power production is the extraction of kinetic energy from the wind. It is one of the renewable energies. As an alternative to fossil fuels, wind power is plentiful, renewable, widely distributed. It produces no greenhouse gas emissions during operation and does not require as much land as traditional energies [7]. People have been harnessing wind energy for over hundreds of years. From old Holland to farms in America, windmills have already been used to

pump water as well as grind grain. The use of wind power is increasing at an annual rate of 20%, with a worldwide installed capacity of 238,000 megawatts (MW) at the end of 2011[24][20][42],and is widely used in Europe, Asia, and the United States[39]. Figure 1.2 is from the Global Wind Energy Council (GWEC). It shows worldwide installed wind power capacity before 2010 and the forecast for the next 20 years. The conservative 'reference' curve is based on the projections in the 2009 World energy. The 'Moderate' curve takes into account all policy measures to support renewable energy either already enacted or in the planning stages around the world. The 'Advanced' curve, which is the most ambitious one, examines the best extent to which this industry can grow. From Figure 1.2, the cumulative installed wind power capacity will increase at least by around 200% before 2030.

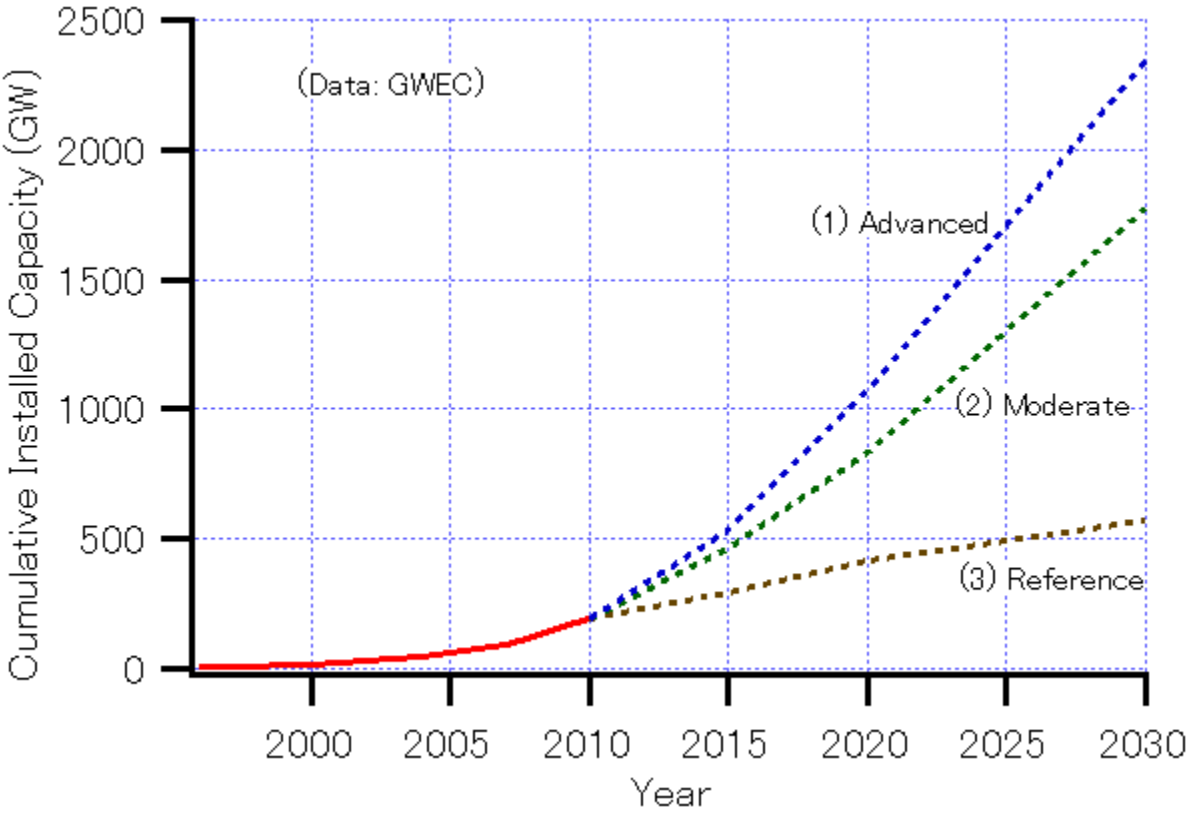


Figure 1.2 Worldwide installed wind power capacity forecast [38]

Today, the windmill's modern equivalent, a wind turbine, is used to capture wind energy to generate electric power.

Wind turbine

Wind turbines are devices which convert wind kinetic energy into electrical energy. Wind flowing over the wind turbine blades makes the wind turbine rotate. The resulting power can be used in many ways. If the power is used for producing electricity, the device is called a wind power plant. If the power is used for driving machinery such as grinding grain or pumping water, the device is called a wind mill or wind pump. If the power is used for charging batteries, it is called a wind charger. Figure 1.3 is of wind turbines.

Figure 1.3 On-shore wind turbines [10]

Types of wind turbines

Though there are various designs for wind turbines, they can be broadly classified into two categories according to the rotating axis orientation: VAWTS(Vertical Axis Wind Turbines), and HAWTS(Horizontal Axis Wind Turbines) and examples are shown in Figure 1.4.

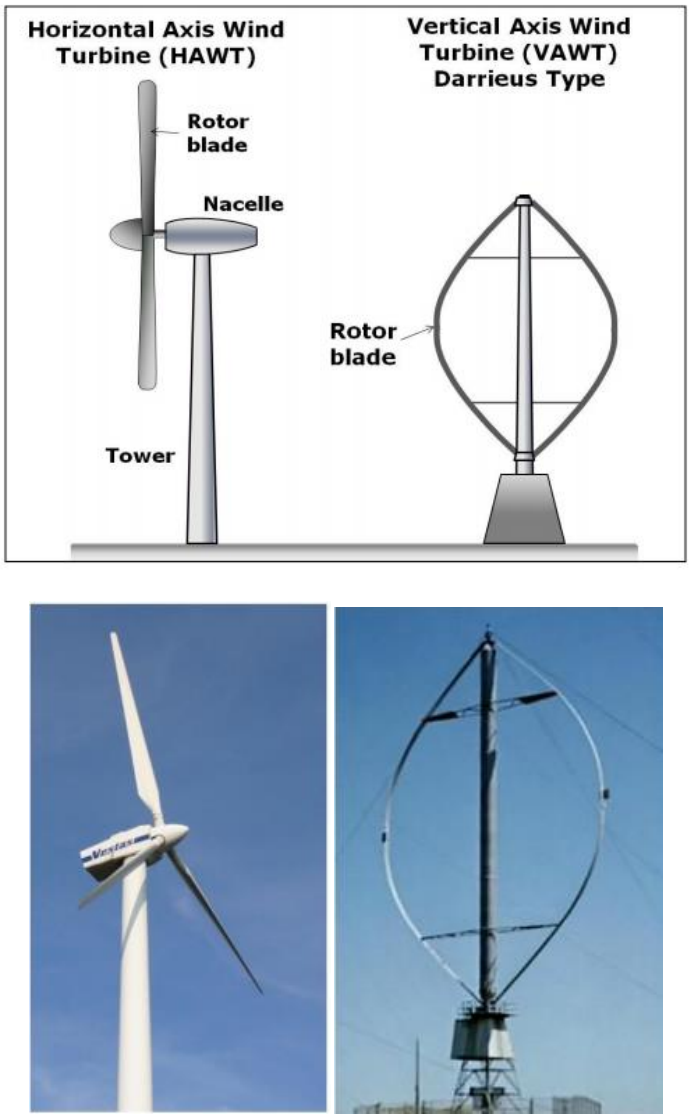


Figure 1.4 HAWT and VAWT wind turbines [44]

Most of HAWTS today feature rotors which resemble aircraft propellers that operate based on almost the same aerodynamic principles, where HAWTs use lift forces of the airflow which is perpendicular to the airflow. The wind flows over the blades generates a lifting force to turn the rotor. The nacelle on a HAWT usually contains a gearbox as well as a generator. HAWTS are usually placed on towers so as to take advantage of higher winds farther from the ground.

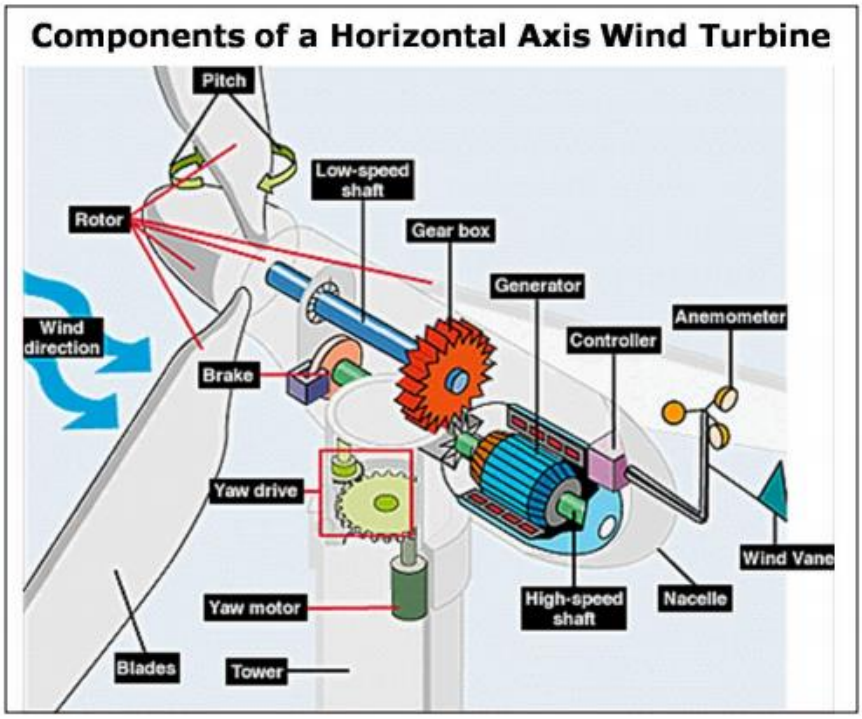


Figure 1.5 Components of a HAWT [24]

The HAWT capture area, the area blades sweep over to "capture" the wind, is given by $A= \pi(D/2)^2$, where D is diameter of the rotor. However, this capture area formula is for the rotor facing directly to the wind, which maximizes power generation. Thus HAWTS require a yawing mechanism to make sure the nacelle can rotate into the wind. The detailed components of a HAWT are shown as Figure 1.5.

Generally, for VAWTS, there are mainly two types: the Savnoius and the Darrieus. The Savonius wind turbine uses drag forces of the airflow which is in the same direction of airflow. It operates like a water wheel using drag forces, while the Darrieus uses blades which are similar to those of HAWTS. Typically VAWTS are close to the ground, which allows people to place heavy equipment such as the generator and gearbox near ground level rather than in the nacelle. However, for the same wind and same capture area, less power will be produced because winds are lower near ground level. Additionally, VAWTS don't require any yaw mechanism, because they harness wind from all directions.

Because HAWTS are the most commonly used type of wind turbines, research in this thesis is based on HAWTS. Turbines used in wind farms for commercial production of electric power are usually three-bladed and pointed into the wind by computer-controlled motors. These have high tip speeds of over 320 km/h (200 mph), high efficiency, and low torque ripple, which contribute to good reliability. The blades are usually colored white for daytime visibility by aircraft and range in length from 20 to 40 meters (66 to 130 ft) or more. The tubular steel towers range from 60 to 90 meters (200 to 300 ft) tall or more. The blades rotate at 10 to 22 revolutions per minute. At 22 rotations per minute the tip speed exceeds 90 meters per second (300 ft/s) on blades longer than 40 meters[38][43].

Wind Turbine locations

Locations of wind turbines can be divided into two categories: onshore and offshore. Onshore wind turbines refer to the wind turbines constructed on land. Offshore wind turbines refer to the wind turbines constructed in bodies of water (See in Figure 1.6). For onshore wind turbines, it is easier for installation, operation and maintenance. However, better wind speeds can be found offshore with less turbulent air. Also, building offshore wind turbines can save high valued real estate.

Figure 1.6 Off shore wind turbines [26]

Europe is the world leader in offshore wind power, with the first offshore wind farm being installed in Denmark in 1991[33]. At the end of 2011, there were 53 European offshore wind farms in waters off Belgium, Denmark, Finland, Germany, Ireland, the Netherlands, Norway, Sweden and the United Kingdom, with an operating capacity of 3,813 MW [33], while 5,603 MW is under construction [27].

Literature review of wind turbine control

Variable speed wind turbines are more popular than fixed speed turbines because of the improvement of generator technology, higher efficiency and reduced cost. Lucy Y. Pao and Kathryn E. Johnson reported the operating status of wind turbines can be divided into 3 regions [25] which are shown in Figure 1.7. In region 1, the wind speed is too low to start the turbine. The blades are with pitch angles which produce smallest aerodynamic torque. When the wind speed is large enough to start the wind turbine, the rotor is going to be accelerated and generator will start power production at some point in region 2. In regions 3, the wind speed is too large. The turbine speed will be regulated at the rated speed usually by changing the blade pitch. In these 3 regions, region 2 is the region we are most interested in, because for most of the time, wind turbines are operated in region 2. Maximization of wind power capture is only meaningful in region 2. Due to economic reasons, people want to maximize the power capture of wind turbines. Defining the rotor power coefficient as the ratio of power captured to the power available in the wind, it can be expressed as

$$\text{rotor power coefficient} = \frac{\text{power captured}}{\text{power available in wind}}. \tag{1.1}$$

For a given wind speed in region 2, the power capture coefficient will first increase as rotor speed increases. It will reach its maximum at some values of rotor speed and blade pitch. After that it will decrease as the rotor speed continues increasing. The rotor power coefficient will also change with blade pitch. Unfortunately, the values of rotor speed and blade pitch corresponding to the maximum of rotor power coefficient are in general unknown so that some control methods need to be used for wind turbines to determine those values automatically. In this paper, a two loop control strategy will be applied on wind turbines for maximizing the rotor power coefficient.

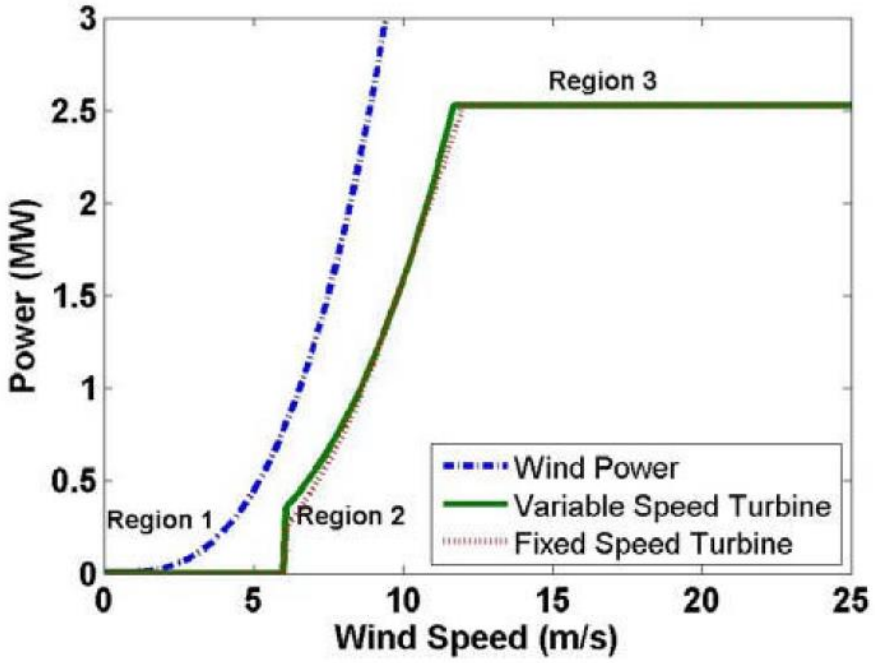


Figure 1.7 Operations regions of a wind turbine [25]

A wind turbine is such a complicated nonlinear system that a good control system is needed to assure good performance. As wind turbines become more popular, many researchers are focusing on controller design for turbines. Based on different control objectives, researchers have applied various kinds of control methods to wind turbines. Karl A. Stol (2002) designed a torque controller using Disturbance Tracking Control Theory to optimize the wind turbine power capture with wind disturbances. In Y.D. Song (2000), a nonlinear adaptive controller has been tried for speed regulation of wind turbine to achieve smooth speed tracking. Brice Beltran (2006) proposed a sliding mode controller for ensuring stability and simulated the controller on a 1.5-MW three blade wind turbine model. The results showed very good robustness against uncertainties of parameters and inaccuracies coming out from the modeling procedure.

Blade pitch control and generator torque control are two major systems for controlling wind turbines. Blade pitch control changes the aerodynamics forces by changing orientation of the blades. Blade pitch control is mainly used for limiting the aerodynamic power in region 3 in order to keep the turbine within its design limits. It is also possible to use it for optimizing energy capture in region 2. Blade pitch control using classical PID (Proportional, Integral and Derivative) has been used to regulate speed of wind turbine in A.D.Wright and L.J.Fingersh (2008) and K.Stol and M.J.Balas (2002). The rotational speed error is used as feedback to the

controller. PID gains are chosen based on different goals of turbine performance in M.M.Hand and M.J.Balas (2000). For generator torque control, the controller regulates the generator torque. For variable speed wind turbines, generator torque control is primarily used to limit the torque in region 3 and to control the rotor speed in region 2 in order to maximize energy capture. Numerous generator torque controllers have been used for wind turbines in K.E.Johnson et al. (2006) as well as a combination of both torque control and pitch control in Adel Merabet et al. (2011).

During past years, some advanced control methods have been applied to optimize the power capture of wind turbines systems by researchers. Kathryn E. Johnson (2004) used an adaptive controller to maximize the power capture of wind turbines operating in region 2 and achieved good results. But the stability of the controller had not been addressed in the paper and the adaptation period is comparatively long. K. B. Ariyur and M. Krstic (2003) used extremum seeking algorithm which relied on a dither perturbation and measurements of the objective function to maximize the captured power. Tony Hawkins et al. (2011) applied a tracking control strategy proposed by B. Xian et al. (2004) to wind turbine systems for speed regulation as well as estimation of aerodynamic torque. Also, he proposed an extreme seeking control strategy based on a Lyapunov approach. The simulation results using the Simulink Wind Blockset in F.Iov (2004) were good. The nonlinear controller is very good for speed regulation and estimating aerodynamic torque and the rotor power coefficient. However, using the Lyapunov approach, the tip speed ratio and blade pitch cannot be updated simultaneously and uncertainties in the rotor power estimate will be a problem for extreme seeking.

In the work of Hawkins et al. (2011), a gradient based steepest ascent approach was taken which was both time consuming and had problems near the peak of the C_p curve owing to perturbation size and small C_p slope. The goal here is to improve on Hawkins' technique by using an alternate approach to the outer loop, one that is more efficient and does not have perturbation size problems near the peak C_p value.

The control strategy in this paper which contains two loops can update both the tip speed ratio and blade pitch at the same time. These two loops are called in this work as inner loop and outer loop. The inner loop is the loop consisting of the wind turbine and a speed regulating controller which is responsible for tracking the desired rotor speed and estimating the aerodynamic torque. The outer loop provides the inner loop a trajectory of rotor speed and blade

pitch for the extremum seeking in the power capture. Since the power captured by a wind turbine is not reliably measurable in Z.Chen and E.Spooner (2001), the inner loop control strategy presented here is also based on the nonlinear controller, used by Hawkins et al. (2011), for tracking because it can provide an estimate of the aerodynamic torque which is proportional to the rotor power while regulating the rotor speed to the desired value. Using the estimated rotor power coefficient as the objective function, the process of Simultaneous Perturbation Stochastic Approximation (SPSA) will be used as the strategy for the outer loop to achieve extreme seeking. By combining both the inner loop and outer loop together, the whole system is driven to the configuration when the tip speed ratio and blade pitch correspond to the maximum rotor power coefficient.

In Chapter 2, the wind turbine system and the dynamic model of the wind turbine are introduced. Chapter 3 is about the speed regulation and extreme seeking algorithms. Chapter 4 shows the results of simulations using the algorithms applied to the dynamic model of Chapter 2. In Chapter 5, the use of FAST, NREL's (National Renewable Energy Laboratory) code for wind turbine modeling and simulation, is presented. System and simulations using it are presented. The controllers presented in Chapter 3 are included with the FAST simulations. The results in both Chapter 4 and Chapter 5 show that the control strategy works very well. In Chapter 4, inner loop, outer loop and combination of them are all applied to the dynamic model to test the performance. The control strategy is applied to the model of a 5MW turbine using the FAST software. Finally, conclusions are drawn in Chapter 6 based on the results and comparisons of Chapter 4 and Chapter 5.

Chapter 2 - Wind turbine system

In this Chapter, the wind turbine system is introduced, especially for components that are important for maximizing rotor power capture such as blade pitch, tip speed ratio, and rotor power coefficient. The dynamic model of the wind turbine system is then presented which will be used in the simulations presented in Chapter 4.

Important components

Because the research objective in this paper is to maximize the power capture of a wind turbine system, blade pitch, tip speed ratio, and the rotor power coefficient are then the most important quantities for controller design.

Blade pitch, denoted as β here, refers to the angle measure of the blades about a line along the axial direction of the blade. The pitch controls the amount of lift (torque) the blade receives. The pitch is measured in radians. For modern wind turbines, it is common to use individual actuators for each blade which increases the number of available inputs to the designer. However, in this work, it is assumed that the blade pitch is the same for all blades which is referred to as collective pitch. The pitch command loop is shown in Figure 2.1.

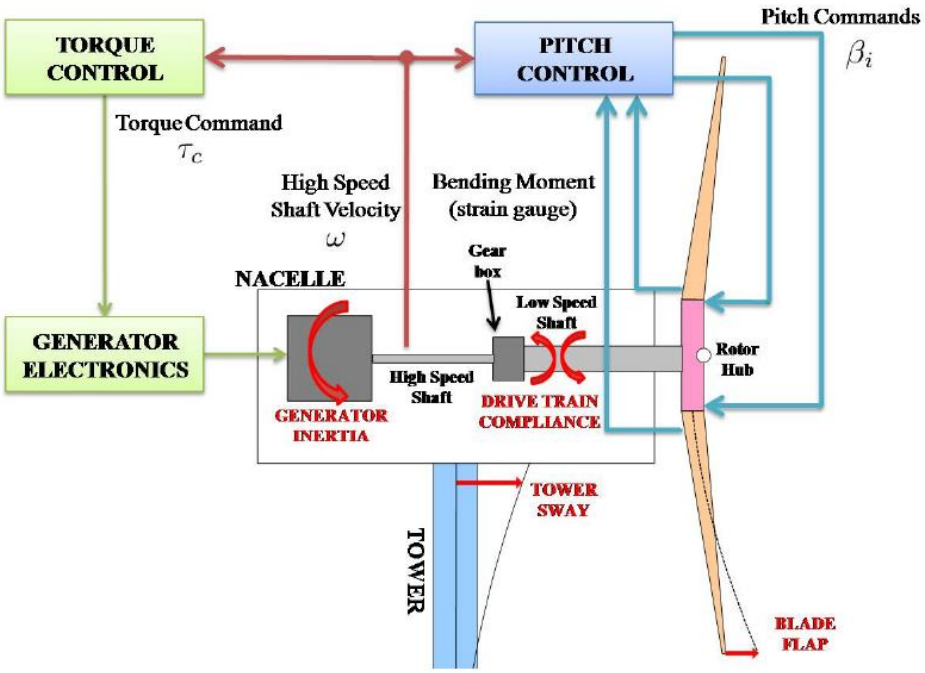


Figure 2.1 Signal flow of the wind turbine control system [21]

Tip speed ratio, denoted as λ, is the ratio of the rotational speed of the blade tip and the free stream velocity of the wind. It is a non-dimensional number given as

$$\lambda = Tip\ speed\ ratio = \frac{Tip\ speed\ of\ blade}{speed\ of\ wind} = \frac{\omega R}{v}. \tag{2.1}$$

Here, ω is the rotor rotational speed in radians/second, R is the rotor radius in meters, and v is the wind speed in meters/second. For a given wind turbine, the radius is a constant. From the equation above, if the wind is stable, the tip-speed ratio is proportional to the rotor rotational speed. As ω, R, v are all mearsurable, the tip-speed ratio can be considered measurable.

Assume that there is only horizontal wind and the yaw control has already been performed so that the shaft is in the same direction as the wind. The power available from the wind is expressed as

$$P_w = \frac{1}{2}\rho A v^3 = \frac{1}{2}\rho \pi R^2 v^3. \tag{2.2}$$

Here P_w is the power in the wind, ρ is air density, A is the swept area of the blades, R is the blade radius, and v is the wind speed along the shaft direction. Of course, it is desirable to extract the greatest power from the wind. However, unfortunately Betz's law in A.Betz (1926) states that, at most, 59.26% of the power available in the wind can be extracted under ideal conditions (ideal winds and ideal turbine rotor with infinite blades). Practically, there are only two or three blades for a wind turbine due to economic and structural reasons. So usually, 0.4 to 0.5 times of available wind power can be extracted by a wind turbine system. This ratio of extracted power to the wind power available is the rotor power coefficient, C_p, defined as

$$C_p = \frac{P_r}{P_w}. \tag{2.3}$$

Here P_r is the power captured by the rotor. Because C_p is related to both tip speed ratio and blade pitch, it can be considered as the function $C_p(\lambda, \beta)$.

The power capture coefficient is a very important parameter of the wind turbine. For a wind turbine with blade length of 63 meters, if it is in region 2 for 75% of the time and the average wind speed in region 2 is 80% of the rated speed, which is assumed as 12.5 m/sec, the total energy available from the wind in region 2 for 20 years is

$$Energy\ abailable = P_w t = \frac{1}{2}\rho\pi R^2 v^3 t$$

$$= \frac{1}{2} \times 1.25 \times 3.14 \times 63^2 \times (12.5 \times 0.8)^3 \times (20 \times 365 \times 24 \times 0.75)$$

$$= 1.03 \times 10^9 \text{ kWh.}$$

If the power coefficient is 1% off from the peak which is assumed as 0.45, the lost energy is

$$Energy\ lost = (C_{p\_max} - C_p)P_w t = 0.01 \times 0.45 \times 1.03 \times 10^9 = 4635000 \text{ kWh}$$

which is a huge amount of energy, where C_{p_max} is the maximum power capture coefficient and C_p is the power coefficient 1% off from the peak. The average price people in the U.S. pay for electricity is about 12 cents per kilowatt-hour. Then 4635000 kWh energy is $556,200. So maximization of the power capture coefficient is very necessary and useful.

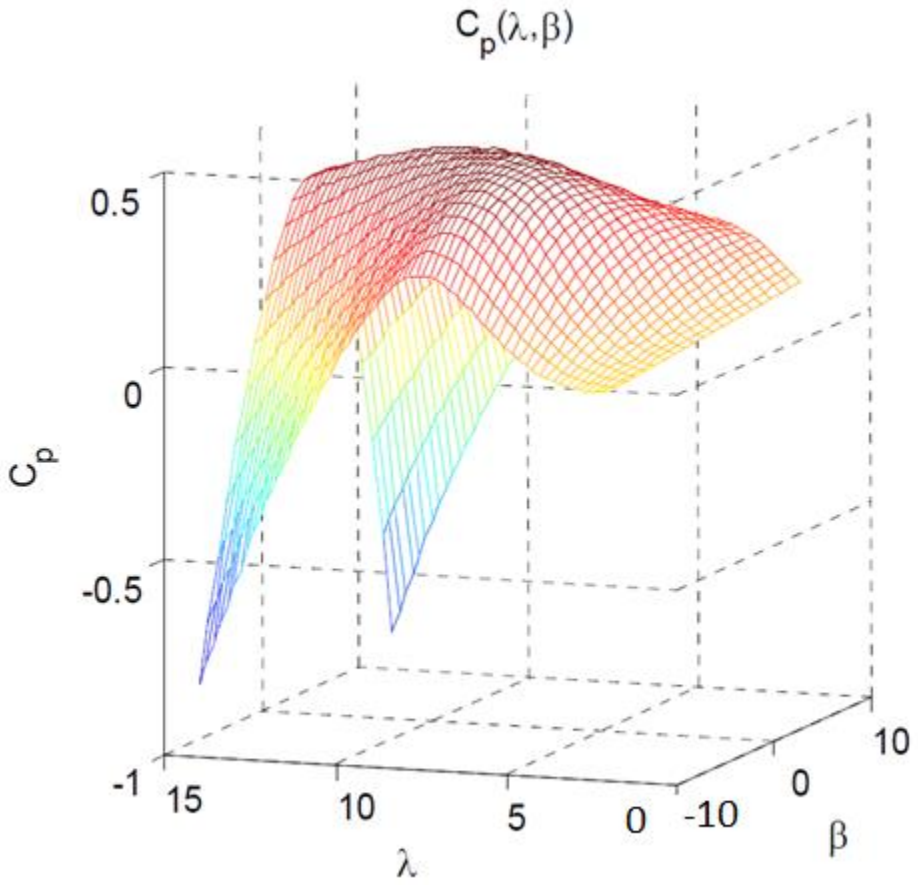


Figure 2.2 3D plot of $C_p(\lambda, \beta)$ [12]

Figure 2.2 shows the relation between $C_p(\lambda, \beta)$ and its variables λ, β for a wind turbine in which the unit of β is degree. From the figure, it can be seen that the power curve of this wind turbine has a global peak at certain values of λ and β. Let us denote these values as λ^* and β^*, then $C_p(\lambda^*, \beta^*)$ is the global extreme of rotor power coefficient. That is, when $\lambda = \lambda^*, \beta = \beta^*$, the wind turbine system can extract the greatest percentage of power from the wind.

From the previous development, the power generated by a wind turbine system can be expressed as

$$P_r = C_p P_w = \frac{1}{2} C_p(\lambda, \beta) \rho A v^3 = \frac{1}{2} C_p(\lambda, \beta) \rho \pi R^2 v^3 \ . \tag{2.4}$$

Because $P_r = \tau_{aero} \omega$, we also get

$$\tau_{aero} = \frac{P_r}{\omega} = \frac{\frac{1}{2} C_p(\lambda, \beta) \rho \pi R^2 v^3}{\omega} \tag{2.5}$$

where, τ_{aero} is the aerodynamic torque driving the wind turbine.

Dynamic model of wind turbine

The wind turbine system can be depicted simply as in Figure 2.3. Here J_m and J_e are mass moments of inertia for the turbine and generator, respectively, ω is the angular velocity of rotor, ω_e is the generator angular velocity, and T_m, T_e, T , and T_p are the torques at the turbine end, generator end, and before and after the gear box, respectively. Let D_m and D_e be viscous damping coefficients of the system for the rotor side and generator side, respectively.

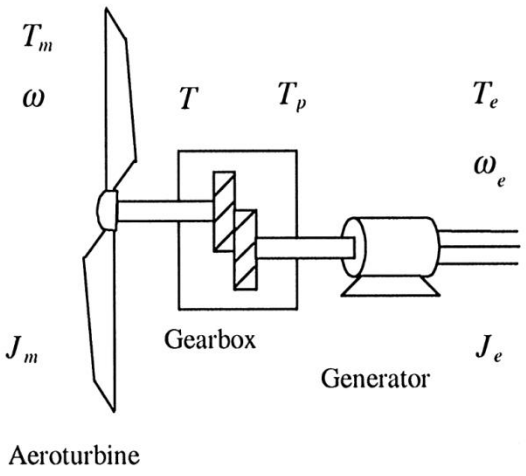


Figure 2.3 Diagram of wind power system [29]

The dynamic equations for both the turbine shaft and the generator are

$$T_m - T = J_m \frac{d^2\theta_m}{d^2t} + D_m \frac{d\theta_m}{dt} \tag{2.6}$$

and $$T_p - T_e = J_e \frac{d^2\theta_e}{d^2t} + D_e \frac{d\theta_e}{dt} \ . \tag{2.7}$$

Assume the gear ratio is $\frac{N_1}{N_2}$. We then have $\frac{\theta_m}{\theta_e} = \frac{N_2}{N_1}$ and

$$T = T_p \frac{N_1}{N_2} = \left(\frac{N_1}{N_2}\right)^2 \left(J_e \frac{d^2\theta_e}{d^2t} + D_e \frac{d\theta_e}{dt}\right) + \frac{N_1}{N_2} T_e. \tag{2.8}$$

Then

$$T_m = J_{equiv} \frac{d^2\theta_m}{d^2t} + D_{equiv} \frac{d\theta_m}{dt} + \frac{N_1}{N_2} T_e \tag{2.9}$$

Here, $J_{equiv} = J_1 + J_2 \left(\frac{N_1}{N_2}\right)^2$, $D_{equiv} = D_1 + D_2 \left(\frac{N_1}{N_2}\right)^2$. Because $\omega = \frac{d\theta_m}{dt}$, (2.9) can be also written as

$$T_m = J_{equiv}\dot{\omega} + D_{equiv}\omega + \frac{N_1}{N_2} T_e. \tag{2.10}$$

Generally, defining $T_m = \tau_{aero}$, $\frac{D_{equiv}}{J_{equiv}} = C_D \ and \ \frac{N_1}{N_2} T_e = \tau_C$, the dynamic model of wind turbine system can be described as

$$\dot{\omega} = \frac{1}{J}(\tau_{aero} - C_D\omega - \tau_C), \tag{2.11}$$

where τ_{aero} is the torque from the wind and τ_C is the reaction torque from the generator. $\dot{\omega}$ is angular acceleration of the rotor shaft. J is a constant parameter of the system and is the mass moment of inertia seen from the rotor shaft. C_D is the damping coefficient.

This dynamic model equation will be used in Chapter 4. Simulations there of the inner loop, outer loop, and both loops together are based on this dynamic model.

Chapter 3 - Control method

As mentioned in Chapter 2, there are two loops for the extremum seeking control in this work, the inner loop and the outer loop. The inner loop drives the wind turbine to a desired angular velocity and estimates the rotor power coefficient. The outer loop is responsible for seeking a path to maximize rotor power coefficient at the optimal tip speed ratio and blade pitch.

In this chapter, control methods for both the inner loop and the outer loop are introduced. The control strategy for the inner loop was proposed in B. Xian et al. (2004), and Tony Hawkins, et al. (2011) have used this controller for wind turbine power capture optimization. The proof of convergence of the controller which was given in B. Xian et al. (2004) is sound but hard for readers to see the origin of the control design. Here, a Lyapunov based deductive way is introduced to obtain the controller which provides an intuitive basis for the control design. The control method for the outer loop, SPSA, is then introduced including part of the technique's proof. Simulations of the method using MATLAB are provided.

Inner loop control strategy

Define the rotor speed error as

$$e(t) = \omega_d - \omega, \tag{3.1}$$

and filtered error as

$$r = \dot{e} + \alpha e \tag{3.2}$$

where α is a positive constant.

According to the dynamic model (2.11), it is assumed that the control law for the speed regulation is given by

$$\tau_{control} = -J\dot{\omega}_d - C_D\omega - J\alpha e + \hat{f}(t), \tag{3.3}$$

where ω_d is the desired angular velocity which is known while ω and e are obtained from the output of the wind turbine. $\hat{f}(t)$ is a function of time. Its expression will be given later in this section. It will be seen that $\hat{f}(t)$ will be an estimate of the aerodynamic torque.

Choose the candidate Lyapunov function as

$$V = \frac{1}{2}e^2 + \frac{1}{2}Jr^2 + P. \tag{3.4}$$

The Lyapunov function should be positive, so P in this equation needs to be derived so that it is non-negative. Taking the time derivative of V produces

$$\dot{V} = e\dot{e} + Jr\dot{r} + \dot{P} \tag{3.5}$$
$$= e(r - \alpha e) + rJ\dot{r} + \dot{P}.$$

Using (3.1), and (3.2) , it can be shown that $J\dot{r} = J\ddot{e} + J\alpha\dot{e}$. Substituting time derivation of (2.11) into (3.5), (3.5) becomes

$$\dot{V} = er - \alpha e^2 + r\left(-\dot{\tau}_{aero} - \dot{\hat{f}}(t)\right) + \dot{P}. \tag{3.6}$$

Choose

$$\dot{\hat{f}}(t) = Ar + B(t). \tag{3.7}$$

where A is a positive constant and $B(t)$ is an unknown function of t. Note that both sides of (3.7) are functions of time. The value of A needs to be determined together with the expression for $B(t)$ which make P positive and $\dot{V}$ negative. Substituting (3.7) into (3.6) produces

$$\dot{V} = er - \alpha e^2 + r\left(-\dot{\tau}_{aero} - rA - B(t)\right) + \dot{P} \tag{3.8}$$
$$= er - \alpha e^2 - r\dot{\tau}_{aero} - r^2A - rB(t) + \dot{P}$$
$$= -\alpha e^2 - r^2A + r(-\dot{\tau}_{aero} + e) - rB(t) + \dot{P}.$$

As $-\tau_{aero} + e$ is a function of angular velocity and time, $-\dot{\tau}_{aero} + e$ is a function of time as well as the angular velocity and its first time derivative. The quantity $-\dot{\tau}_{aero} + e$ defined as

$$N(t) = -\dot{\tau}_{aero}(\omega, \dot{\omega}, t) + e = N(\omega, \dot{\omega}, t), \tag{3.9}$$

and define
$$N_d(t) = N_d(\omega_d, \dot{\omega}_d, t) = -\dot{\tau}_{aero}(\omega_d, \dot{\omega}_d, t), \tag{3.10}$$

Let
$$\widetilde{N}(t) = N(\omega, \dot{\omega}, t) - N_d(\omega_d, \dot{\omega}_d, t) = N(t) - N_d(t). \tag{3.11}$$

where $\widetilde{N}(t)$ is the difference between $N(t)$ and $N_d(t)$.Then substituting $N(t)$ for $\dot{\tau}_{aero} + e$ while adding and subtracting $rN_d(t)$ produces

$$\dot{V} = -\alpha e^2 - r^2A + rN(t) - rN_d(t) + rN_d(t) - rB(t) + \dot{P} \tag{3.12}$$
$$= -\alpha e^2 - r^2A + r\left(N(t) - N_d(t)\right) + rN_d(t) - rB(t) + \dot{P}$$
$$= -\alpha e^2 - r^2A + r\widetilde{N}(t) + rN_d(t) - rB(t) + \dot{P}.$$

It is shown that $\widetilde{N}(t)$ is upper bounded in B. Xian et al. (2004), so, for a large enough A, $-\alpha e^2 - r^2A + r\widetilde{N}(t)$ is negative. As a result, A is chosen as (k_s+1), where k_s is a large positive constant. To make $\dot{V}$ negative, we require $rN_d(t) - rB(t) + \dot{P}$ to be zero. Choose

$$rN_d(t) - rB(t) + \dot{P} = 0 \tag{3.13}$$

so that
$$\dot{P} = rB(t) - rN_d(t) \tag{3.14}$$
$$= \dot{e}B(t) + \alpha eB(t) - \dot{e}N_d(t) - \alpha eN_d(t).$$

Notice that when t<0, there is no signal in the system, so both e and $\dot{e}$ are both zero. Integrating both sides of (3.14) with respect to time shows that

$$P = \int_0^t [\dot{e}B(\tau) + \alpha e B(\tau) - \dot{e}N_d(\tau) - \alpha e N_d(\tau)]d\tau \tag{3.15}$$

$$= \int_0^t [(\dot{e} + \alpha e)B(\tau)]\, d\tau - \int_0^t [(\dot{e} + \alpha e)N_d(\tau)]d\tau + C,$$

where C is a constant of inegration. Let $B(t) = \beta_c sgn(e)$, where $sgn()$ denotes the signum function. Then (3.15) becomes

$$P = \int_0^t [(\dot{e} + \alpha e)\beta_c sgn(e)]\, d\tau - \int_0^t [(\dot{e} + \alpha e)N_d(\tau)]d\tau + C \tag{3.16}$$

$$= \beta_c \int_0^t \dot{e} sgn(e)\, d\tau + \beta_c \int_0^t \alpha e sgn(e)\, d\tau - \int_0^t \dot{e}N_d(\tau)d\tau - \int_0^t \alpha e N_d(\tau)d\tau + C$$

$$= \beta_c sgn\big(e(t)\big)e(t) - \beta_c sgn\big(e(0)\big)e(0) - \left\{e(\tau)N_d(\tau)|_0^t - \int_0^t e(\tau)d[N_d(\tau)]\right\}$$

$$+ \int_0^t \alpha e[\beta_c sgn(e) - N_d(\tau)]d\tau + C$$

$$= \beta_c|e(t)| - \beta_c|e(0)| - e(t)N_d(t) + e(0)N_d(0) + \int_0^t e(\tau)d[N_d(\tau)]$$

$$+ \int_0^t \alpha e[\beta_c sgn(e) - N_d(\tau)]d\tau + C$$

$$= [\beta_c|e(t)| - e(t)N_d(t)] + [e(0)N_d(0) - \beta_c|e(0)|]$$

$$+ \int_0^t e(\tau)\big[\alpha\beta_c sgn(e) + \dot{N}_d(\tau) - \alpha N_d(\tau)\big]d\tau + C.$$

Because $-N_d(t) \geq -|N_d(t)|$, $e(\tau)\dot{N}_d(\tau) \geq -\big|\dot{N}_d(\tau)\big||e(\tau)|$ and $e(\tau)N_d(\tau) \geq -|N_d(\tau)||e(\tau)|$, then

$$P \geq |e(t)|(\beta_c - |N_d(t)|) + [e(0)N_d(0) - \beta_c|e(0)|]$$

$$+ \int_0^t \alpha\beta_c|e(\tau)| - \big|\dot{N}_d(\tau)\big||e(\tau)| - \alpha|N_d(\tau)||e(\tau)|]d\tau + C. \tag{3.17}$$

To make P positive, β_c should satisfy

$$\beta_c > |N_d(t)| \; and \; \beta_c > |N_d(t)| + \left|\frac{\dot{N}_d(\tau)}{\alpha}\right|. \tag{3.18}$$

So, choose $\beta_c > |N_d(t)| + \left|\frac{\dot{N}_d(\tau)}{\alpha}\right| = |\dot{\tau}_{aero}| + \left|\frac{\ddot{\tau}_{aero}}{\alpha}\right|$ and let $C = -e(0)N_d(0) + \beta_c|e(0)|$, then P is a positive value. Then, according to (3.4) and (3.12), V is positive and $\dot{V}$ is negative. Thus,

$$\dot{\hat{f}}(t) = (k_s + 1)r + B(t)\beta_c sgn(e). \quad (3.19)$$

Integrating both sides, we get

$$\hat{f}(t) = (k_s + 1)[e(t) - e(0)] + \int_0^t [(k_s + 1)\alpha e(\tau) + \beta_c sgn(e(\tau))]d\tau. \quad (3.20)$$

Then substituting this into (3.3), we have

$$\tau_{control} = -J\dot{\omega}_d - C_D\omega - J\alpha e + (k_s + 1)[e(t) - e(0)] + \int_0^t \left[(k_s + 1)\alpha e(\tau) + \beta_c sgn\big(e(\tau)\big)\right]d\tau. \quad (3.21)$$

It is shown that as $t \to \infty$, $\hat{f}(t) \to \tau_{aero}$ in B. Xian et al. (2004). Using this property as well as the relationship between τ_{aero} and C_p , the rotor power coefficient can be estimated.

SPSA method

The optimization algorithm is a procedure to change the adjustable parameters step by step from the initial value to a value which improves the objective function. Several classical gradient-based algorithms have been developed and applied to various areas. However, for problems having many variables and uncertain measurements, these methods are not that efficient. Motivated by these reasons, gradient-free stochastic algorithms have been studied by many researchers in the past decades [P. Ghisbain (2009)].

Simultaneous Perturbation Stochastic Approximation (SPSA) is a gradient-free, extremum seeking algorithm proposed in Spall, J. C (1998). Usually, due to uncertainties and complexity of parameters as well as relationships between them, it is very hard to develop an exact model of a system so that the gradient is hard to evaluate. However, SPSA is an excellent method for finding the global minimum of such systems, regardless of the dimension. It only needs two objective function measurements to take a step toward the next extremum approximation. Manzie, C., & Krstic, M. (2009) shows that convergence towards the extremum of a static map can be guaranteed with a stochastic extremum seeking algorithm, and quantify the behavior of a system with Gaussian-distributed perturbations at the extremum in terms of the extremum seeking constants and map parameters. Figure 3.1 shows a typical extremum seeking path using the SPSA methods. It can be seen that the objective function is not uniformly

decreasing, but finally it will converge to the extremum. The SPSA method has already been successfully applied to areas such as signal timing for traffic control in Spall, James C., and Daniel C. Chin (1998), optimal targeting of weapon systems in Heydon, B.D. and Hill, S.D. (2003), and so on.

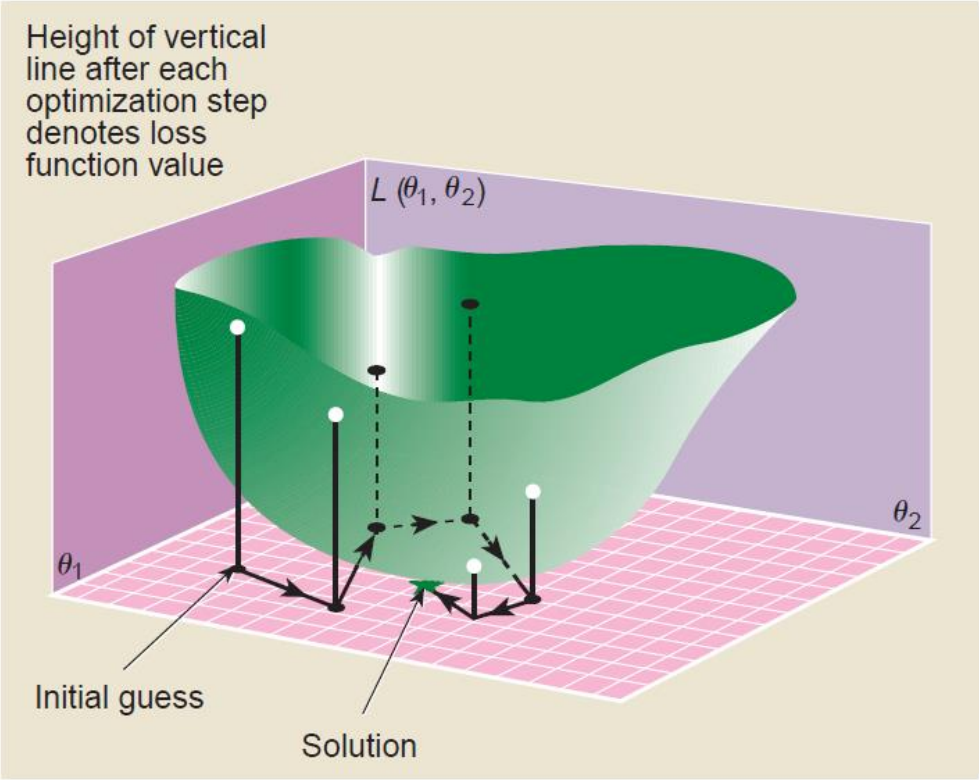


Figure 3.1 3D plot of the path of the SPSA method [30]

Let $L(\theta)$ and θ^* to be the objective function and its optimal argument, respectively. Then,

$$\theta^* = \arg\min L(\theta)\{\theta \epsilon U\} \tag{3.22}$$

where U is the domain of $\theta \in \boldsymbol{R}^p$ and 'arg min' means the arguments for the minimum of $L(\theta)$. Denote

$$J(\theta) = L(\theta) + \varepsilon \tag{3.23}$$

where $J(\theta)$ is the measurement of $L(\theta)$ and ε is the error in the measurement. In the SPSA method, because the gradient of $L(\theta)$ is not available, a random simultaneous perturbation of θ

denoted as $C_n\Delta_n = C_n(\Delta_1, \Delta_2, \dots, \Delta_p)^T$ is taken for perturbing the arguments and estimating the gradient and C_n is defined below. The SPSA algorithm for updating the arguments is given by

$$\theta_{n+1} = \theta_n - a_n \hat{g}_n(\theta_n) \tag{3.24}$$

where, $\theta_n = ((\theta_n)_1, (\theta_n)_2, , , (\theta_n)_p)^T$ is the n_{th} iterate, $\hat{g}_n(\theta_n)$ is the estimated gradient of $g(\theta)$, namely $J(\theta)$ evaluated at θ_n and $\{a_n\}$ is a positive scalar sequence converging to zero. Taking two measurements, $J^+(\theta_n) = L(\theta_n + c_n\Delta_n) + \varepsilon^+, J^-(\theta_n) = L(\theta_n - c_n\Delta_n) + \varepsilon^-$, we obtain the i_{th} component of $\hat{g}_n(\theta_n)$ as an unbiased estimator of $L_{\theta_i}(\theta_n)$ which will be shown later. Here, J^+, J^- are two measurements and $L_{\theta_i} = \frac{\partial L}{\partial \theta_i}$ and

$$\hat{g}_n(\theta_n)_{\,i} = \frac{J(\theta_n + c_n\Delta_n) - J(\theta_n - c_n\Delta_n)}{2c_n(\Delta_n)_i}, i \in \{1,2, \dots, p\} \tag{3.25}$$

where c_n is a small positive number which is decreasing as n increases. The standard choice of c_k is $c_k = {}^{c}/_{k^\gamma}$ for some $\gamma > 0$. $\Delta_n = (\Delta_1, \dots, \Delta_p)_n^T$ is the random vector for the n_{th} iteration changing on all arguments with $(\Delta_i)_n$ $(i = 1,2, \dots, p)$ being mutually independent, zero-mean, and bounded second order moments. A standard choice is a Bernoulli sequence, where Δ_i = ±1with probability1/2 for each possibility. Let $g(\theta_n) = \left(g_1(\theta_n), \dots, g_p(\theta_n)\right), \Delta_n^{-1} = (\Delta_1^{-1}, \dots, \Delta_p^{-1})_n^T$, then substituting (3.22) and (3.24), we have

$$g(\theta_n) = \Delta_n^{-1}\frac{L(\theta_n + c_n\Delta_n) - L(\theta_n - c_n\Delta_n)}{2c_n} + \frac{\Delta_n^{-1}(\epsilon^+ - \epsilon^-)}{2c_n}. \tag{3.26}$$

Using a Taylor expansion of the objective function $L(u_n + c_n\Delta_n)$ and$L(u_n - c_n\Delta_n)$, the first term on the right side in (3.25) can be written as

$$\Delta_n^{-1}\frac{L(\theta_n + c_n\Delta_n) - L(\theta_n - c_n\Delta_n)}{2c_n} = \Delta_n^{-1}{\Delta_n}^T L_\theta(\theta_n) + O({c_n}^2). \tag{3.27}$$

For the first term on the right hand side of (3.26), the expected value of $\Delta_n^{-1}{\Delta_n}^T$ denoted as $\left(\Delta_n^{-1}{\Delta_n}^T\right)$ would be $E\left(\Delta_n^{-1}{\Delta_n}^T\right) = I$. It follows that

$$E\left(\Delta_n^{-1}{\Delta_n}^T L(\theta_n)\right) = L(\theta_n) \tag{3.28}$$

where $E(\cdot)$ is the expected value. The error includes three terms which are expressed as

$$e_r = \left(\Delta_n^{-1}{\Delta_n}^T - I\right)L_\theta(\theta_n), e_a = O({c_n}^2), e_m = \frac{\Delta_n^{-1}(\epsilon^+ - \epsilon^-)}{2c_n} \tag{3.29}$$

where e_r comes from the randomization of $(\Delta_i)_n$, e_a comes from approximation, and e_m comes from the measurement. The details of the proof of convergence of the SPSA method for differentiable and convex objective functions can be seen in M. C. Fu and S. D. Hill (1997). Because the rotor power coefficient $C_p(\lambda, \beta)$ is a differentiable convex function of the tip speed ratio and the blade pitch, the SPSA method can be applied to a wind turbine system for finding the global extreme point. From the stability analysis of the inner loop controller, we know that when $t \to \infty$, $\hat{f}(t) \to \tau_{aero}$. So τ_{aero} can be estimated by using $\hat{f}(t)$ after performing the estimation for a long enough time period. Thus,

$$\hat{P}_r = \hat{\tau}_{aero}\omega = \hat{f}(t)\omega \tag{3.30}$$

and

$$\hat{C}_p = \frac{P_r}{P_w} = \frac{\hat{f}(t)\omega}{\frac{1}{2}\rho\pi R^2 v^3}\,. \tag{3.31}$$

Compared to a classical gradient based method, SPSA has two main advantages. Firstly, the SPSA method is specifically efficient for multi-variable extremum seeking problems. Usually, classical methods vary one variable at a time. If the objective function is p dimensional, at least $n \cdot p$ measurements are needed. For the SPSA method, it varies multiple-variables at a time. No matter what is the value of p, it only needs 2 measurements. Secondly, it is shown in Spall, James C (1998) that the SPSA method can do much better than the classical methods when the measurements are corrupted by noise.

Simulations using MATLAB[see Appendix A for code listing] have been made to test the SPSA method using parameters of Table 3.1 and the function $\mathrm{Cp} = -0.6e^{-0.03(\lambda-6)^2}e^{-0.03(\beta-2)^2}$ as the objective function with two arguments which has a global peak of -0.6 for $\lambda = 6$ and $\beta = 2$ for testing the SPSA method. Because SPSA is for finding the mimimum of the objective function while maximum of Cp is the goal, the opposite of Cp is used as the objective function. Here, it is assumed in the simulations that noise free values of objective function are available. The results for λ, β and the objective function y are shown in Figs. 3.2,3.3, and 3.4, respectively.

Table 3.1 Parameters of m-file of SPSA method

Parameter	Value	Description
p	2	Dimension of parameters in objective function
a	5	Related to step size ak
A	1	Related to step size ak
c	20	Related to step size ck
alpha	0.2	Related to step size ak
gamma	5	Related to step size ck
n	500	Steps for running the code
theta	[0 4]'	Initial value of the parameters
theta*	[6 2]'	Values of parameters of extreme point

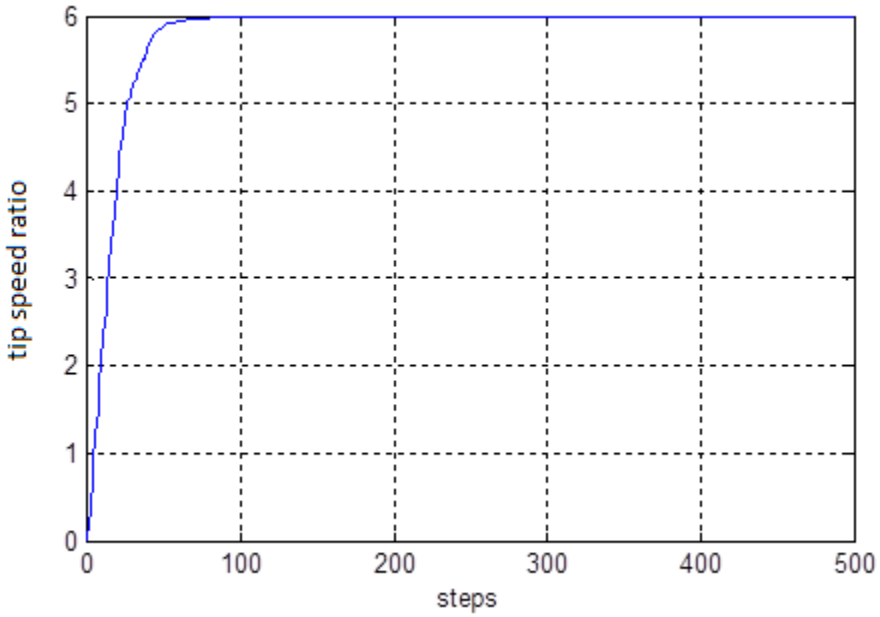


Figure 3.2 The plot of tip speed ratio

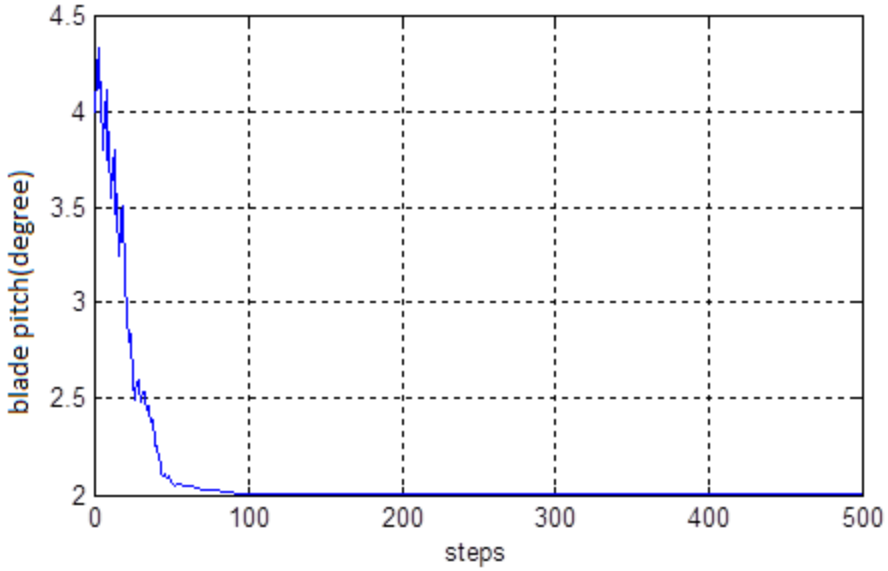


Figure 3.3 The plot of blade pitch

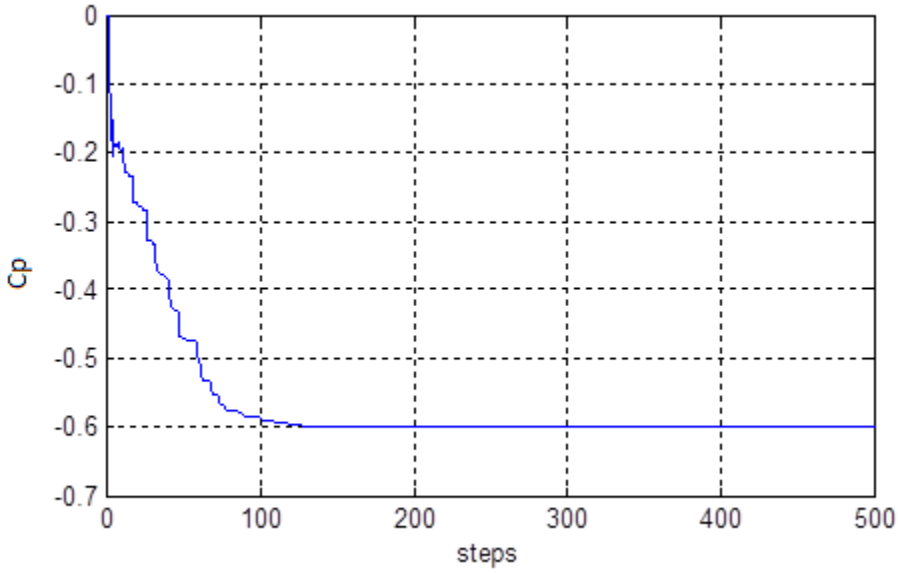


Figure 3.4 The plot of Cp

In Figs. 3.2-3.4, it can be seen that the SPSA method completed the extreme seeking task very well requiring about 100 steps for λ and β to reach the extremum point of the objective function in a stable manner.

In this paper, $C_p(\lambda, \beta)$ is a two dimensional, differentiable, convex function. Because $\hat{g}_n(u_n) = \frac{J(u_n+c_n\Delta_n)-J(u_n-c_n\Delta_n)}{2c_n(\Delta_n)_i}$ is just an approximation for the gradient which require two measurements of the objective function, can $\hat{g}_n(u_n)$ be computed by $\hat{g}_n(u_n) = \frac{J(u_n+c_n\Delta_n)-J(u_n)}{c_n(\Delta_n)_i}$ as an approximation which only needs one measurement of the objective function? The answer is yes. In Chapter 4, both means of computing $\hat{g}_n(u_n)$ will be tested. The results will be seen to be good.

Chapter 4 - Simulations in MATLAB

In Chapter 2, a dynamic model of a wind turbine system was introduced. In this chapter, simulations based on that model will be conducted using the MATLAB platform to test both the inner loop control on speed regulation and aerodynamic torque estimation together with the SPSA method for extremum seeking.

As mentioned in Chapter 2, the model can be expressed by the differential equation

$$\dot{\omega} = \frac{1}{J}(\tau_{aero} - C_D\omega - \tau_C). \tag{4.1}$$

All model parameters are shown in Table 4.1.

Table 4.1 Parameters of the dynamic model

Parameter	Value	Unit	Description
J	100000	$Kg.m^2$	Moment of inertia
R	40	m	Radius of blade
C_D	1	$Kg.m^2/sec$	Damping coefficient
ρ	1.25	Kg/m^3	Air density

In order to simulate the wind turbine system and test the effect of the SPSA method, the power coefficient is chosen to be

$$C_p = 0.6e^{-0.3(x-4)^2}e^{-0.3(y-2)^2} \tag{4.2}$$

which has a peak of 0.6 with a tip-speed ratio of 4 and a blade pitch of 2 degrees. A surface plot of (4.2) is shown in Figure 4.1. This is the negative of the power coefficient used in the SPSA simulations of Chapter 3.

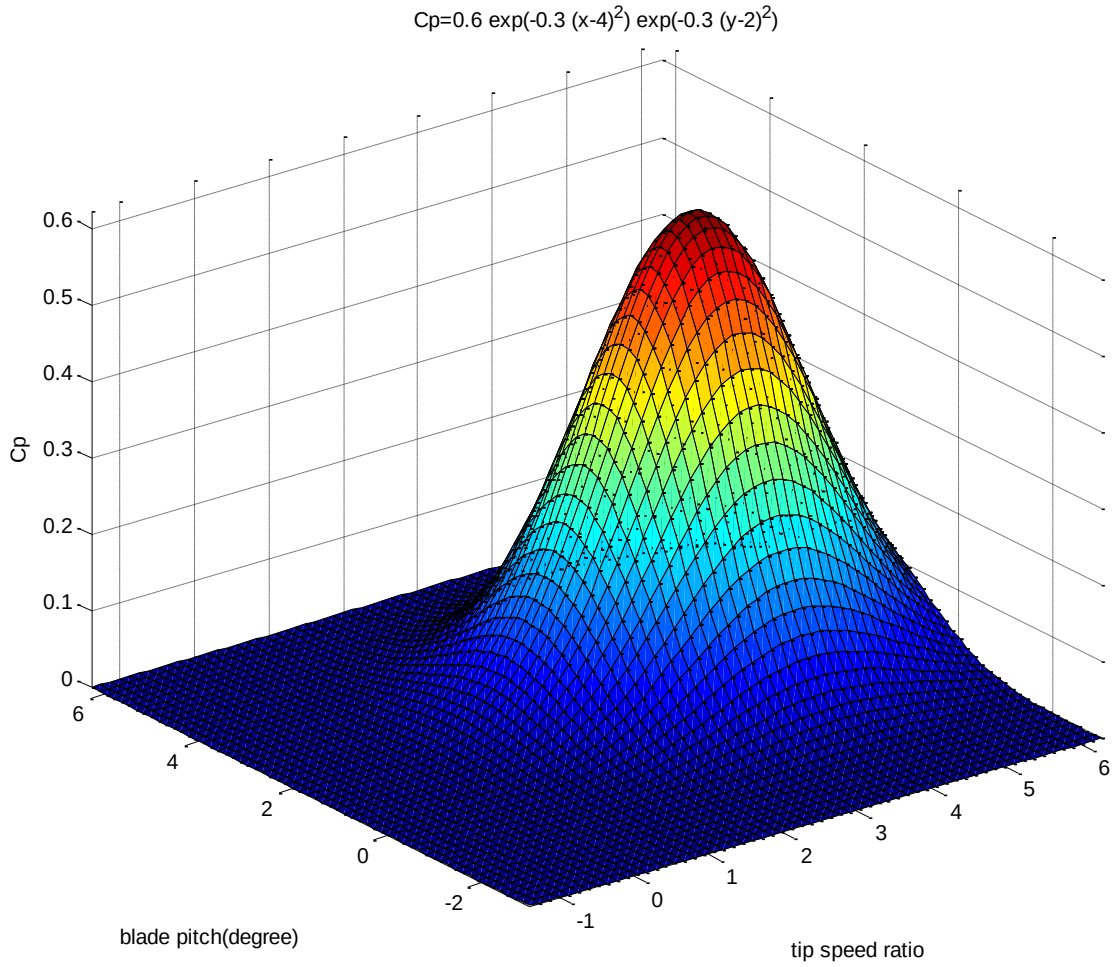


Figure 4.1 3D plot of $0.6e^{-0.3(x-4)^2}e^{-0.3(y-2)^2}$

Inner loop simulation

This simulation will test the speed regulation for the inner loop controller. A step change in rotor speed will be used in this simulation to investigate the performance of the system. The wind speed and blade pitch are both set to initial values. The wind speed is 10m/s and the blade pitch is 2 degrees. The initial angular velocity is set as 0.01 rad/sec. Table 4.2 shows the values of parameters of this robust inner loop controller.

Table 4.2 Parameters of inner loop controller

Parameter	Value	Description
k_s	10^7	controller gain
β_c	2	controller gain
α	3	controller gain

The simulation diagram is shown in Figure 4.2 in which the actual Cp is used in the model but not in the regulator. The control block collects the signals of angular velocity of the turbine and the desired angular velocity to create a control signal according to the inner loop control law. The update signal, which is set as zero and not used here, will be used for the outer loop control. The variables block collects wind speed, blade pitch, and angular velocity of the turbine to calculate necessary variables, such as torque available from the wind, lambda, beta, as well as the power coefficient for determining the aerodynamic input torque to the turbine differential equation. The plant block contains the dynamic differential equation of the turbine. The contents of the sub-blocks are listed in Appendix B.

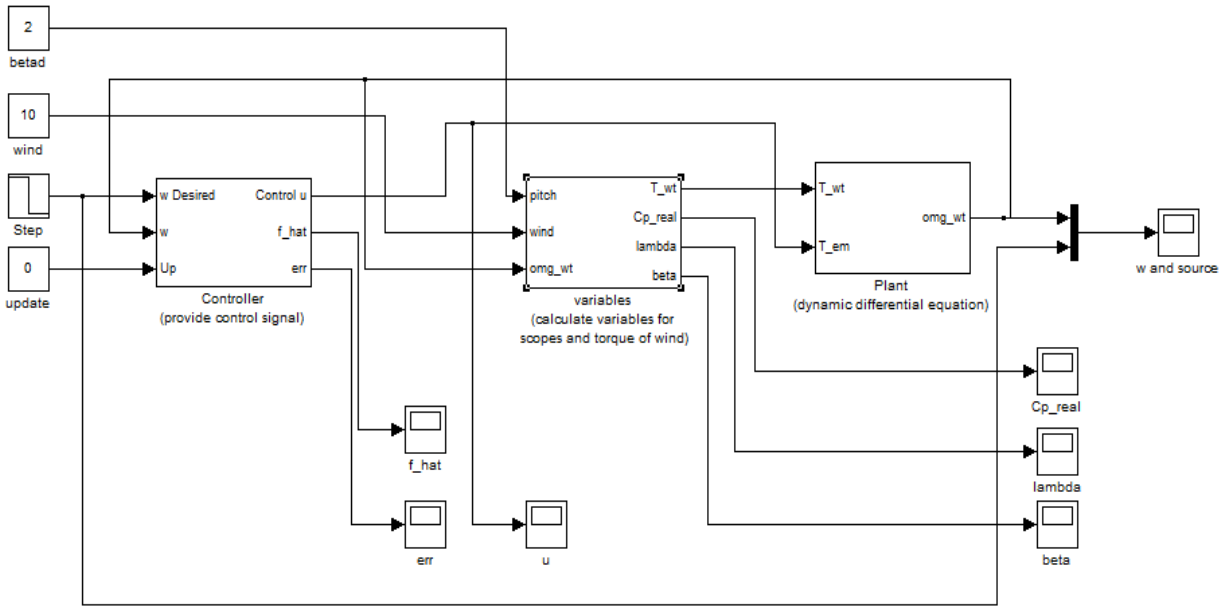
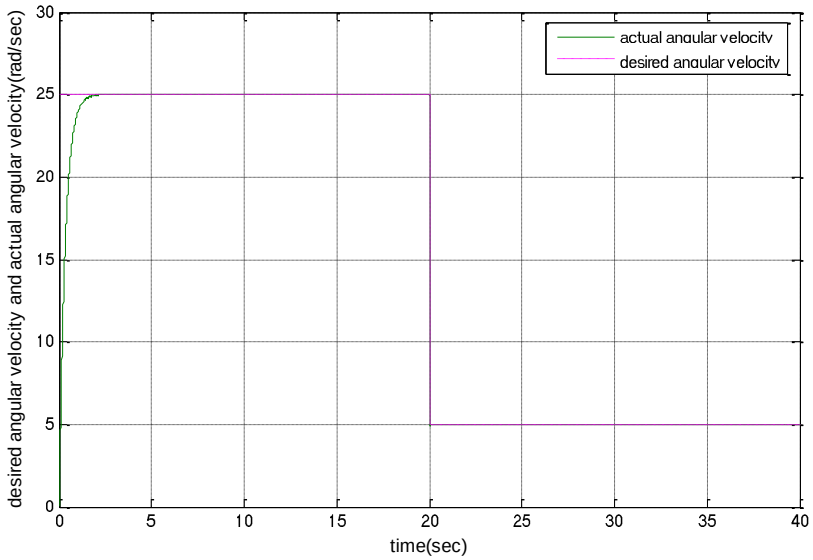
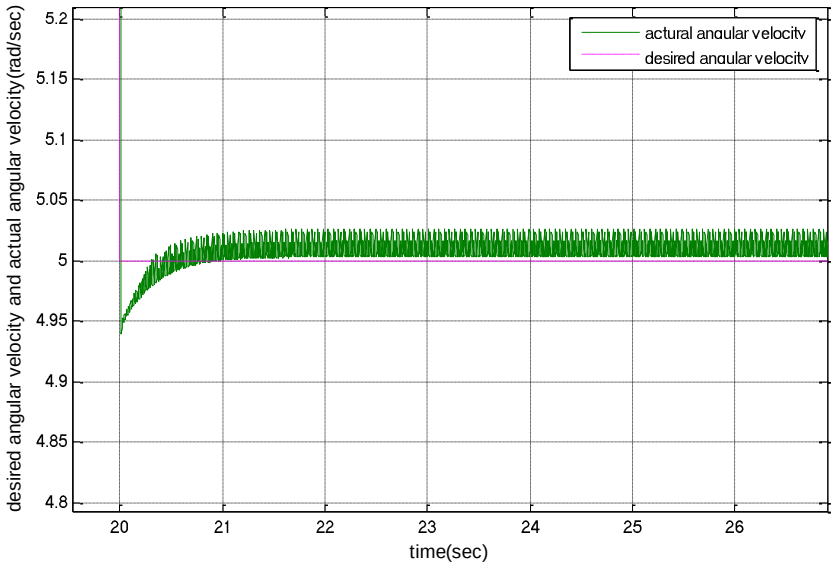


Figure 4.2 Simulation diagram of inner loop

Figure 4.3 shows a plot of the angular velocity response of the plant as well as the desired angular velocity. Figure 4.3(b) is the exploded part of the negative going portion of 4.3(a).The purple curve is the desired rotor speed which is a step command, while the green curve is the rotor angular velocity. The rotor angular velocity response follows the input signal very well. It takes 2 second to drive the angular velocity to the desired value.



(a)



(b)

Figure 4.3 Plot of actual angular velocity comparing with desired one

Outer loop simulation

For testing the SPSA method for outer loop extremum seeking regulation, the Cp expression of (4.2) will still be used as our rotor power coefficient function. Because the SPSA algorithm is for seeking a minimum point of the objective function, $-C_p$ is used as the objective function. Parameters shown in Table 4.1 are still used for the SPSA algorithm. Considering that it will take 2 seconds for the system to reach a stable estimate of Cp and to achieve the desired angular velocity and blade pitch, the time of updating the desired tip speed ratio and blade pitch is set as 2 seconds. The SPSA method for optimization of the objective function requires two points where Cp must be estimated. A period of 6 seconds is alloted to arrive at the next desired position on the Cp surface. The first 2 seconds in the period is for estimating the objective function at the first random point. The next 2 seconds in the period is for obtaining an estimate of objective function at the second random point. During the last 2 seconds in a period, the system is driven to a new desired point base on the first two. The simulation diagram is shown in Figure 4.4. The contents of the sub-blocks are listed in Appendix B.

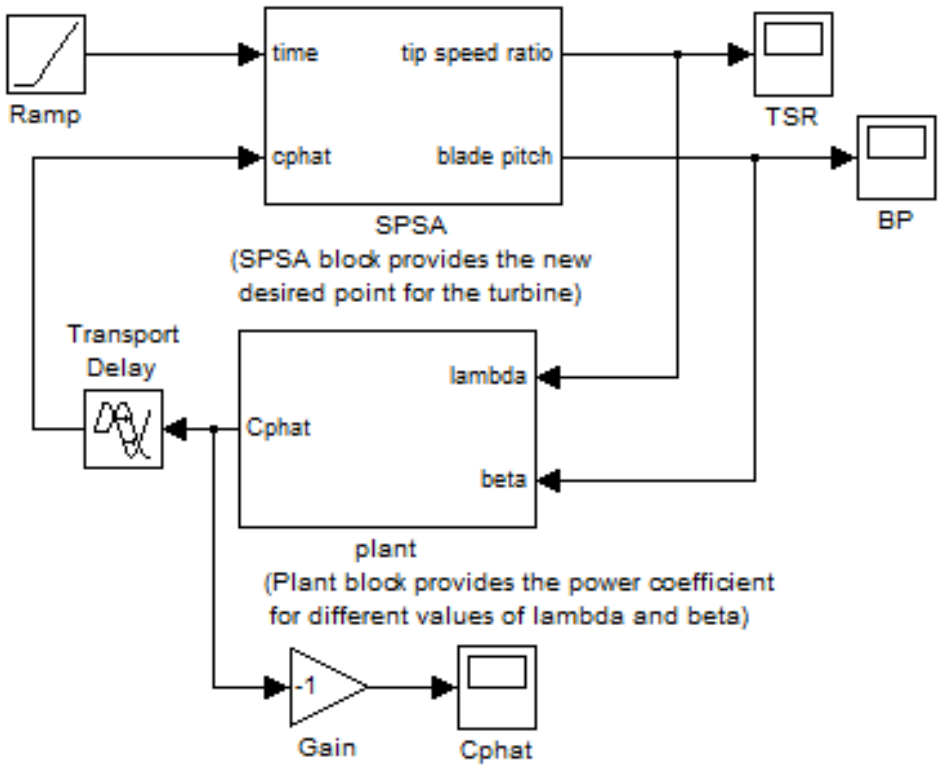
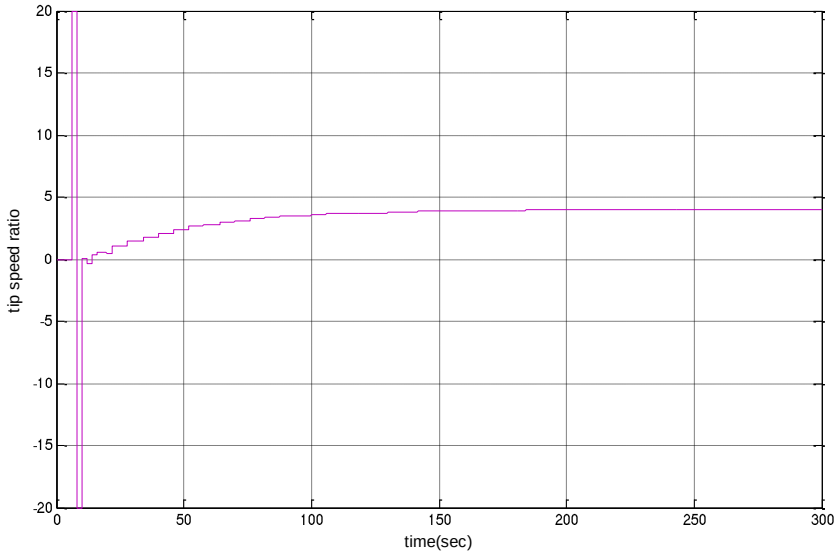


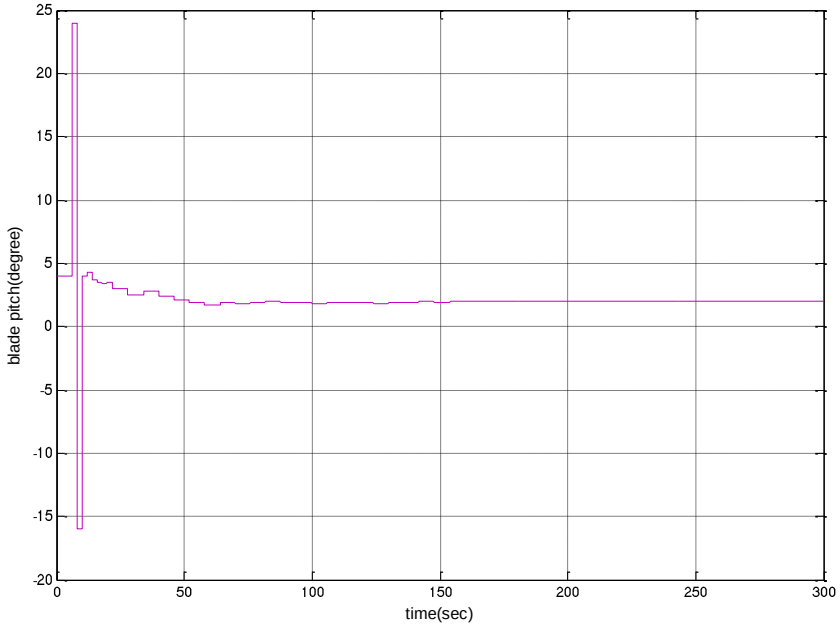
Figure 4.4 Diagram of outer loop

The SPSA block in Figure 4.4 collects measurements of Cp for different values of lambda and beta and then provides new values of lambda and beta for the next step. The plant block provides values of the power coefficient for different values of lambda and beta. The transport delay block, in which the delay is set as zero seconds, is to block any arithmetic loop messages in MATLAB.

The SPSA is the algorithm for the outer loop. The ramp is used as a time signal for calculating the time used to time the update. Figure 4.5 shows the outputs of angular velocity and blade pitch.



(a) tip speed ratio



(b) blade pitch

Figure 4.5 Plot of tip speed ratio and blade pitch of outer loop simulation

From these two plots, it can be seen that both the tip speed ratio and blade pitch go to the values maximizing Cp in around 150 seconds. In Figure 4.6, it can be seen that Cp goes to its maximum of 0.6 at about 150 seconds when lambda and beta become constant.

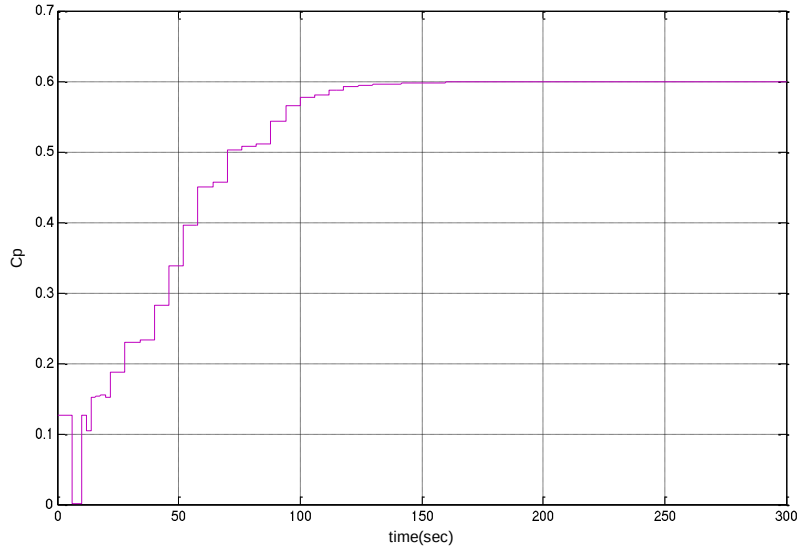


Figure 4.6 Cp of outer loop simulation

Next, the modified SPSA method is going to be simulated in MATLAB and it will be seen that the performance is better than the original SPSA algorithm. All simulation parameters stay the same as listed in Table 4.1. The simulation diagram is shown in Figure 4.7 in which the content of Plant block is the same as in Figure 4.4, while the code in the SPSA block is slightly different which can be seen in Appendix B. The results are shown in Figures 4.8-4.10.

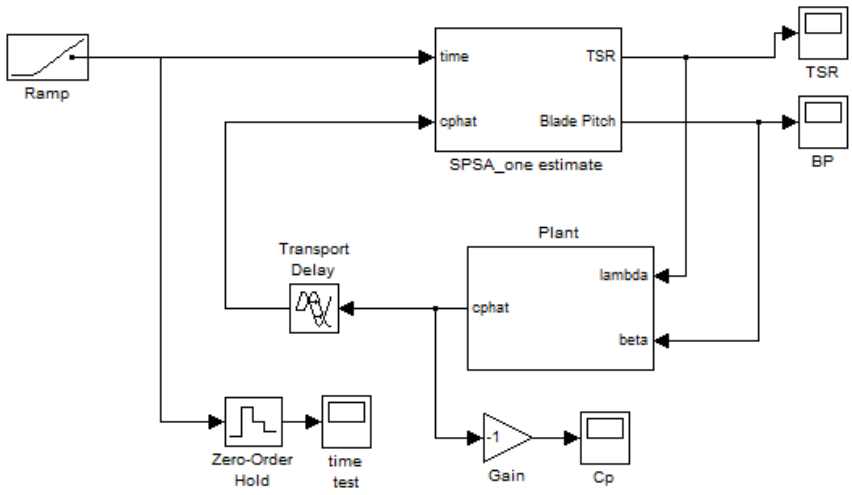


Figure 4.7 Simulation diagram of outer loop

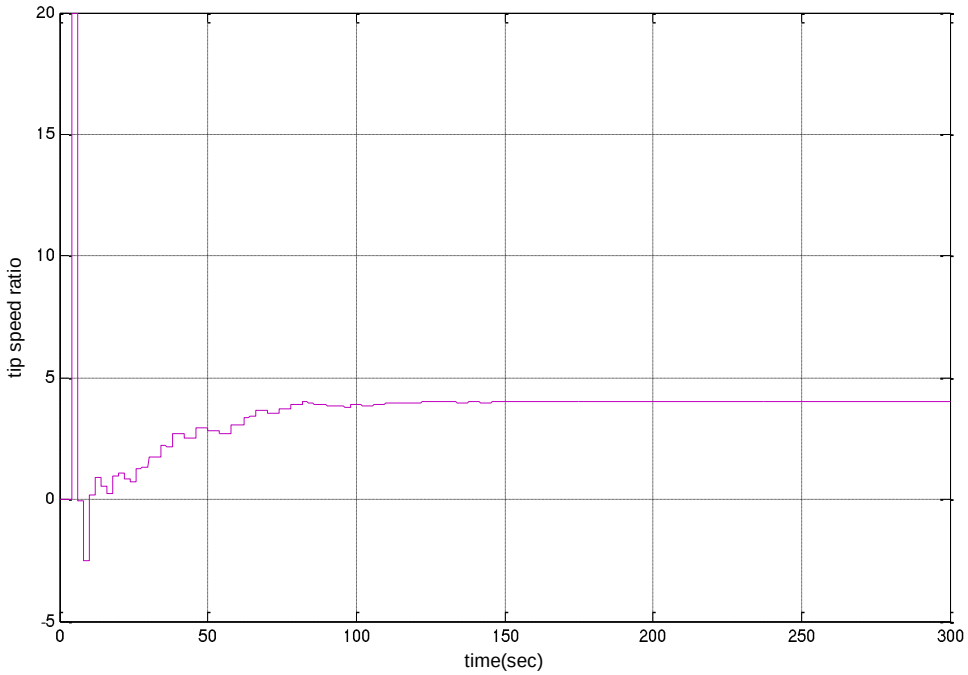


Figure 4.8 tip speed ratio

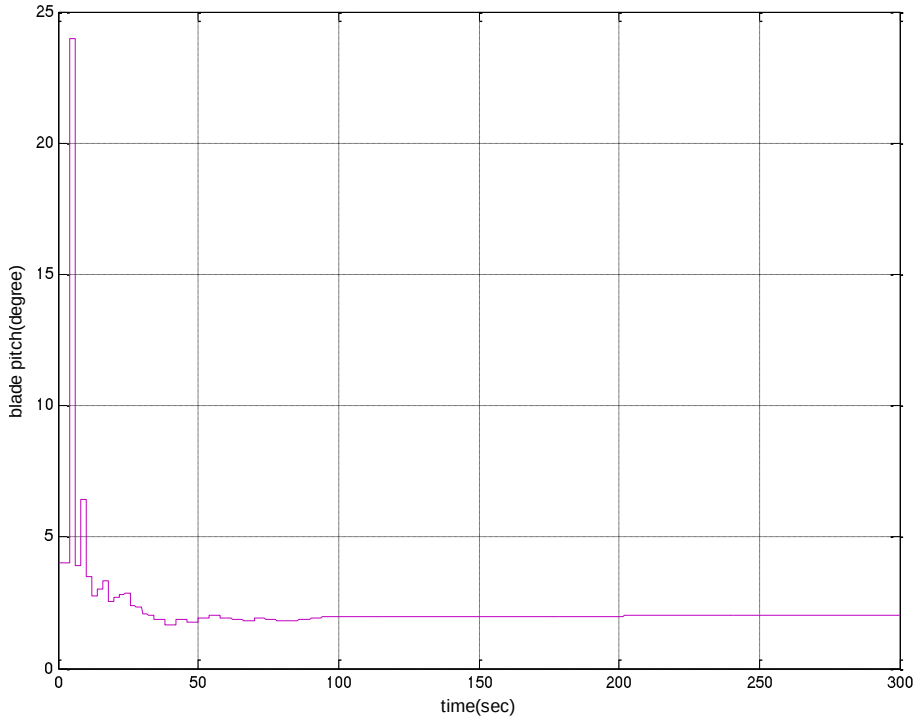


Figure 4.9 blade pitch

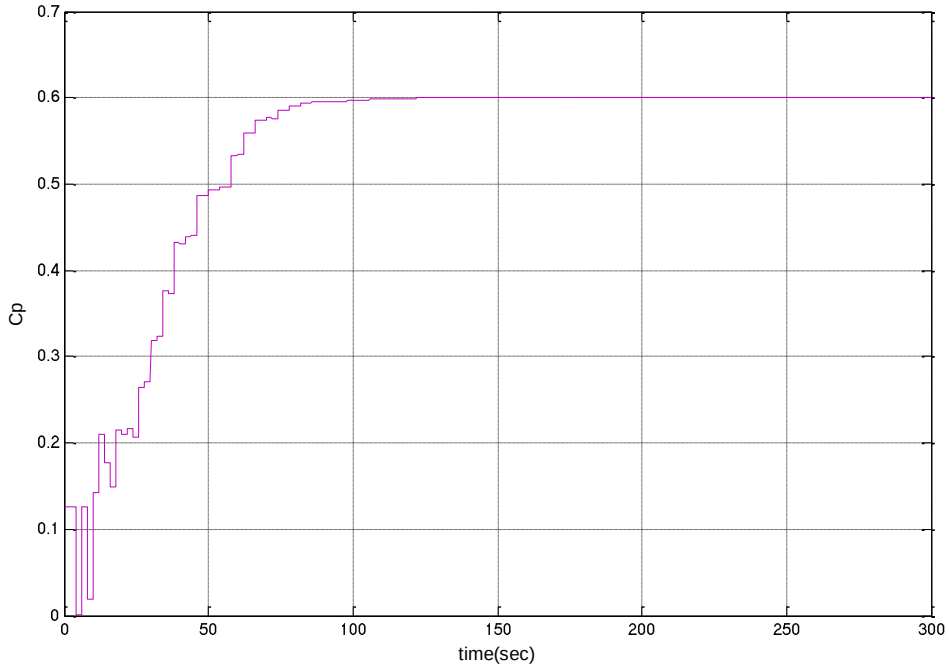


Figure 4.10 Cp

Comparing the results of the modified SPSA method with the original SPSA algorithm, it is seen that both drive the system to the extremum. However, the modified algorithm reaches the extremum in less time. It reaches the maximum in less than 100 seconds which is smaller than the 150 seconds of the original SPSA method. This is because only one rotor power coefficient is estimated before determining the desired tip speed ratio and blade pitch for next step. Also, the negative overshoot at the very beginning for TSR plot is much smaller. This means the wind turbines do not need to change the spin direction often. Similarly, the negative overshoot for blade pitch disappeared completely. In the next section, the modified SPSA will be used with the inner loop instead of the original SPSA method because it has better performances.

Simulation combining inner and outer loop

After introducing the dynamic model of the wind turbine, simulating the inner loop controller and the outer loop modified SPSA algorithm, these separate parts will be combined together. The blade pitch dynamics are neglected in this section. However, for the simulation in FAST, the blade pitch dynamics could not be neglected because it is already coded in FAST, so a PI controller will be used. The parameters of the wind turbine are as shown in Table 4.1. The Cp function is still $C_p = 0.6e^{-0.3(\lambda-4)^2}e^{-0.3(\beta-2)^2}$. Figure 4.11 shows the simulation diagram and Figures 4.12-4.14 show the results. The sub-block contents are listed in Appendix B.

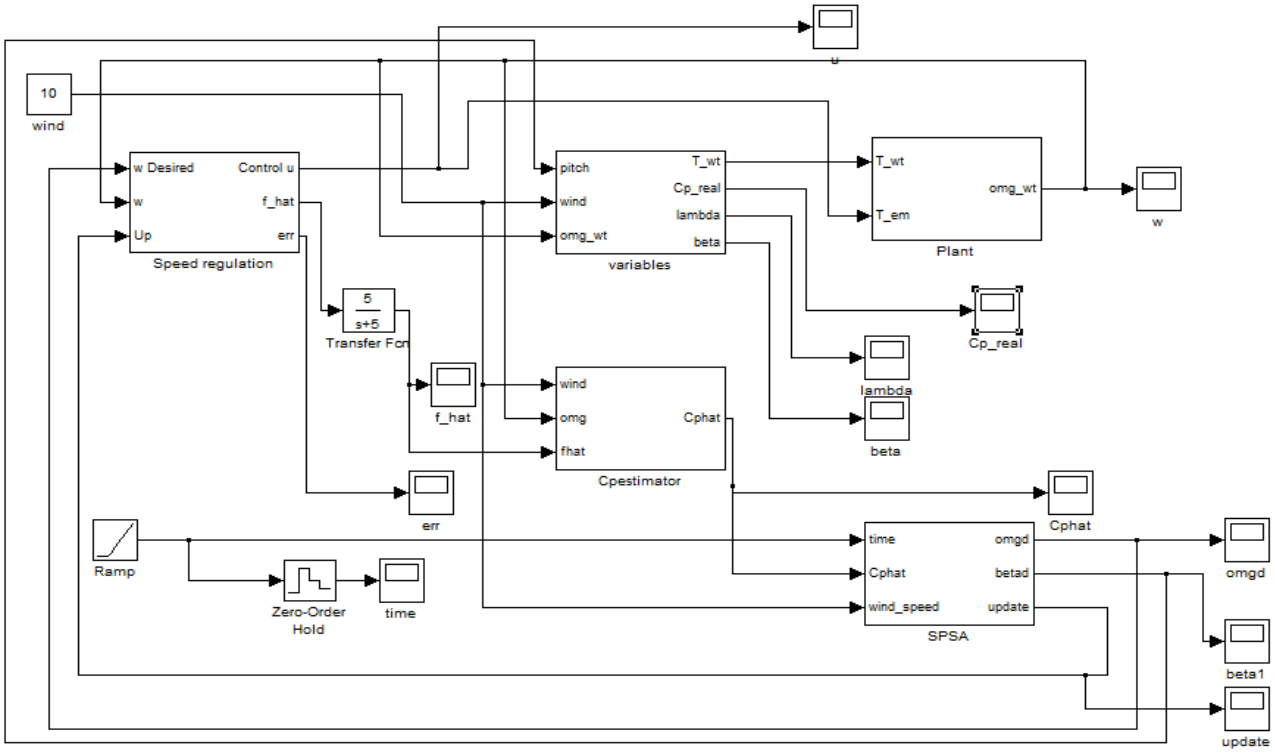


Figure 4.11 Simulation diagram with both inner loop and outer loop

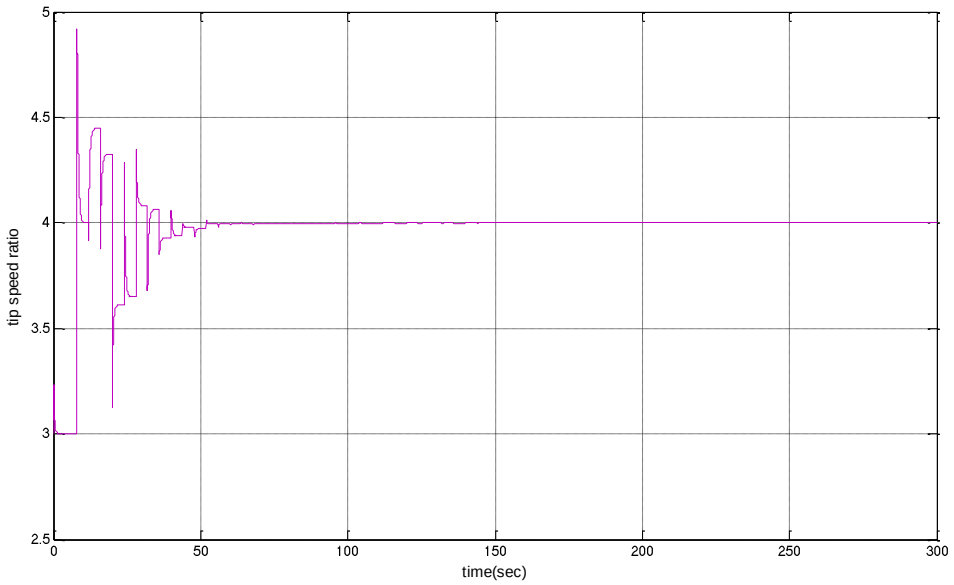


Figure 4.12 Tip Speed Ratio

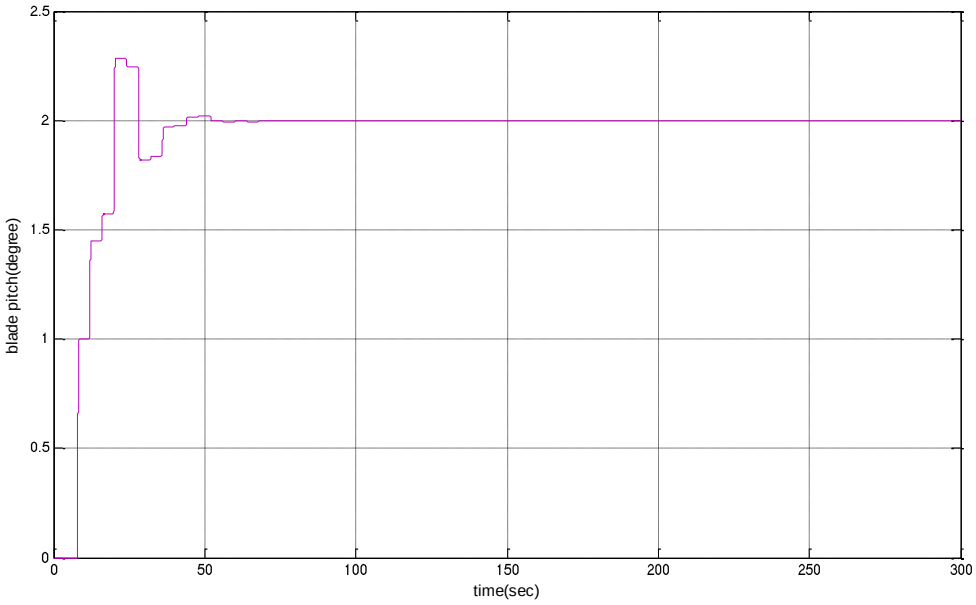


Figure 4.13 Blade Pitch

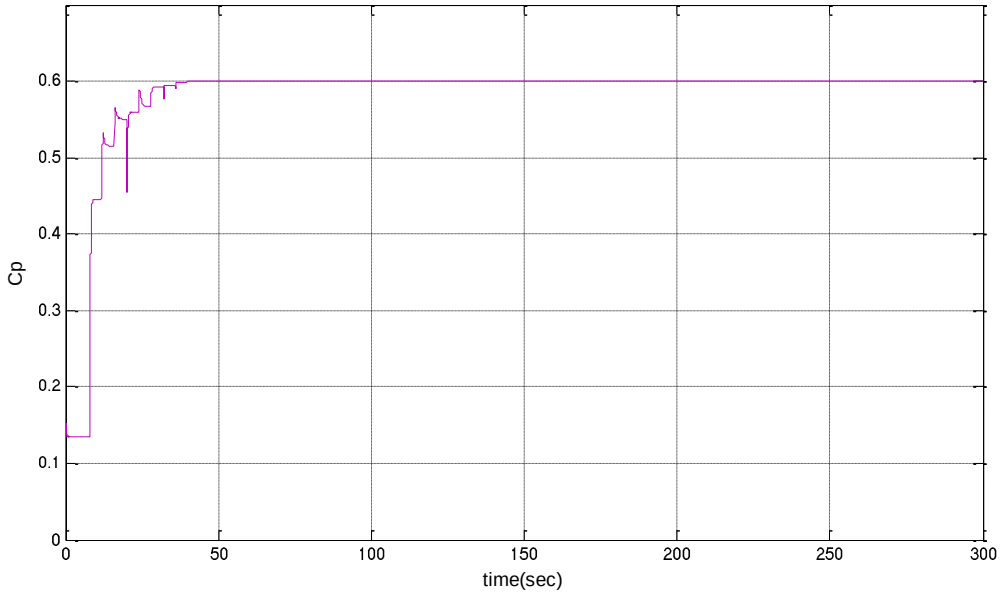


Figure 4.14 Cp

It can be seen that the system performs well using the modified SPSA method. The tip speed ratio and the blade pitch reach 4 and 2, respectively, at around 50 seconds. The rotor power coefficient also reaches to its maximum quickly. After tuning the parameters, the new values used in this modified SPSA algorithm are listed in Table 4.3.

Table 4.3 Parameters of SPSA method for simulation with both inner loop and outer loop

Parameter	Value	Description
p	2	Dimension of parameters in objective function
a	10	Related to step size a_k
A	1	Related to step size a_k
c	1	Related to step size c_k
α	1.1	Related to step size a_k
γ	3	Related to step size c_k

In this Chapter, simulations were conducted in MATLAB by using the dynamic model of the wind turbine mentioned in Chapter 2. From the results, it can be seen that SPSA together with the speed regulation control law introduced in Chapter 3 work well for the wind turbine. In Chapter 5, the SPSA method as well as the speed regulation control law will be implemented on MATLAB platform for FAST which is coded by The National Renewable Energy Laboratory (NREL). Additionally, the results of the SPSA algorithm will be compared with the classical gradient based extremum seeking algorithm.

Noticing that perturbation decreases as k increases, as a result, when wind speed changes, the value of k should be reset to restart the modified SPSA method. Figure 4.15 shows us the results with changing wind speed. The wind speed signal is set as a step command changing at the 64_{th} second from 10 m/s to 20 m/s, and in the SPSA block, the value of k is changed at the 64_{th} second. Other parts of the simulation diagram are the same as Figure 4.11. From Figure 4.15, it can be seen that rotor speed firstly converged to 1 maximizing C_p at 0.6, and then after the change of wind speed, it converged to the value of 2 which maximized C_p at its peak of 0.6.

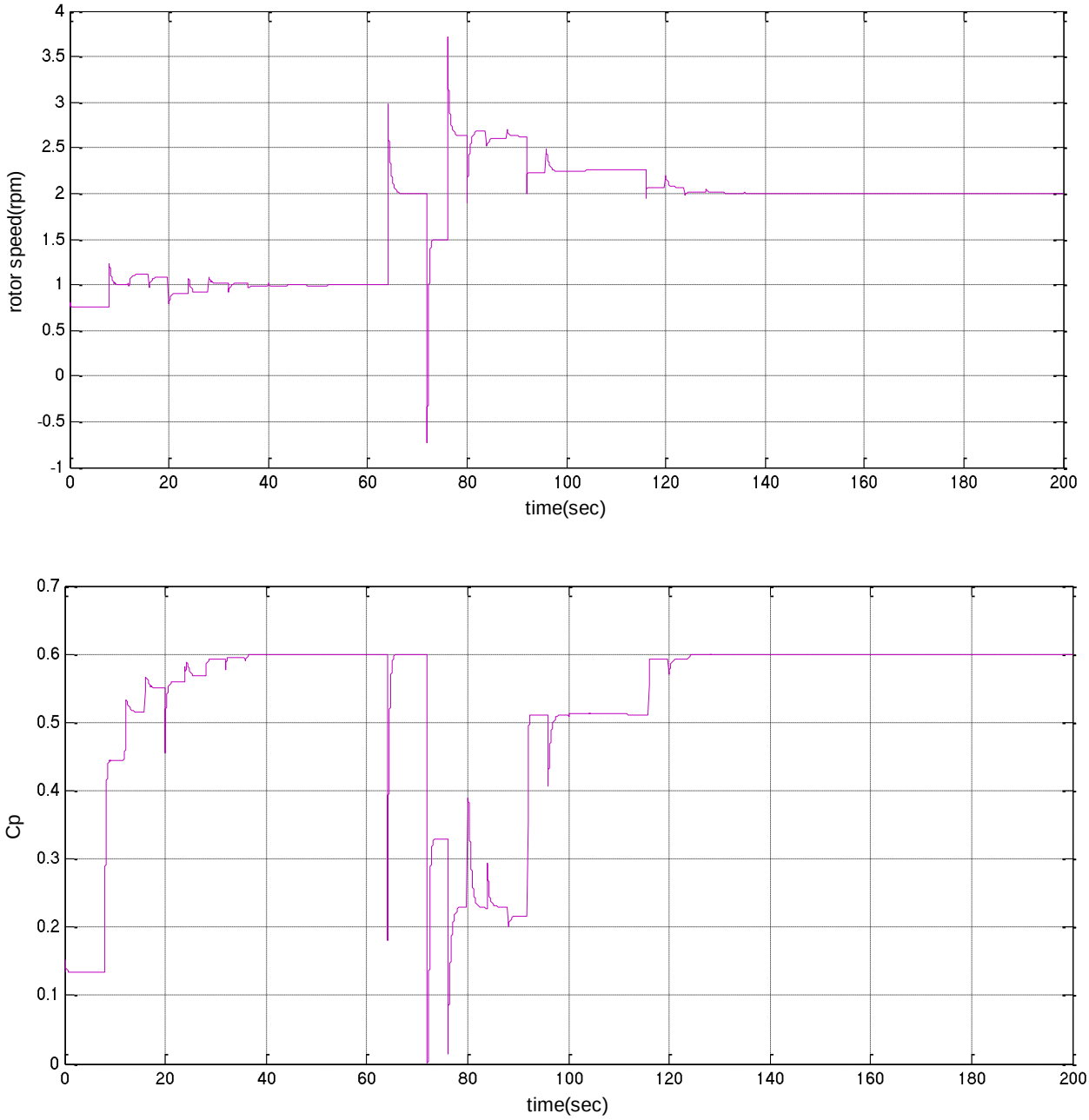


Figure 4.15 results of rotor speed and Cp with changing wind speed

Chapter 5 - SPSA in FAST

In this chapter simulations using FAST with the speed regulation controller and the SPSA method mentioned in Chapters 3 and 4 are conducted. Additionally, the results are compared with that of a typical classical gradient based algorithm.

FAST

The National Renewable Energy Laboratory (NREL) is a primary laboratory of renewable energy study in the United States. It has developed various computer aided engineering (CAE) tools for wind turbines. FAST, which is coded using FORTRAN, simulates a wind turbine's mechanical responses to various wind conditions. It can model both two and three bladed, horizontal axis wind turbines. It also provides some basic inputs for blade pitch commands, generator torque commands, and yaw commands. The newest version of FAST is v7.02 which can be seen in B. J. Jonkman, J. M. Jonkman (2013). In this study, the Simulink interface of FAST is used in MATLAB to model wind turbine and to implement the control.

As Simulink can incorporate FORTRAN via an S-Function block in MATLAB and FAST is coded with FORTRAN, the FAST subroutines are linked with MATLAB so that the FAST equations of motion can be used in an S-Function block which is able to be incorporated in a Simulink model. This introduces tremendous flexibility in the wind turbine controls implementation during simulation. In the Simulink environment, various control modules can be designed for use with the complete nonlinear aeroelastic wind turbine equations of motion in FAST. The equations of motion are formulated in the FAST S-function but solved using one of the Simulink solvers, namely fixed step fourth order Runge-Kutta.

In Figure 5.1, we can see that there are 3 inputs to the FAST Wind Turbine Block which are Generator Torque (Nm) and Power (W), Yaw position and Rate (rad/s), and Blade Pitch Angles (rad). In this paper, the Yaw control input is not used. We assume the nacelle is facing into the wind and the wind direction does not change. The details of the FAST Wind Turbine Block in Figure 5.1 are masked. FAST appears in MATLAB as a whole block with three inputs and specified outputs are shown in Figure 5.2.

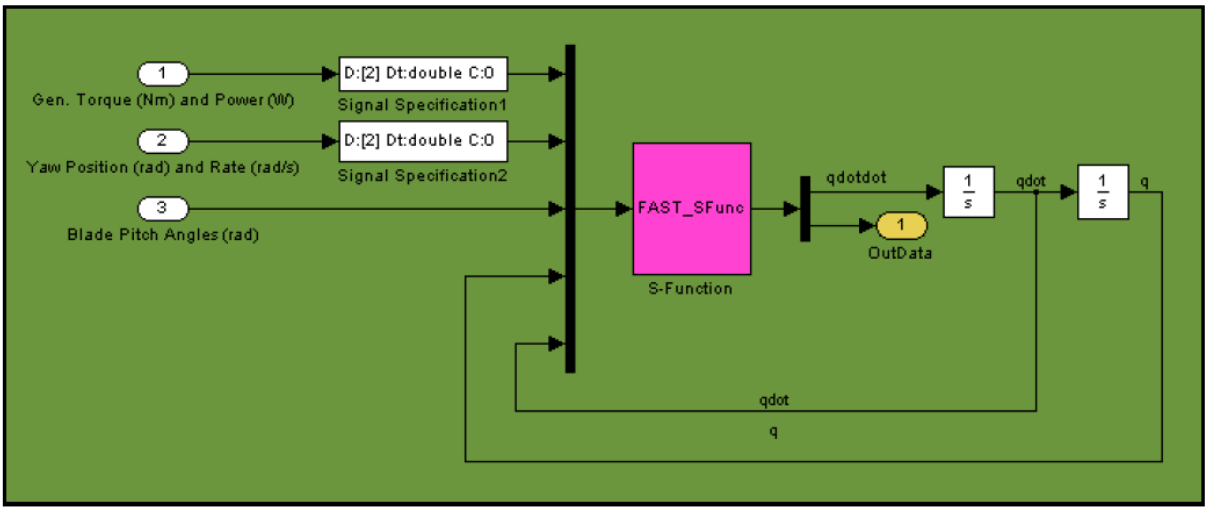


Figure 5.1 FAST Wind Turbine Block(B. J. Jonkman, J. M. Jonkman(2013))

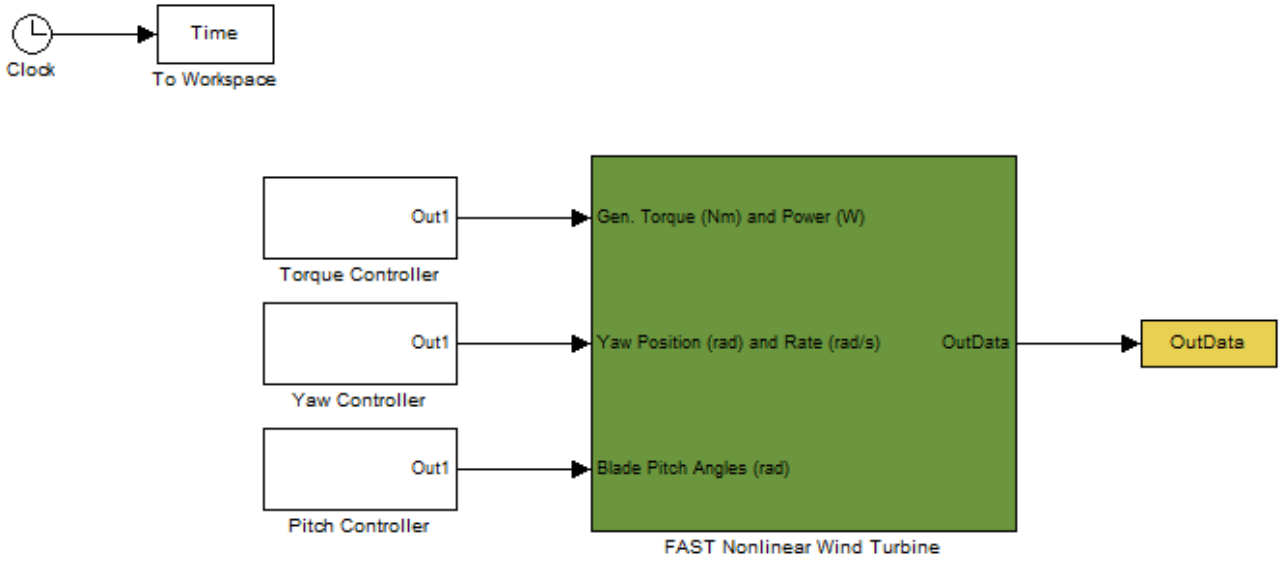
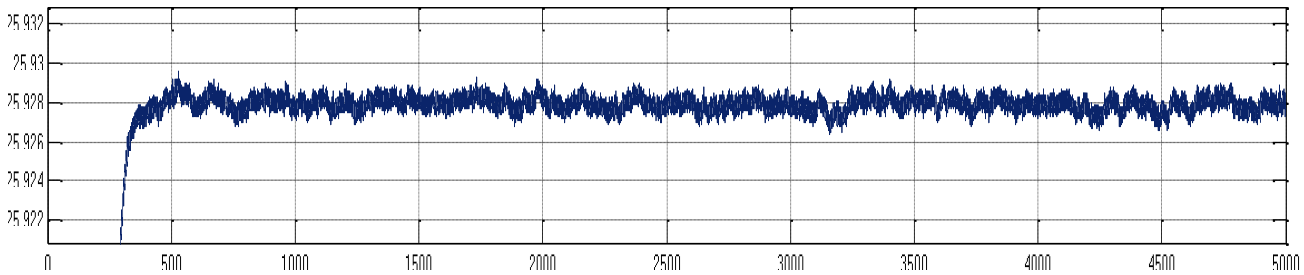
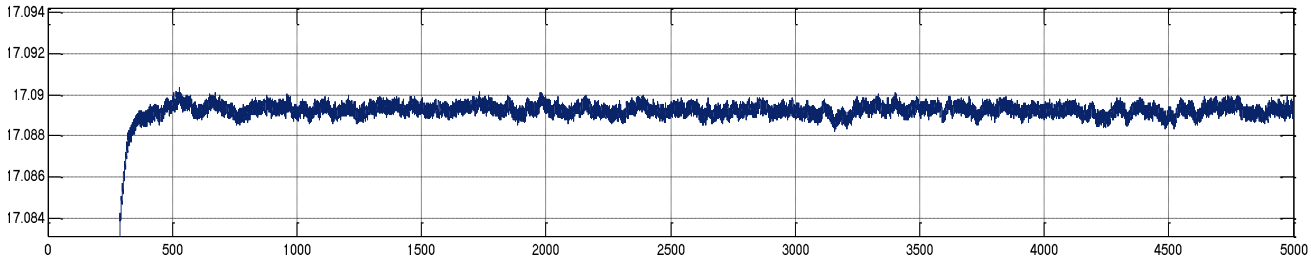


Figure 5.2 Configuration of FAST in MATLAB

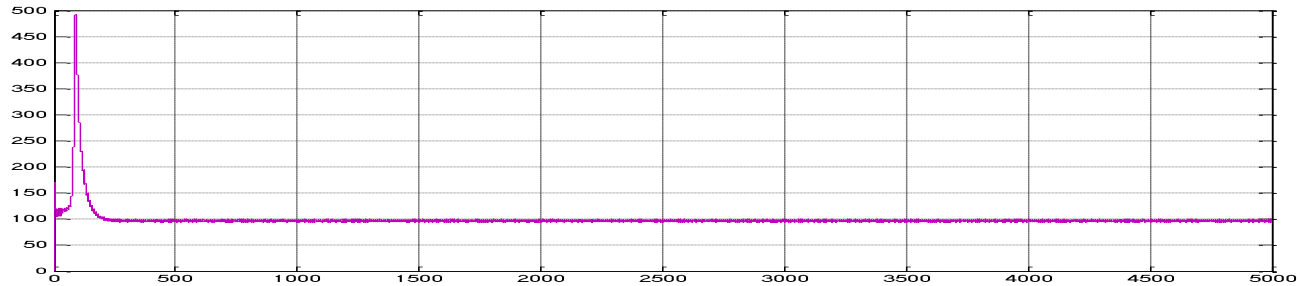
Setting parameters as Appendix C, when the input Generator Torque and Electrical Power are respectively 1000 Nm and 1000 W, Yaw control and blade pitch are both zero, the dynamic performance of the FAST Simulink Model is shown in Figure 5.3. The turbine being simulated is the NREL 5MW reference turbine [J. Jonkman et al. (2009)] in FAST.



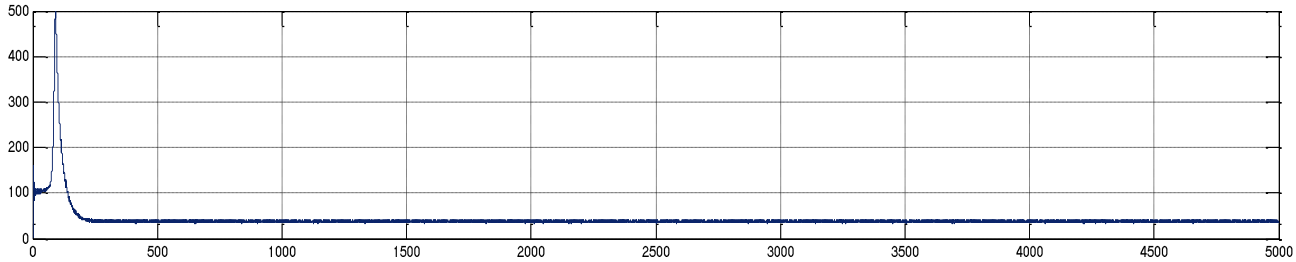
(a) Rotor speed (RPM)



(b) Tip speed ratio



(c) Rotor torque (kN-m)



(d) The sum of the blade root in-plane bending moments

Figure 5.3 Dynamic performance of 5MW turbine model in FAST

Simulations using SPSA with FAST

There are three main parts of this simulation which are the FAST block, the inner loop controller, and the outer loop SPSA controller. Parameters for each of these three parts need to be specified. In this paper, the NREL 5MW wind turbine [J. Jonkman et al. (2009)] is used. For consistency, parameters of the Simulink Model are specified as shown in Appendix C. Parameters of the inner and outer loops are specified as shown in Table 5.1 and Table 5.2.

Table 5.1 Parameters of the speed regulation

Parameter	Value	Unit	Description
J	43858348.0	Kg-m^2	Rotor Moment of inertia
R	63.0	m	Radius of blade
C_D	1	Kg-m^2/sec	Damping coefficient
ρ	1.275	Kg/m^3	Air density
k_s	10^7	-	Controller gain
β_c	20	-	Controller gain
α	0.4	-	Controller gain

Table 5.2 Parameters of the SPSA algorithm

Parameter	Value	Description
p	2	Dimension of parameters in objective function
a	10	Related to step size a_k
A	1	Related to step size a_k
c	20	Related to step size c_k
alpha	2	Related to step size a_k
gamma	3	Related to step size c_k
theta	[8 1]'	Initial value of the tip speed ratio and the blade pitch

After setting up the parameters, the whole wind turbine system with the nonlinear inner loop controller and the one measurement outer loop controller is shown in Figure 5.4. The green block is the nonlinear wind turbine model with the FAST s-Function inside. The torque controller is the inner loop nonlinear controller which is mentioned in Chapter 3. The outer_loop_control block is the one-measurement SPSA controller for extremum seeking based on the estimated rotor power coefficient produced by the inner loop controller. The pitch controller block contains a MATLAB block 1/(s+1) having a pole at -1. The details of these blocks and the codes can be found in Appendix C.

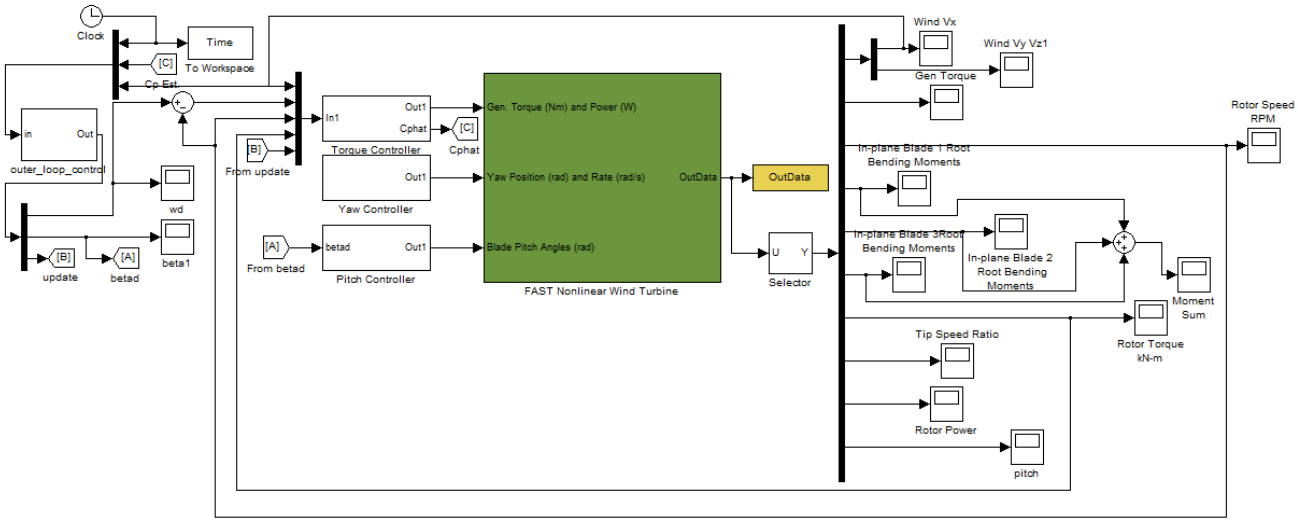
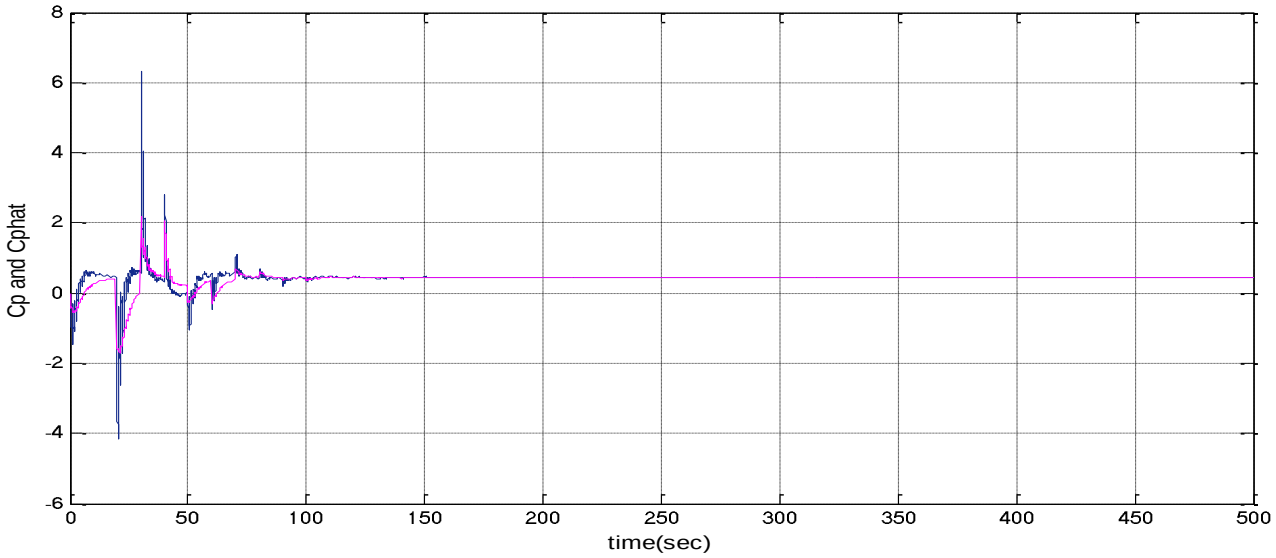


Figure 5.4 Simulation diagram

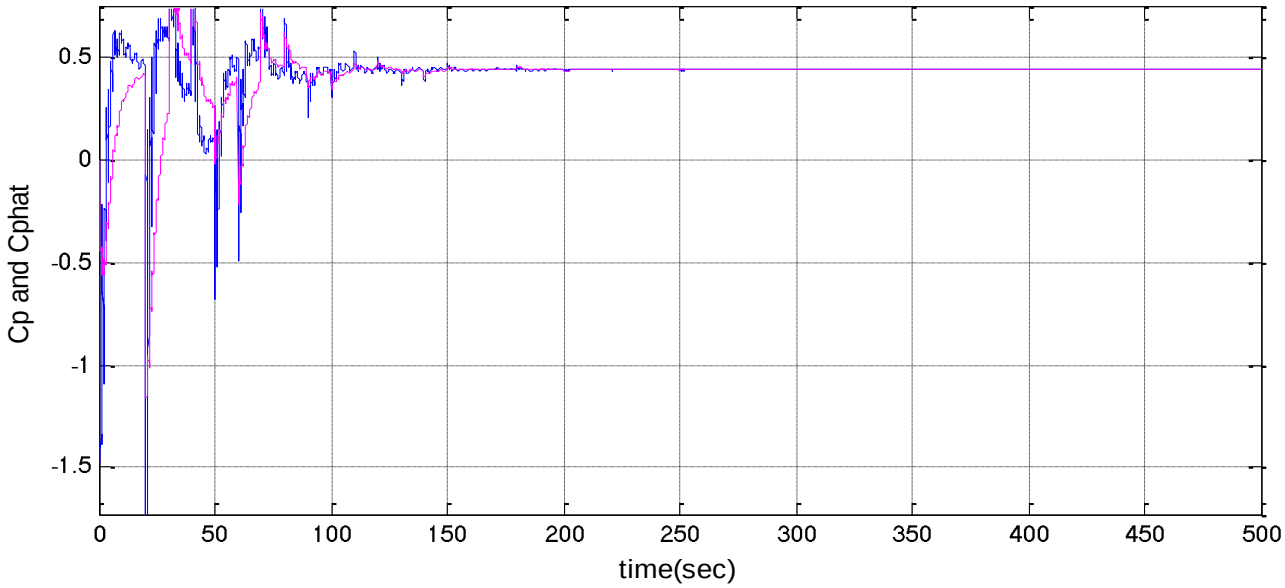
Not all outputs are needed for extremum seeking of the rotor power coefficient. Some are there for the convenience of debugging.

In FAST, the real Cp can be calculated from the rotor torque and speed outputs. Figure 5.5 shows the simulation results of Cp and the estimate of Cp. The purple curve is the estimated Cp and the blue curve is the real Cp calculated in FAST. We can see that, the estimated Cp follows the real Cp very well and it approaches a value of 0.46 which is very close to the ideal maximum Cp value given by J. Jonkman et al. (2009). The spikes in the plot come from the

perturbations of the modified SPSA method and inner loop control law. So in future work, these have to be considered. Figure 5.5 (b) is the exploded plot of Figure 5.5 (a).



(a)



(b)

Figure 5.5 Cp and Cphat

Figure 5.6 shows the plots of rotor speed (rpm) and Figure 5.7 shows the plot the tip speed ratio. It can be seen that rotor speed goes to 9.4 rpm in a stable manner in Figure 5.6 and the tip speed ratio goes to 6.2 in Figure 5.7. It can be easily verified that

$$\frac{\omega R}{v_{wind}} = \left(9.4 \times \frac{2\pi}{60}\right) \times \frac{63}{10} = 6.2.$$

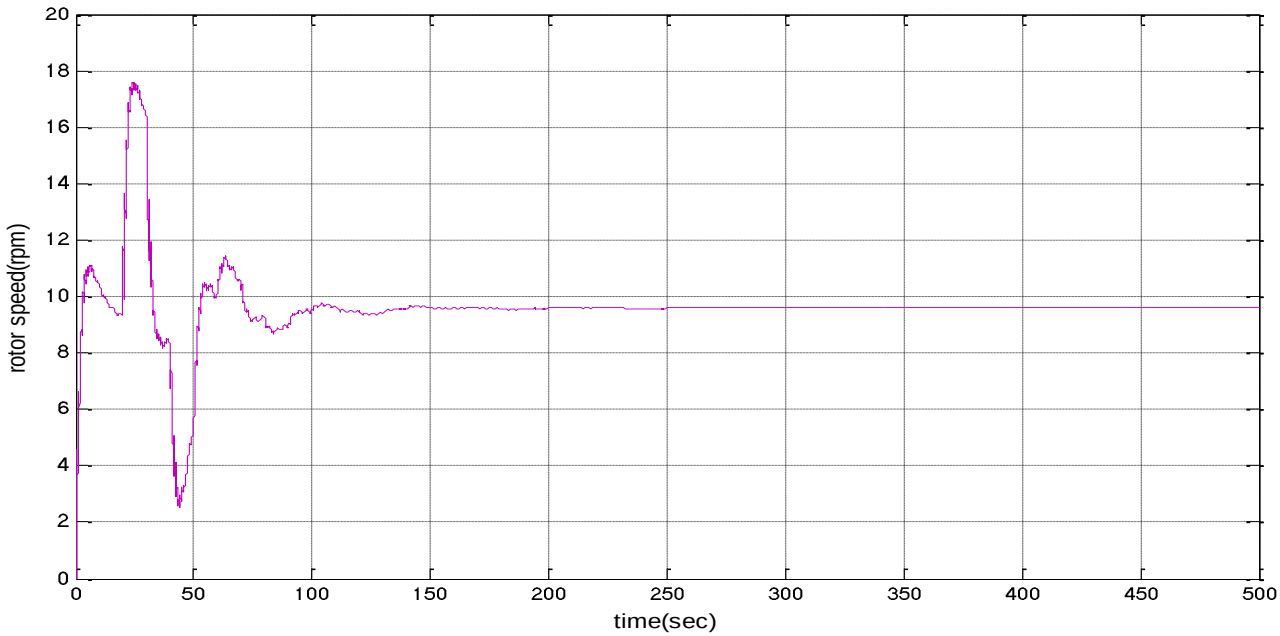


Figure 5.6 Rotor speed

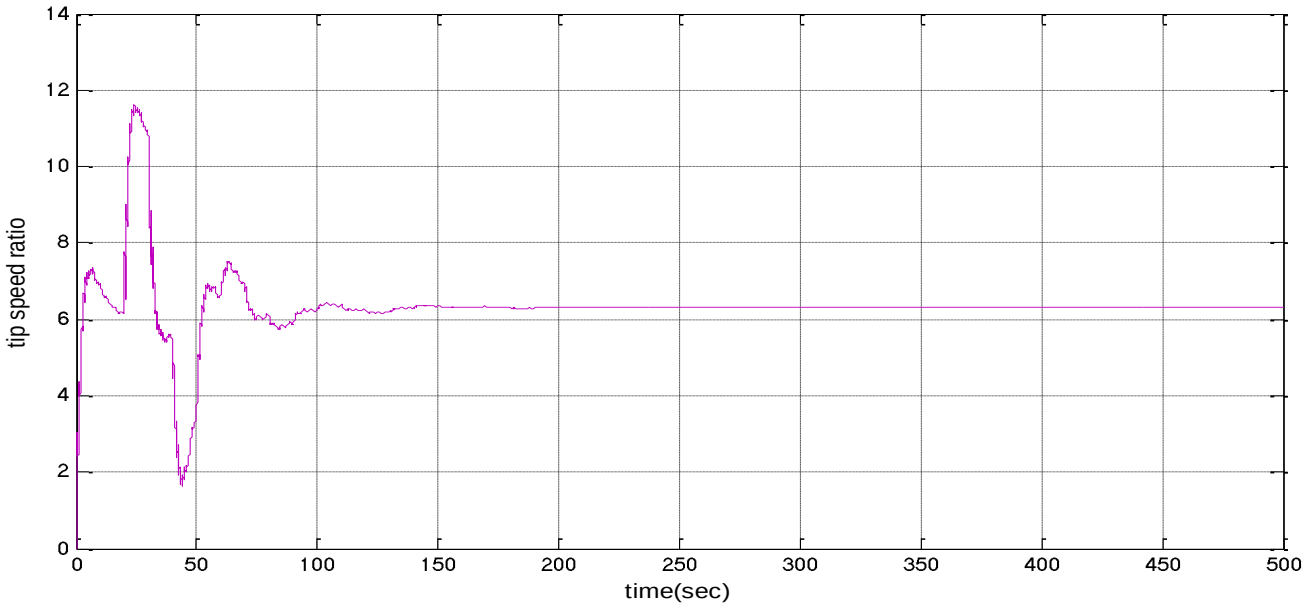


Figure 5.7 Tip speed ratio

Figure 5.8 shows the simulation output for the rotor torque which attains a value of 2.9 kN-m.

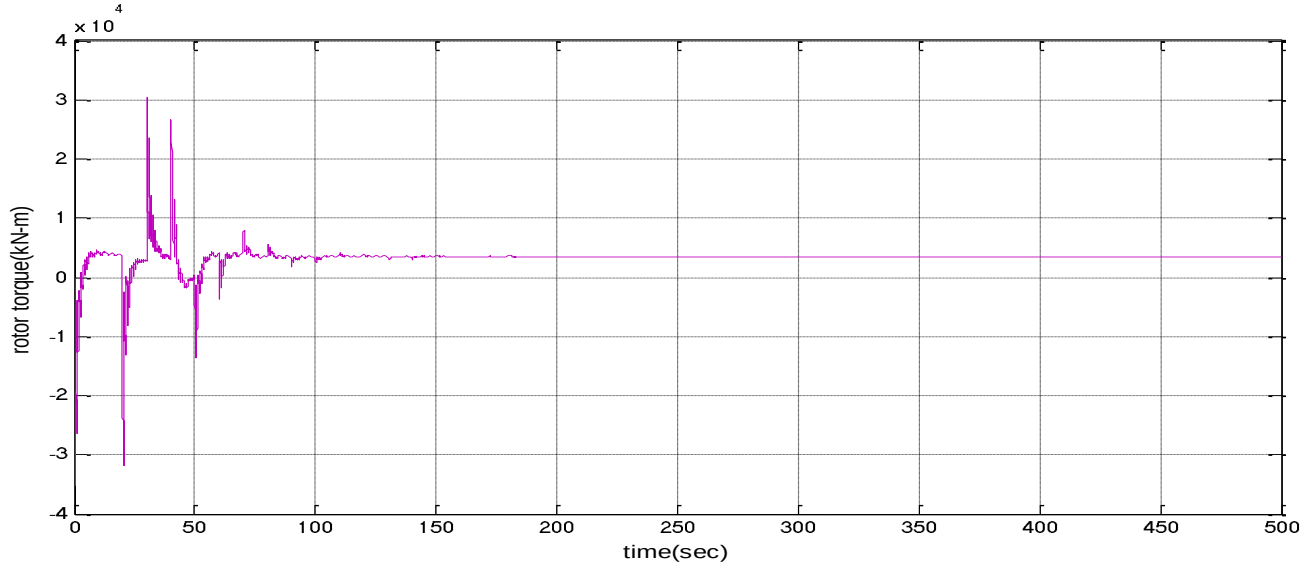


Figure 5.8 Rotor torque

Having obtained the result of the SPSA algorithm with FAST, a comparison can be made between the SPSA and a classical, gradient based optimization method, used as the outer loop

controller. Figure 5.9 is the simulation diagram of the classical, gradient based extremum seeking algorithm. This method is similar to the work presented by Hawkins et al. (2011).The torque controller block is used for speed regulation and Cp estimation. It has the same parameters as that of the SPSA method which are shown in Table 5.1. The pitch controller is also the same as that of the SPSA method. Find_Max block is for extremum seeking using the classical gradient based algorithm and its codes are in Appendix C.

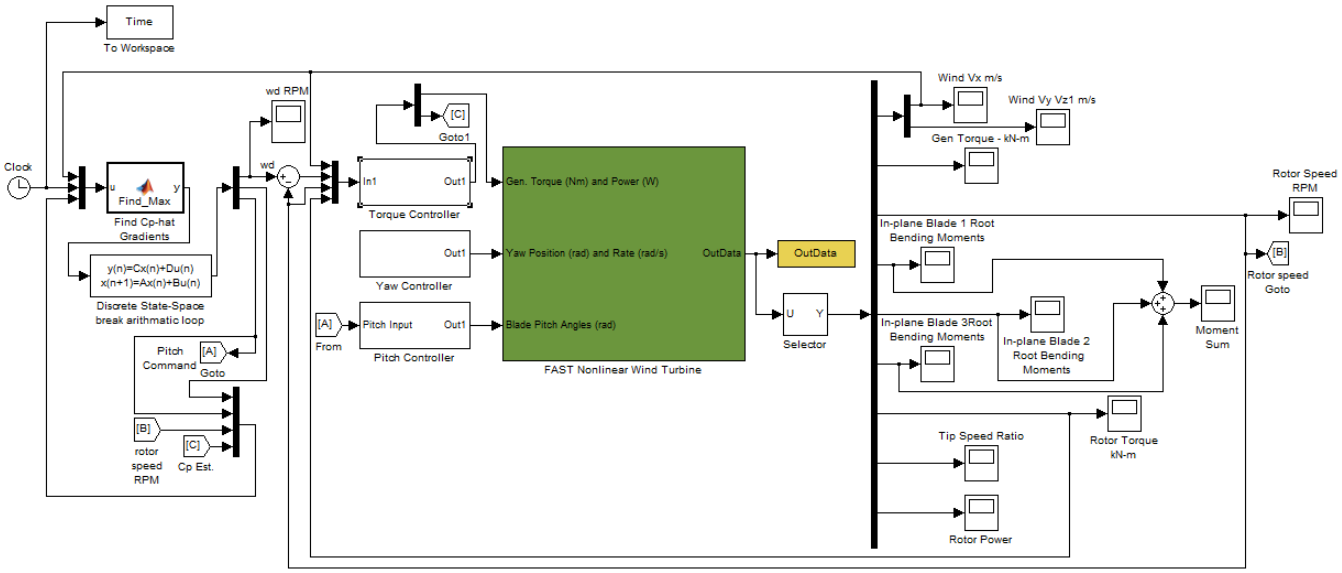


Figure 5.9 Simulation diagram using classical gradient based algorithm

Figure 5.10 shows that the estimate of Cp follows the real Cp well and they go to a value of 0.46 which is close the maximum value of the turbine given by J. Jonkman et al. (2009). Comparing Figure 5.5 and Figure 5.10, it can be seen that simulation using modified SPSA method reaches maximum Cp faster than that using the classical gradient method and has smaller oscillations.

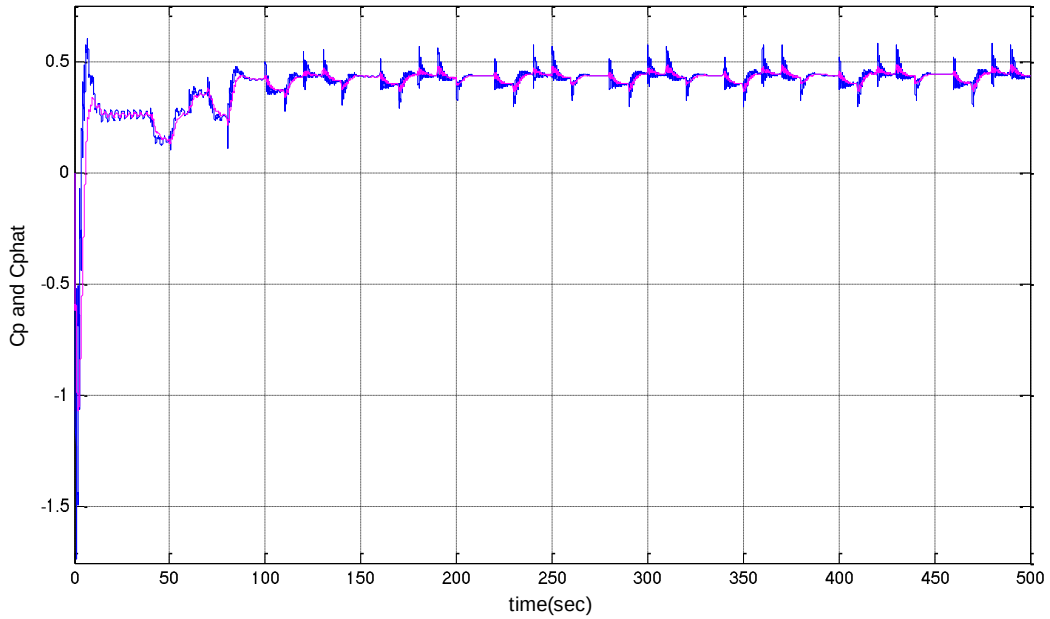


Figure 5.10 Cp and Cp estimate

Figure 5.11 and 5.12 show the plots of rotor speed and tip speed ratio of the classical method. From the plots, it can be seen that they are proportional to each other due to the constant wind velocity. And they are increasing slowly but have not reached a constant value in 500 seconds.

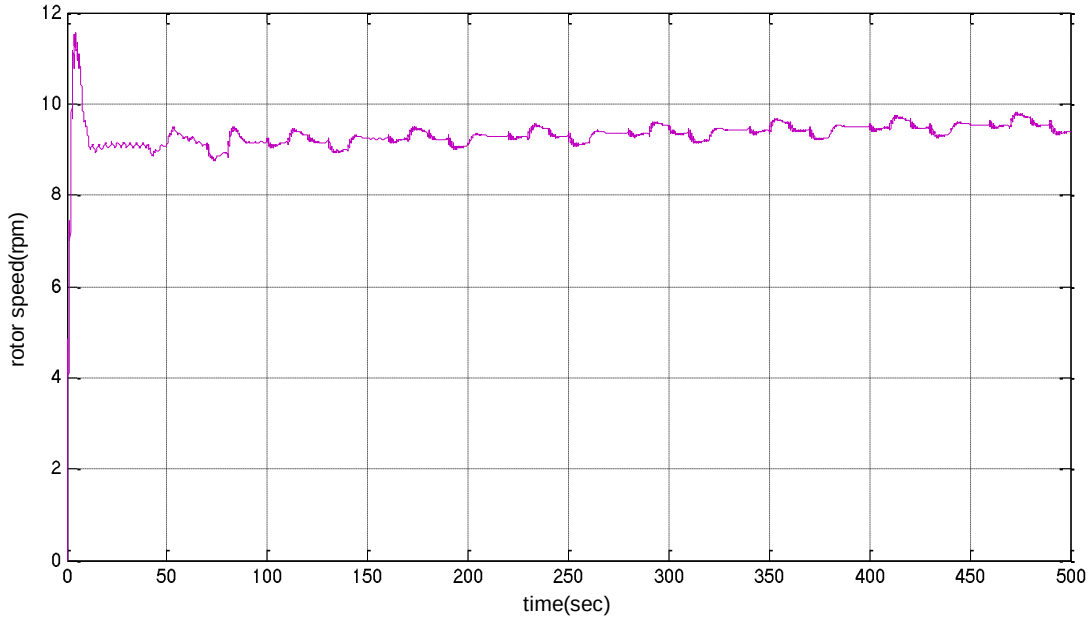


Figure 5.11 Rotor speed

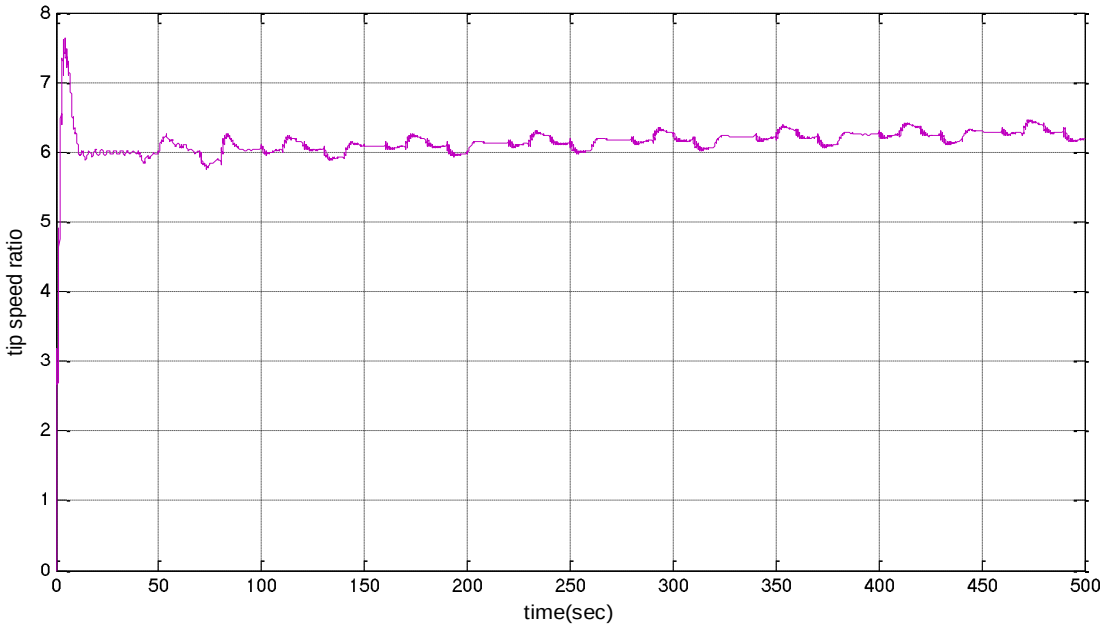
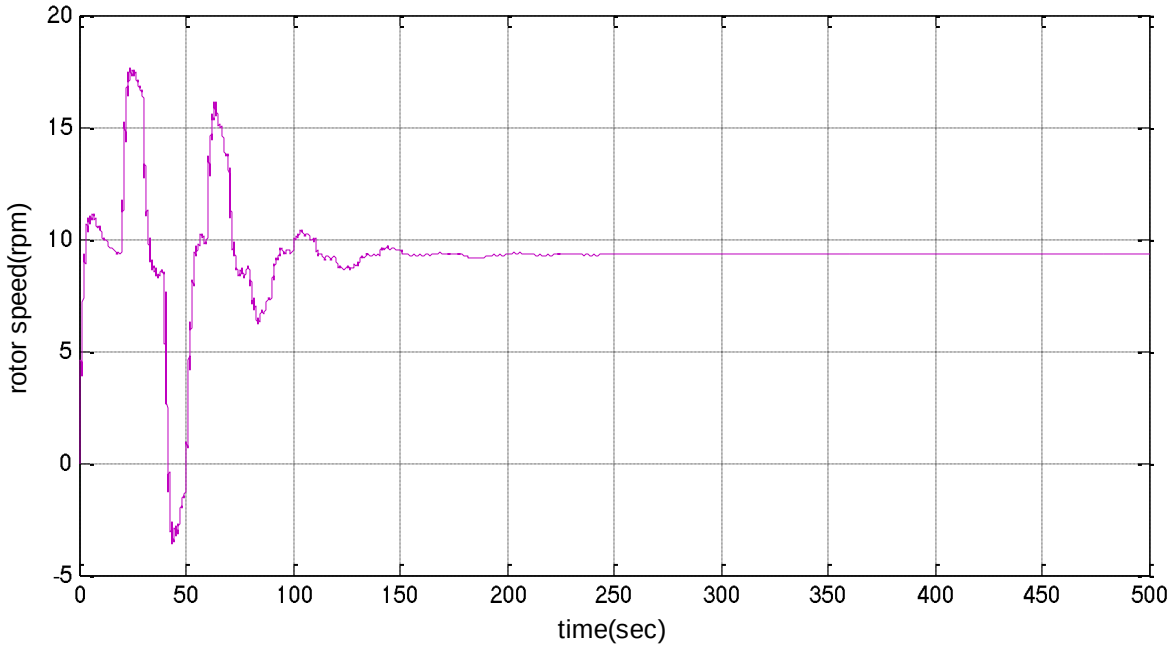


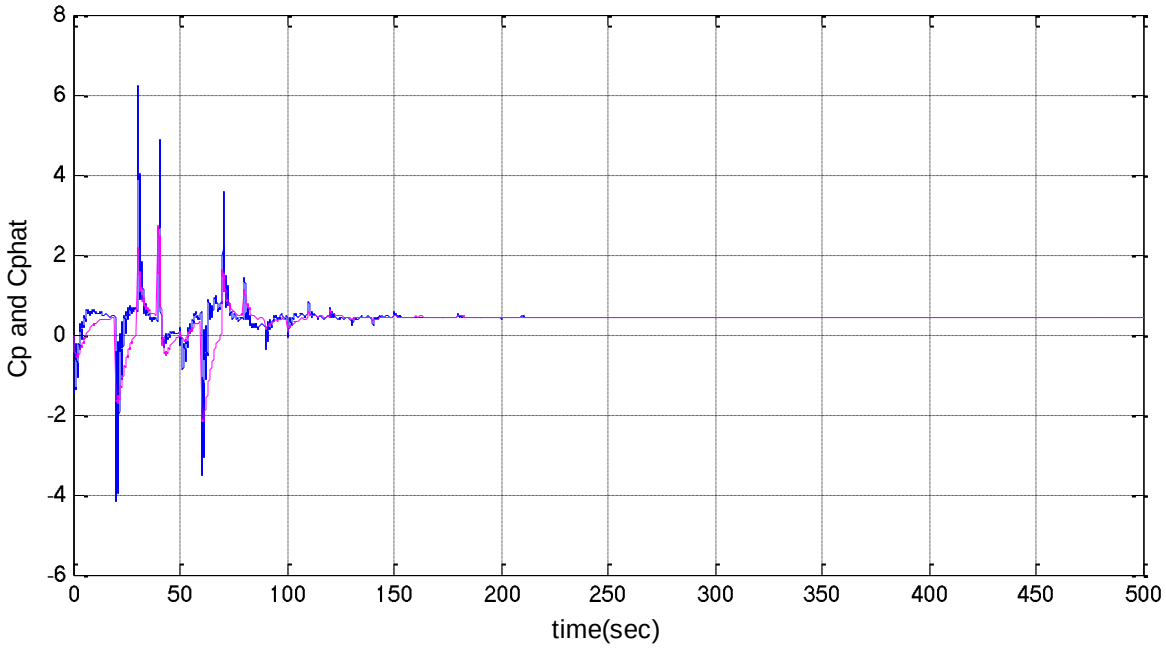
Figure 5.12 Tip speed ratio

The results of the SPSA method and the classical gradient based extremum seeking algorithm shows that the SPSA method has better performance in some aspects. Firstly, the SPSA method reaches the extremum faster than the classical algorithm. From the plots, it can be seen that the simulation with the SPSA method reaches the maximum Cp in less than 100 seconds, and the simulation using the classical method has not converged on the maximum Cp in 500 seconds. Secondly, the simulation of the classical method has oscillations when it gets close to the extremum because it has constant signed perturbations even when it is close to the maximum Cp. While the simulation of SPSA method does not have oscillations because the perturbation of the SPSA methods get smaller as it approaches the extremum. It will be close to zero when the turbine power coefficient is close enough to the maximum.

Because $c_n = \frac{c}{k^\gamma}$, we can change the c and γ to obtain different performances of the modified SPSA methods for seeking maximum C_p. When c increases or γ decreases, c_n increases. Figure 5.13 and Figure 5.14 show the results for the modified SPSA method with different c and γ. Comparing these results with Figure 5.5 and Figure 5.6, it can be seen that, perturbation increases when c increases or γ deceases. While because c_n also shows up in the denominator of (3.25), the gradient gets smaller and the simulations takes longer time to converge.



(a)



(b)

Figure 5.13 Result of modified SPSA method with $c = 40, \gamma = 3$

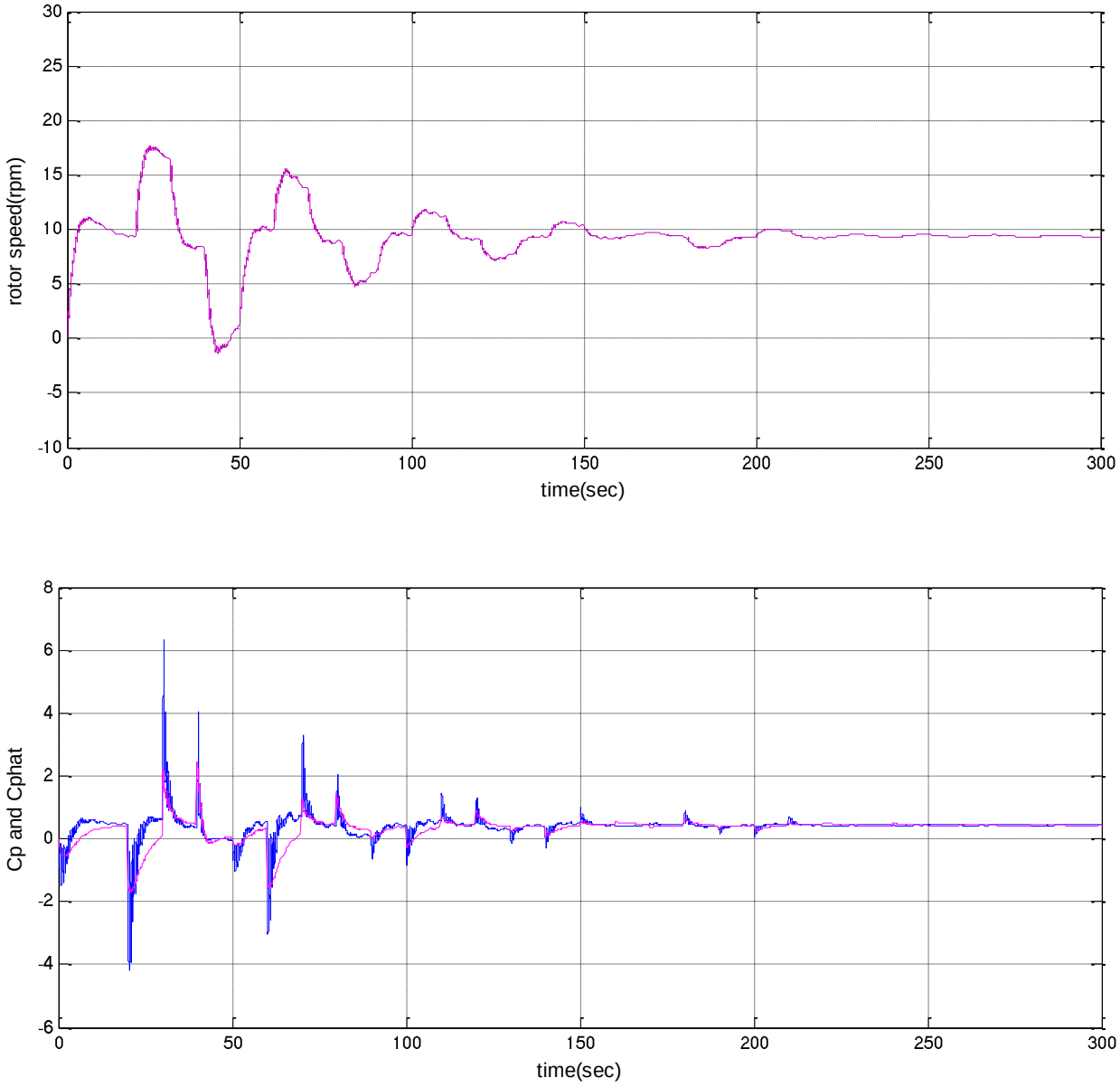


Figure 5.14 Result of modified SPSA method with $c = 10, \gamma = 1.5$

Chapter 6 - Conclusion and recommendation

In this thesis, a wind turbine system is modeled in two ways. One is the based on its dynamic model equation which is simpler and easier to use for checking the control algorithms. The other one is the FAST nonlinear model created by NREL that is much more complicated and closer to a real world wind turbine. A robust nonlinear controller derived by a Lyapunov direct method is used as the inner loop controller for speed regulation. This speed controller provides a very good estimate of the rotor power coefficient. The actual value of rotor power coefficient is not easily measured. The modified one measurement SPSA method has been applied to drive the whole wind turbine system to the maximum Cp.

From the simulation for both of the models, it can be seen that the controllers of the inner loop and outer loop performed well for seeking the extreme Cp value. From the results of the previous chapters, it can be seen that the inner loop controller regulates the rotor speed to the desired value accurately and quickly while the outer loop SPSA method drives the system to the maximum Cp value quickly. During the extreme seeking procedure, only one estimate value of the rotor power coefficient is needed for the SPSA method. No information of the wind turbine model was used for obtaining the maximum Cp value. Compared to the classical, gradient based extremum seeking methods, SPSA changes all of the important variables simultaneously. From the simulation results, the one measurement SPSA method makes the wind turbine system reach its extreme Cp value faster than the classical, gradient based extremum seeking method and the two step SPSA method.

Because the FAST model is very close to a real wind turbine, it is possible for the controller in this paper to be applied on a real wind turbine successfully. However, for using it on real wind turbine system, there are two more considerations that need to be addressed. One is the model of the wind. In this work, the wind has a steady value and direction. In the real world, the wind speed is not a constant, it is a time varying signal with many environmental variations. So a practical wind model should be used for testing in which the counter of the SPSA algorithm should be changed when Cp changes significantly due to the variation of the wind speed. When the wind speed changes, the algorithm step number denoted as k in the SPSA method will be reset as a relatively small number, otherwise the algorithm does not have large enough perturbations. The other consideration is about the saturation of the angular velocity for a wind

turbine. Boundaries, as that in Chapter 5 can be used to limit the output of the SPSA method for controlling the saturation of the torque controller. Additionally, a Saturated RISE feedback control is proposed by N. Fischer, Z. Kan and W. E. Dixon (2012) for Euler-Lagrange Systems. It might be useful for the wind turbine system. The last consideration is about the collective pitch. When using a non-constant wind model, the relative wind speed would differ for different blades. So, using individual blade pitch would be necessary for the wind turbine to have better performance.

References

1. Ariyur, Kartik B., and Miroslav Krstic. Real-time optimization by extremum-seeking control. Wiley. com, 2003.
2. M. J. Balas, Y. J. Lee, and L. Kendall (1998), “Disturbance Tracking Control Theory with Application to Horizontal Axis Wind Turbines,” Proc. AIAA/ASME Wind Energy Symp., pp. 95–99, Jan. 1998.
3. B. Beltran, T. Ahmed-Ali, M. E. H. Benbouzid (2006), “Sliding Mode Power Control of Variable-Speed Wind Energy Conversion Systems,” IEEE Transactions on Energy Conversion Vol. 23, No. 2, June 2006.
4. A. Betz (1926), “Wind Energy and its Exploitation by Windmills,” Gottingen: Van-denhoeck und Ruprccht, p. 64.
5. Z. Chen, E. Spooner (2001), “Grid Power Quality with Variable Speed Wind Turbines,” IEEE Transactions on Energy Conversion, Vol. 16, No. 2, June, 2001.
6. N. Fischer, Z. Kan, W. E. Dixon (2012), “Saturated RISE Feedback Control for Euler-Lagrange Systems”. American Control Conference, 2012.
7. V. Fthenakis, H. C. Kim.(2009) "Land use and electricity generation: A life-cycle analysis". *Renewable and Sustainable Energy Reviews* 13 (6–7): 1465.
8. M. C. Fu and S. D. Hill (1997), “Optimization of discrete event systems via simultaneous perturbation stochastic approximation,” IIE Trans., vol. 29, no. 3, pp. 233–243, Mar. 1997.
9. L. Gerencs´er (2005), SIAM Journal on Control and Optimization 44, 2123–2188 (2005).
10. P. Ghisbain (2009), “Application of a gradient-based algorithm to structural optimization”. Massachusetts Institute of Technology, 2009.
11. K. P. Green, “Wind subsidies’ pernicious effect on conventional energy investment”, aei-ideas.org,[Online].Available: http://www.aei-ideas.org/2012/09/wind-subsidies-pernicious-effect-on-conventional-energy-investment.
12. T. Hawkins, W. White, G. Hu, and F. D. Sahneh. "Region II wind power capture maximization using robust control and estimation with alternating gradient search." American Control Conference (ACC), 2011. IEEE, 2011.
13. M. M. Hand and M. J. Balas (2000). Systematic controller design methodology for variable-speed wind turbines. Wind Engineering, 24(3):169–187, 2000.
14. B.D. Heydon and S.D. Hill (2003),"Maximizing Target Damage Through Optimal Aimpoint Patterning,"A/AA 3rd Biennial National Forum on Weapon System Effectiveness, 18-20 November 2003, Seal Beach, CA (simulation-based optimization to aid in targeting and minimizing collateral damage) (distribution restricted to U.S. government agencies and their contractors).
15. F. Iov, A.D. Hansen, P. Sorensen, F. Blaabjerg (2004), “Wind Turbine Blockset in Matlab/Simulink”, tech. rep. Aalborg University, Denmark, 2004.
16. K. E. Johnson, L. J. Fingersh, M. J. Balas, L. Y. Pao (2004), “Methods for Increasing Region 2 Power Capture on a Variable-Speed Wind Turbine,” Transactions of the ASME Vol. 126, November, 2004.

17. K. E. Johnson, L. Y. Pao, M. J. Balas, and L. J. Fingersh (2006). Control of variable-speed wind turbines: Standard and adaptive techniques for maximizing energy capture. IEEE Control Systems Magazine,26(3):70–81, June 2006.
18. J. Jonkman, S. Butterfield, W. Musial, and G. Scott (2009), "Definition of a 5-MW Reference Wind Turbine for Offshore System Development". February 2009.
19. B. J. Jonkman, J. M. Jonkman (2013),"Addendum to the User's Guides for FAST, A2AD, and AeroDyn Released March 2010 – February 2013 ALPHA VERSION". February 2013.
20. L. Kroldrup (2010). "Gains in Global Wind Capacity Reported" *Green Inc.*, February 15, 2010.
21. J. H. Laks, L. Y. Pao, and A. D (2009). "Wright. Control of wind turbines: Past, present, and future". In IEEE Proc. Amer. Control Conf., pages 2096–2103, St. Louis, MO, 2009.
22. C. Manzie, and M. Krstic (2009). "Extremum seeking with stochastic perturbations." Automatic Control, IEEE Transactions on 54.3 (2009): 580-585
23. A. Merabet , J. Thongam and J. Gu (2011). "Torque and Pitch Angle Control for Variable Speed Wind Turbines in All Operating Regimes". Environment and Electrical Engineering (EEEIC), 10th International Conference.
24. A. Morales (2012). "Wind Power Market Rose to 41 Gigawatts in 2011, Led by China". *Bloomberg*, February 7, 2012.
25. L. Y. Pao and K. E. Johnson (2009), "A Tutorial on the Dynamics and Control of Wind Turbines and Wind Farms," Proc. Amer. Ctrl. Conf., June 2009.
26. C. Ricketts, "Offshore wind farm provides shelter for some species", earthtimes.org,[Online]. Available: http://www.earthtimes.org/energy/offshore-wind-farm-shelter-species/1227.
27. P. Sennekamp (2011),"17 EU countries planning massive offshore wind power". *ROV world*, 30 November 2011. Accessed: 10 December 2011.
28. M. Singh and S. Santoso (2011), "Dynamic Models for Wind Turbines and Wind Power Plants". NREL/SR-5500-52780 , October 2011
29. Y.D. Song, B. Dhinakaran, X.Y. Bao (2000), "Variable speed control of wind turbines using nonlinear and adaptive algorithms," Journal of Wind Engineering and Industrial Aerodynamics 85 (2000) 293-308.
30. J. C. Spall (1998). "An overview of the simultaneous perturbation method for efficient optimization." Johns Hopkins APL Technical Digest 19.4 (1998): 482-492.
31. J. C. Spall and D. C. Chin (1997). "Traffic-responsive signal timing for system-wide traffic control." Transportation Research Part C: Emerging Technologies 5.3-4 (1997): 153-163.
32. K. Stol and M. J. Balas (2002). Periodic disturbance accommodating control for speed regulation of wind turbines. In Proc. AIAA/ASME Wind Energy Symp., pages 310–320, Reno, NV, 2002.
33. J. Wilkes (2012). "The European offshore wind industry key 2011 trends and statistics", *European Wind Energy Association*, January 2012. Accessed: 26 March 2012.

34. A. D. Wright and L. J. Fingersh (2008), “Advanced Control Design for Wind Turbines Part I: Control Design, Implementation, and Initial Tests,” NREL Report No. TP-500-42437, National Renewable Energy Laboratory, March 2008.

35. B. Xian, D. M. Dawson, M.S. de Queiroz, J. Chen (2004), “A Continuous Asymptotic Tracking Control Strategy for Uncertain Nonlinear Systems,” IEEE Transactions on Automatic Control Vol. 49, No. 7, July 2004.

36. Climate and Energy. “Energy Resources & Some Alternatives”. energysustained.com.[Online]. Available: http://www.energysustained.com/energy_resources.htm.

37. Environmental and Energy Study Institute (October 2010). "Offshore Wind Energy".

38. GE,Products & Services. Ge-energy.com.

39. Global wind energy council,“Wind power”, wikipedia.com,[Online].Available: http://en.wikipedia.org/wiki/Wind_power.

40. “Greenland’s melting icebergs”, *news.yahoo.com*, [Online]. Available: http://news.yahoo.com/photos/greenland-s-melting-icebergs-slideshow.

41. GWEC,”Global wind energy markets continue to boom – 2006 another record year”. Gwec.net.

42. REN21 (2011). "Renewables 2011: Global Status Report". p. 15.

43. “Technical Specs of Common Wind Turbine Models”. AWEO.org.

44. “Wind turbines introduction”,Mechanical Engineering, Boston University. Available: http://people.bu.edu/noahb/files/wind_turbine_main.pdf.

Appendix A - MATLAB codes of SPSA algorithm in Chapter 3

In Chapter 3, the SPSA algorithm was introduced and some MATLAB were used to test the SPSA method. The following are codes of the SPSA algorithm and the custom objective function.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The SPSA algorithm
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear;
%initialization
theta=[0.0 4]';%lambda,beta
n=500;
% thetaplot=zeros(n,2);
p=2;
a=5;
A=1;
c=20;%converge speed,the smaller the faster
alpha=0.2;
gamma=5;%converge speed,the larger the faster
%loop
for k=1:n
    ak=a/(k+A)^alpha;
    ck=c/k^(1*gamma);
    delta=2*round(rand(p,1))-1;
    thetaplus=theta+ck*delta;
    thetaminus=theta-ck*delta;
    yplus=loss(thetaplus);
    yminus=loss(thetaminus);
    ghat=(yplus-yminus)./(2*ck*delta);
    theta=theta-ak*ghat;
    thetaa(k,:)=theta;
    y(3*k-2)=yplus;
    y(3*k-1)=yminus;
    y(3*k)=loss(theta);
end
theta;
k=1:1:n;
figure(1);
length(k);
length(thetaa(:,1))
plot(k,thetaa(:,1));
grid;
figure(2);
plot(k,thetaa(:,2));
grid;
kk=1:1:3*n;
figure(3);
plot(kk,y);
grid;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The custom objective function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function y=loss(x)
lambdaex = 6;
betaex = 2;
y = -0.6*exp(-0.03*(x(1)-lambdaex)^2)*exp(-0.03*(x(2)-betaex)^2);
end
```

Appendix B - Simulink Embedded Code and Subsystem Blocks in Chapter 4

In Chapter 4, Figure 4.2 shows the diagram of the inner loop used to simulate speed regulation of the wind turbine, Figure 4.4 and 4.7show the diagram of the outer loop use to simulate the original SPSA method and the modified SPSA method for extremum seeking, and Figure 4.11 shows the simulation combining both the inner loop and the outer loop. This appendix will show in detail all of the information contained within simulations.

Inner loop simulation

This section will introduce the details of the subsystems in Figure 4.2 as well as the embedded codes used in those blocks.

Figure B.1 shows the block diagram of the controller subsystem in Figure 4.2. The inputs of this subsystem are desired angular velocity, actual angular velocity of the turbine and an update signal which is used to update the desired angular velocity. Figure 4.2 is just for the inner loop, so the update signal is set as constant zero. The difference between desired angular velocity and actual angular velocity is the error. The input ‘mu’ is the term about error and sign of error in the control law which is calculated by integrating its time derivative. The outputs of this subsystem are the control signal, f_hat, and the error.

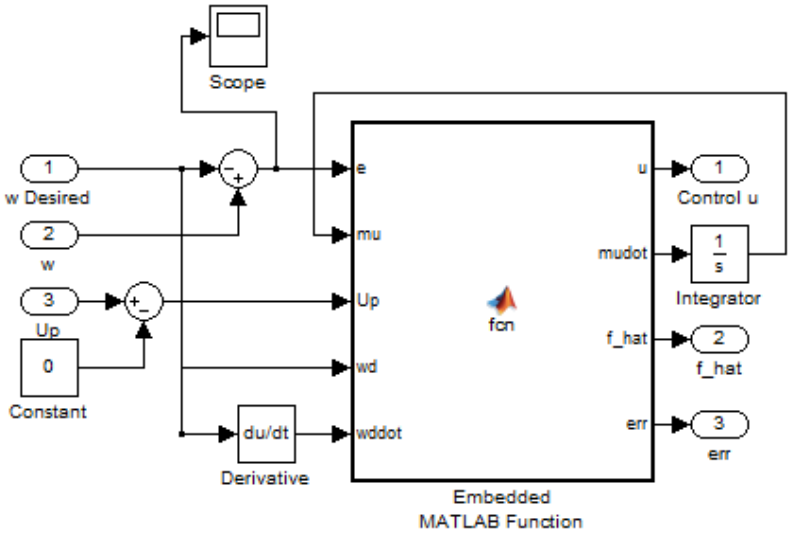


Figure B.6.1 Block diagram of controller subsystem in Figure 4.2

The following MATLAB code is the custom function that is inside the Embedded MATLAB Function.

```
function [u,mudot,f_hat,err] = fcn(e,mu,Up,wd,wddot)
J=100000;
persistent e0 wdk1
k1=10000000; k2=2; a1=3;
if isempty(e0)
    e0=e;
    wdk1=wd;
end
if Up==1
    e0=wd-wdk1;
    wdk1=wd;
end
f_hat=(k1+1)*(e-e0)+mu;
mudot=(k1+1)*a1*e+k2*sign(e);
w=e+wd;
err=e;
Cd=1;
u=-J*wddot-J*a1*e+f_hat;
%///satuation
umax=10e10; umin=-10e10;
if u>umax
    u=umax;
end
if u<umin
    u=umin;
end
%satuation//
wd_k1=wd;
end
```

Figure B.2 shows the block diagram of the variables subsystem. This subsystem uses wind speed, actual angular velocity, and blade pitch as inputs to calculate tip speed ratio (lambda), actual power coefficient (Cp), as well as the aerodynamic torque of the wind. Here, Cp is calculated by the theoretical equation. Cq is a variable for calculating Cp which equals to Cp/γ. The embedded MATLAB code here is as follows.

```
function Cq  = fcn(lambda,beta)
Cq=0.6*exp(-0.3*(lambda-4)^2)*exp(-0.3*(beta-2)^2)/lambda;
```

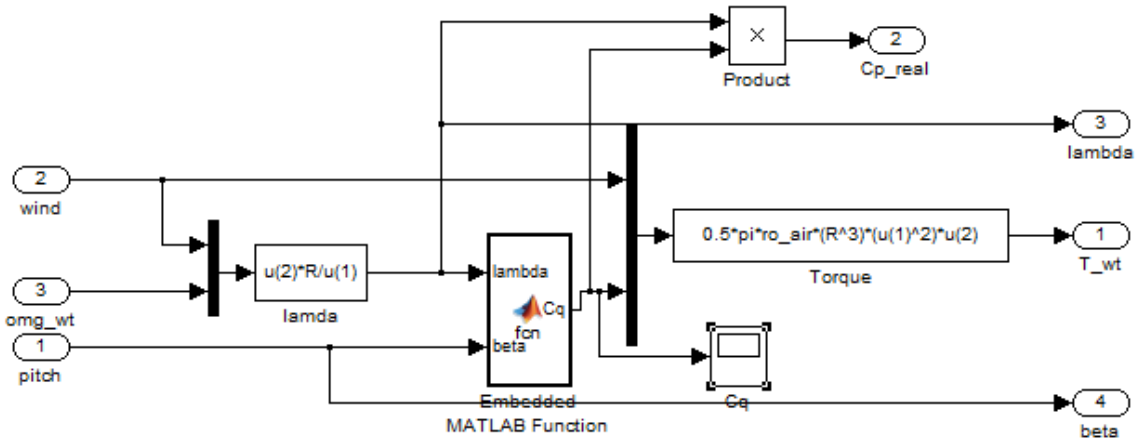


Figure B.6.2 Block diagram of variables subsystem in Figure 4.2

Figure B.3 shows the block diagram of the plant subsystem which is considered as a wind turbine plant in Figure 4.2. It uses the dynamic equation of a wind turbine mentioned in Chapter 2.

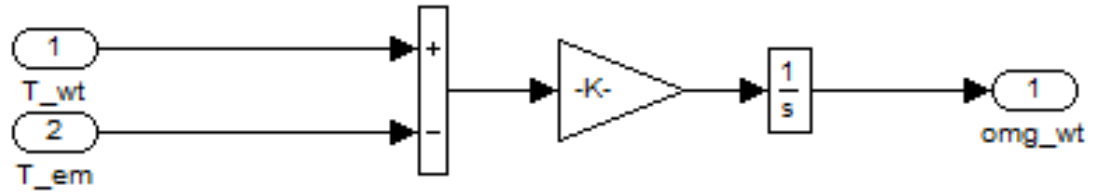


Figure B.6.3 Block diagram of the plant subsystem in Figure 4.2

Outer loop simulation

This section will introduce the details of the subsystems in Figure 4.4 and 4.7 which are Simulink diagram of outer loop using original SPSA method and modified SPSA algorithm. There are two blocks in both Figure 4.4 and Figure 4.7 which are the SPSA block and the plant block. The plant block, which is considered as a wind turbine plant, contains the dynamic equation of the wind turbine. Its block diagram is shown as Figure B.4.

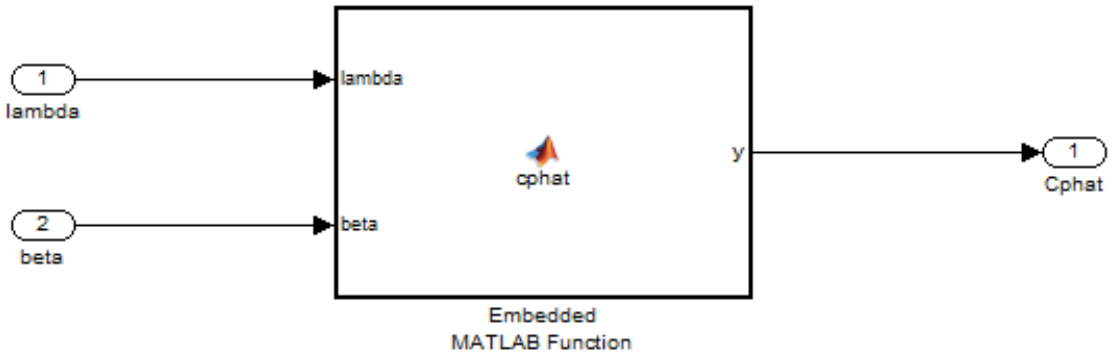


Figure B.6.4 Block diagram of plant subsystem in Figure 4.4

Its embedded codes are as follows. Its global extremum is at (4,2).

```
function y=cphat(lambda,beta)
lambdaex = 4;
betaex = 2;
y = -0.6*exp(-0.03*(lambda-lambdaex)^2)*exp(-0.03*(beta-betaex)^2);
end
```

Figure B.5 is the block diagram of the SPSA subsystem in Figure 4.4. It takes in the time signal and the value of Cp getting from the plant to calculate the path for the plant to reach the extremum using the original SPSA method. The values of Cp here is the actual values. The outputs, tip speed ratio and blade pitch, are the desired ones for next step towards the extremum.

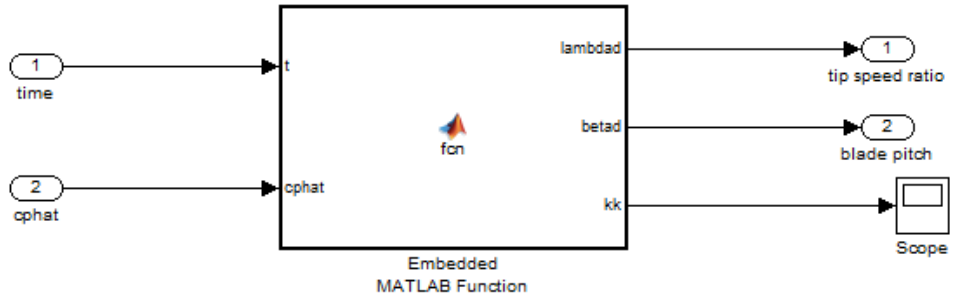


Figure B.6.5 Block diagram of the SPSA subsystem in Figure 4.4

The following codes is the embedded MATLAB function in this block.

```
function [lambdad, betad,kk] = fcn(t,cphat)
%initialization
thetaini=[0.0 4]';%lambda,beta
lambdaini=thetaini(1);
betaini=thetaini(2);
n=30000;
% thetaplot=zeros(n,2);
p=2;
a=5;
A=1;
c=20;%converge speed
alpha=0.1;
gamma=3;
lambdaex = 4;%unused
betaex = 2;%unused
%loop
persistent k theta lambdaout betaout ktk1 ki kik1 yplus yminus ak ck delta
thetaplus thetaminus ghat;
if isempty(k)
    ki=0;
    kik1=0;
    ktk1=0;
    k=0;
    theta=thetaini;
    lambdaout=lambdaini;
    betaout=betaini;
    yplus=0;
    yminus=0;
    ak=0;
    ck=0;
    delta=[0 0]';
    thetaplus=thetaini;
    thetaminus=thetaini;
    ghat=[0 0]';
else
    k=floor(t/6);
    ki=floor((t-6*k)/2);
end


if k>0&&(ktk1-k)<0
        ak=a/(k+A)^alpha;
        ck=c/k^(2*gamma);
        delta=2*round(rand(p,1))-1;
        thetaplus=theta+ck*delta;
        thetaminus=theta-ck*delta;
        lambdaout=thetaplus(1);
        betaout=thetaplus(2);
end
if k>0&&ki==1&&(kik1-ki)<0
    yplus=cphat;
    lambdaout=thetaminus(1);
    betaout=thetaminus(2);
end
if k>0&&ki==2&&(kik1-ki)<0
    yminus=cphat;
```

```
    ghat=(yplus-yminus)./(2*ck*delta);
    theta=theta-ak*ghat;
    lambdaout=theta(1);
    betaout=theta(2);
end
ktk1=k;
kik1=ki;
betad=betaout;
lambdad=lambdaout;
kk=k;
```

Figure 4.7 is the outer loop diagram for the modified SPSA method. The block diagram of the subsystems are completely the same as that of Figure 4.4. The only difference is that it uses the modified SPSA algorithm as the MATLAB function in the SPSA subsystem of Figure 4.7. The codes are as follows.

```
function [lambdad, betad,kk] = fcn(t,cphat)
%initialization
thetaini=[0 4]';%lambda,beta
lambdaini=thetaini(1);
betaini=thetaini(2);
n=30000;
p=2;
a=5;
A=1;
c=20;%converge speed
alpha=0.1;
gamma=3;
persistent k theta lambdaout betaout ktk1 ki kik1 yplus yori ak ck delta
thetaplus thetaminus ghat;
if isempty(k)
    ki=0;
    kik1=0;
    ktk1=0;
    k=0;
    theta=thetaini;
    lambdaout=lambdaini;
    betaout=betaini;
    yplus=0;
    yori=0;
    ak=0;
    ck=0;
    delta=[0 0]';
    thetaplus=thetaini;
    thetaminus=thetaini;
    ghat=[0 0]';
else
    k=floor(t/4);
    ki=floor((t-4*k)/2);
end
if k>0&&(ktk1-k)<0
```

```
        yori=cphat;
        ak=a/(k+A)^alpha;
        ck=c/k^(1*gamma);
        delta=2*round(rand(p,1))-1;
        thetaplus=theta+ck*delta;
        lambdaout=thetaplus(1);
        betaout=thetaplus(2);
end
if k>0&&ki==1&&(kik1-ki)<0
    yplus=cphat;
    ghat=(yplus-yori)./(ck*delta);
    theta=theta-ak*ghat;
    lambdaout=theta(1);
    betaout=theta(2);
end
ktk1=k;
kik1=ki;
betad=betaout;
lambdad=lambdaout;
kk=k;
```

Simulation combining both inner and outer loops

This section describes the details of the subsystems in Figure 4.11 which is the Simulink diagram combining both the inner and outer loops using the modified SPSA algorithm. In Figure 4.11, the variables subsystem and the plant subsystem are exactly the same as that of the inner loop which has been introduced in the first section in Appendix B. The speed regulation has the same block diagram as that of the inner loop. But since there is outer loop here, the update signal is no longer zero. The update signal has a value of one when the inner loop is stable at previous desired value of lambda and beta. The following code is in the embedded MATLAB function of the speed regulation subsystem.

```
function [u,mudot,f_hat,err] = fcn(e,mu,Up,wd,wddot)
J=100000;
persistent e0 wdk1
k1=10000000; k2=2; a1=3;
if isempty(e0)
    e0=e;
    wdk1=wd;
end

if Up==1
    e0=wd-wdk1;
    wdk1=wd;
end
f_hat=(k1+1)*(e-e0)+mu;
mudot=(k1+1)*a1*e+k2*sign(e);
w=e+wd;
err=e;
```

```
Cd=1;
u=-J*wddot-J*a1*e+f_hat;
wdk1=wd;
end
```

The SPSA subsystem in Figure 4.11 is the same as that of outer loop which has been described in the previous section of Appendix B. The difference is that the value of Cp here is no longer the actual power coefficient because the actual Cp is usually not practically available. Instead, the values of Cp here is an estimate of the power coefficient of the wind turbine obtaining from the speed regulation control law. The Cpestimator subsystem in Figure 4.11 is used for calculation the estimate of Cp. Figure B.6 shows the block diagram of the Cpestimator subsystem. This subsystem collects the wind speed, the actual angular velocity of the wind turbine and f_hat from the speed regulation control law to calculate the actual power coefficient shown as the Cphat1 block.

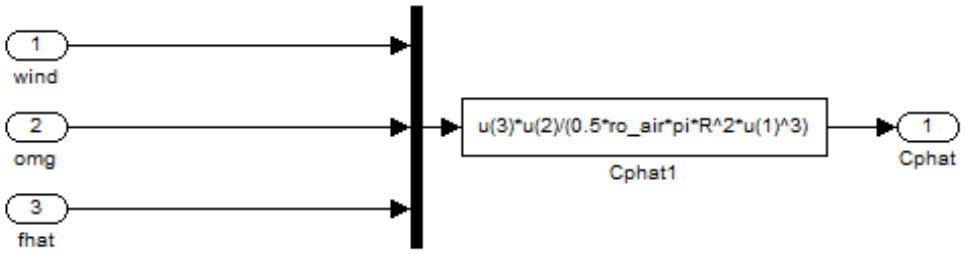


Figure B.6 Block diagram of the Cpestimator subsystem in Figure 4.11

Appendix C – Simulink Embedded Code and Subsystem Blocks in Chapter 5

In Chapter 5, Figure 5.4 shows the diagram of the Simulink model using SPSA for extremum seeking of FAST. For comparison, Figure 5.9 is the simulation diagram using the classical gradient based extremum seeking algorithm. This appendix will show in detail all of the information contained within simulations

Simulation on FAST using SPSA method

The simulation on FAST in Figure 5.4 uses the SPSA method for extremum seeking. Besides the FAST subsystem, there are four other subsystems in this diagram which are torque controller subsystem, yaw controller subsystem, pitch controller subsystem and the outer_loop_control subsystem.

FAST subsystem is used to simulate a wind turbine and its inputs are set as follows.

```
---------------------------------------------------------------------------
------- FAST INPUT FILE -------------------------------------------------
NREL 5.0 MW Baseline Wind Turbine for Use in Offshore Analysis.
Properties from Dutch Offshore Wind Energy Converter (DOWEC) 6MW Pre-Design (10046_009.pdf) and REpower 5M 5MW
(5m_uk.pdf); Compatible with FAST v7.01.
---------------------- SIMULATION CONTROL --------------------------------------
False       Echo        - Echo input data to "echo.out" (flag)
   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)
   3        NumBl       - Number of blades (-)
 630.0      TMax        - Total run time (s)
   0.0125   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) [unused when YCMode=0]
   2        PCMode      - Pitch control mode {0: none, 1: user-defined from routine PitchCntrl, 2: user-defined from
Simulink} (switch)
   0.0      TPCOn       - Time to enable active pitch control (s) [unused when PCMode=0]
   3        VSContrl    - Variable-speed control mode {0: none, 1: simple VS, 2: user-defined from routine UserVSCont,
3: user-defined from Simulink} (switch)
9999.9      VS_RtGnSp   - Rated generator speed for simple variable-speed generator control (HSS side) (rpm) [used only
when VSContrl=1]
9999.9      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]
9999.9      VS_Rgn2K    - Generator torque constant in Region 2 for simple variable-speed generator control (HSS side)
(N-m/rpm^2) [used only when VSContrl=1]
9999.9      VS_SlPc     - Rated generator slip percentage in Region 2 1/2 for simple variable-speed generator control
(%) [used only when VSContrl=1]
   2        GenModel    - Generator model {1: simple, 2: Thevenin, 3: user-defined from routine UserGen} (switch) [used
only when VSContrl=0]
True        GenTiStr    - Method to start the generator {T: timed using TimGenOn, F: generator speed using SpdGenOn}
(flag)
True        GenTiStp    - Method to stop the generator {T: timed using TimGenOf, F: when generator power = 0} (flag)
9999.9      SpdGenOn    - Generator speed to turn on the generator for a startup (HSS speed) (rpm) [used only when
GenTiStr=False]
   0.0      TimGenOn    - Time to turn on the generator for a startup (s) [used only when GenTiStr=True]
9999.9      TimGenOf    - Time to turn off the generator (s) [used only when GenTiStp=True]
   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]
  14.92     BlPitch(1)  - Blade 1 initial pitch (degrees)
  14.92     BlPitch(2)  - Blade 2 initial pitch (degrees)
  14.92     BlPitch(3)  - Blade 3 initial pitch (degrees) [unused for 2 blades]
   0.0      B1PitchF(1) - Blade 1 final pitch for pitch maneuvers (degrees)
   0.0      B1PitchF(2) - Blade 2 final pitch for pitch maneuvers (degrees)
   0.0      B1PitchF(3) - Blade 3 final pitch for pitch maneuvers (degrees) [unused for 2 blades]
---------------------- ENVIRONMENTAL CONDITIONS --------------------------------
   9.80665  Gravity     - Gravitational acceleration (m/s^2)
---------------------- FEATURE FLAGS -------------------------------------------
True        FlapDOF1    - First flapwise blade mode DOF (flag)
True        FlapDOF2    - Second flapwise blade mode DOF (flag)
True        EdgeDOF     - First edgewise blade mode DOF (flag)
False       TeetDOF     - Rotor-teeter DOF (flag) [unused for 3 blades]
True        DrTrDOF     - Drivetrain rotational-flexibility DOF (flag)
True        GenDOF      - Generator DOF (flag)
True        YawDOF      - Yaw DOF (flag)
True        TwFADOF1    - First fore-aft tower bending-mode DOF (flag)
True        TwFADOF2    - Second fore-aft tower bending-mode DOF (flag)
True        TwSSDOF1    - First side-to-side tower bending-mode DOF (flag)
True        TwSSDOF2    - Second side-to-side tower bending-mode DOF (flag)
True        CompAero    - Compute aerodynamic forces (flag)
False       CompNoise   - Compute aerodynamic noise (flag)
---------------------- 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)
  12.1      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 --------------------------------
  63.0      TipRad      - The distance from the rotor apex to the blade tip (meters)
   1.5      HubRad      - The distance from the rotor apex to the blade root (meters)
   1        PSpnElN     - 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] (-)
   0.0      UndSling    - Undersling length [distance from teeter pin to the rotor apex] (meters) [unused for 3 blades]
   0.0      HubCM       - Distance from rotor apex to hub mass [positive downwind] (meters)
  -5.01910  OverHang    - Distance from yaw axis to rotor apex [3 blades] or teeter pin [2 blades] (meters)
   1.9      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)
   1.75     NacCMzn     - Vertical distance from the tower-top to the nacelle CM (meters)
  87.6      TowerHt     - Height of tower above ground level [onshore] or MSL [offshore] (meters)
   1.96256  Twr2Shft    - Vertical distance from the tower-top to the rotor shaft (meters)
   0.0      TwrRBHt     - Tower rigid base height (meters)
  -5.0      ShftTilt    - Rotor shaft tilt angle (degrees)
   0.0      Delta3      - Delta-3 angle for teetering rotors (degrees) [unused for 3 blades]
  -2.5      PreCone(1)  - Blade 1 cone angle (degrees)
  -2.5      PreCone(2)  - Blade 2 cone angle (degrees)
  -2.5      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)
 240.00E3   NacMass     - Nacelle mass (kg)
  56.78E3   HubMass     - Hub mass (kg)
   0.0      TipMass(1)  - Tip-brake mass, blade 1 (kg)
   0.0      TipMass(2)  - Tip-brake mass, blade 2 (kg)
   0.0      TipMass(3)  - Tip-brake mass, blade 3 (kg) [unused for 2 blades]
2607.89E3   NacYIner    - Nacelle inertia about yaw axis (kg m^2)
 534.116    GenIner     - Generator inertia about HSS (kg m^2)
 115.926E3  HubIner     - Hub inertia about rotor axis [3 blades] or teeter axis [2 blades] (kg m^2)
---------------------- DRIVETRAIN -------------------------------------------
 100.0      GBoxEff     - Gearbox efficiency (%)
  94.4      GenEff      - Generator efficiency [ignored by the Thevenin and user-defined generator models] (%)
  97.0      GBRatio     - Gearbox ratio (-)
False       GBRevers    - Gearbox reversal {T: if rotor and generator rotate in opposite directions} (flag)
  28.1162E3 HSSBrTqF    - Fully deployed HSS-brake torque (N-m)
   0.6      HSSBrDT     - Time for HSS-brake to reach full deployment once initiated (sec) [used only when HSSBrMode=1]
            DynBrkFi    - File containing a mech-gen-torque vs HSS-speed curve for a dynamic brake [CURRENTLY IGNORED]
(quoted string)
 867.637E6  DTTorSpr    - Drivetrain torsional spring (N-m/rad)
   6.215E6  DTTorDmp    - Drivetrain torsional damper (N-m/(rad/s))
---------------------- SIMPLE INDUCTION GENERATOR ---------------------------
9999.9      SIG_SlPc    - Rated generator slip percentage (%) [used only when VSContrl=0 and GenModel=1]
9999.9      SIG_SySp    - Synchronous (zero-torque) generator speed (rpm) [used only when VSContrl=0 and GenModel=1]
```

```
9999.9      SIG_RtTq    - Rated torque (N-m) [used only when VSContrl=0 and GenModel=1]
9999.9      SIG_PORt    - Pull-out ratio (Tpullout/Trated) (-) [used only when VSContrl=0 and GenModel=1]
---------------------- THEVENIN-EQUIVALENT INDUCTION GENERATOR -----------------
9999.9      TEC_Freq    - Line frequency [50 or 60] (Hz) [used only when VSContrl=0 and GenModel=2]
9998        TEC_NPol    - Number of poles [even integer > 0] (-) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_SRes    - Stator resistance (ohms) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_RRes    - Rotor resistance (ohms) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_VLL     - Line-to-line RMS voltage (volts) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_SLR     - Stator leakage reactance (ohms) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_RLR     - Rotor leakage reactance (ohms) [used only when VSContrl=0 and GenModel=2]
9999.9      TEC_MR      - Magnetizing reactance (ohms) [used only when VSContrl=0 and GenModel=2]
---------------------- PLATFORM ------------------------------------------------
   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 ---------------------------------------------------
  20        TwrNodes    - Number of tower nodes used for analysis (-)
"NRELOffshrBsline5MW_Tower_Onshore.dat"          TwrFile     - Name of file containing tower properties (quoted string)
---------------------- NACELLE-YAW ---------------------------------------------
9028.32E6   YawSpr      - Nacelle-yaw spring constant (N-m/rad)
  19.16E6   YawDamp     - Nacelle-yaw damping 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) [unused when Furling=False]
---------------------- ROTOR-TEETER --------------------------------------------
   0         TeetMod     - Rotor-teeter spring/damper model {0: none, 1: standard, 2: user-defined from routine UserTeet}
(switch) [unused for 3 blades]
   0.0      TeetDmpP    - Rotor-teeter damper position (degrees) [used only for 2 blades and when TeetMod=1]
   0.0      TeetDmp     - Rotor-teeter damping constant (N-m/(rad/s)) [used only for 2 blades and when TeetMod=1]
   0.0      TeetCDmp    - Rotor-teeter rate-independent Coulomb-damping moment (N-m) [used only for 2 blades and when
TeetMod=1]
   0.0      TeetSStP    - Rotor-teeter soft-stop position (degrees) [used only for 2 blades and when TeetMod=1]
   0.0      TeetHStP    - Rotor-teeter hard-stop position (degrees) [used only for 2 blades and when TeetMod=1]
   0.0      TeetSSSp    - Rotor-teeter soft-stop linear-spring constant (N-m/rad) [used only for 2 blades and when
TeetMod=1]
   0.0      TeetHSSp    - Rotor-teeter hard-stop linear-spring constant (N-m/rad) [used only for 2 blades and when
TeetMod=1]
---------------------- TIP-BRAKE -----------------------------------------------
   0.0      TBDrConN    - Tip-brake drag constant during normal operation, Cd*Area (m^2)
   0.0      TBDrConD    - Tip-brake drag constant during fully-deployed operation, Cd*Area (m^2)
   0.0      TpBrDT      - Time for tip-brake to reach full deployment once released (sec)
---------------------- BLADE ---------------------------------------------------
"NRELOffshrBsline5MW_Blade.dat"                BldFile(1)  - Name of file containing properties for blade 1 (quoted
string)
```

```
"NRELOffshrBsline5MW_Blade.dat"                 BldFile(2)  - Name of file containing properties for blade 2 (quoted
string)
"NRELOffshrBsline5MW_Blade.dat"                 BldFile(3)  - Name of file containing properties for blade 3 (quoted
string) [unused for 2 blades]
---------------------- AERODYN -------------------------------------------------
"NRELOffshrBsline5MW_AeroDyn.ipt"               ADFile      - Name of file containing AeroDyn input parameters (quoted
string)
---------------------- NOISE ---------------------------------------------------
            NoiseFile   - Name of file containing aerodynamic noise input parameters (quoted string) [used only when
CompNoise=True]
---------------------- ADAMS ---------------------------------------------------
            ADAMSFile   - Name of file containing ADAMS-specific input parameters (quoted string) [unused when
ADAMSPrep=1]
---------------------- LINEARIZATION CONTROL -----------------------------------
            LinFile     - Name of file containing FAST linearization parameters (quoted string) [unused when AnalMode=1]
---------------------- OUTPUT --------------------------------------------------
True        SumPrint    - Print summary data to "<RootName>.fsm" (flag)
True        TabDelim    - Generate a tab-delimited tabular output file. (flag)
"ES10.3E2"  OutFmt      - Format used for tabular output except time.  Resulting field should be 10 characters. (quoted
string)  [not checked for validity!]
   0.0      TStart      - Time to begin tabular output (s)
   1        DecFact     - Decimation factor for tabular output {1: output every time step} (-)
   1.0      SttsTime    - Amount of time between screen status messages (sec)
  -3.09528  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)
   2.23336  NcIMUzn     - Vertical distance from the tower-top to the nacelle IMU (meters)
   1.912    ShftGagL    - Distance from rotor apex [3 blades] or teeter pin [2 blades] to shaft strain gages [positive
for upwind rotors] (meters)
   0        NTwGages    - Number of tower nodes that have strain gages for output [0 to 9] (-)
            TwrGagNd    - List of tower nodes that have strain gages [1 to TwrNodes] (-) [unused if NTwGages=0]
   3        NBlGages    - Number of blade nodes that have strain gages for output [0 to 9] (-)
 5,9,13     BldGagNd    - List of blade nodes that have strain gages [1 to BldNodes] (-) [unused if NBlGages=0]
            OutList     - The next line(s) contains a list of output parameters.  See OutList.txt for a listing of
available output channels, (-)
"WindVxi  , WindVyi  , WindVzi"                             - Longitudinal, lateral, and vertical wind speeds
"GenPwr   , GenTq"                                          - Electrical generator power and torque
"OoPDefl1 , IPDefl1  , TwstDefl1"                           - Blade 1 out-of-plane and in-plane deflections and tip
twist
"OoPDefl2 , IPDefl2  , TwstDefl2"                           - Blade 2 out-of-plane and in-plane deflections and tip
twist
"OoPDefl3 , IPDefl3  , TwstDefl3"                           - Blade 3 out-of-plane and in-plane deflections and tip
twist
"BldPitch1"                                                 - Blade 1 pitch angle
"Azimuth"                                                   - Blade 1 azimuth angle
"RotSpeed , GenSpeed"                                       - Low-speed shaft and high-speed shaft speeds
```

```
"TTDspFA  , TTDspSS  , TTDspTwst"                              - Tower fore-aft and side-to-side displacments and top
twist
"Spn2MLxb1, Spn2MLyb1"                                         - Blade 1   Bending    - Blade 1 local edgewise and
flapwise bending moments at span station 2 (approx. 50% span)
"RootFxc1 , RootFyc1 , RootFzc1"                              - Out-of-plane shear, in-plane shear, and axial forces at
the root of blade 1
"RootMxc1 , RootMyc1 , RootMzc1"                              - In-plane bending, out-of-plane bending, and pitching
moments at the root of blade 1
"Spn2MLxb2, Spn2MLyb2"                                         - Blade 2   Bending    - Blade 2 local edgewise and
flapwise bending moments at span station 2 (approx. 50% span)
"RootFxc2 , RootFyc2 , RootFzc2"                              - Out-of-plane shear, in-plane shear, and axial forces at
the root of blade 2
"RootMxc2 , RootMyc2 , RootMzc2"                              - In-plane bending, out-of-plane bending, and pitching
moments at the root of blade 2
"Spn2MLxb3, Spn2MLyb3"                                         - Blade 3   Bending    - Blade 3 local edgewise and
flapwise bending moments at span station 2 (approx. 50% span)
"RootFxc3 , RootFyc3 , RootFzc3"                              - Out-of-plane shear, in-plane shear, and axial forces at
the root of blade 3
"RootMxc3 , RootMyc3 , RootMzc3"                              - In-plane bending, out-of-plane bending, and pitching
moments at the root of blade 3
"RotTorq  , LSSGagMya, LSSGagMza"                              - Rotor torque and low-speed shaft 0- and 90-bending
moments at the main bearing
"YawBrFxp , YawBrFyp , YawBrFzp"                               - Fore-aft shear, side-to-side shear, and vertical forces
at the top of the tower (not rotating with nacelle yaw)
"YawBrMxp , YawBrMyp , YawBrMzp"                               - Side-to-side bending, fore-aft bending, and yaw moments
at the top of the tower (not rotating with nacelle yaw)
"TwrBsFxt , TwrBsFyt , TwrBsFzt"                               - Fore-aft shear, side-to-side shear, and vertical forces
at the base of the tower (mudline)
"TwrBsMxt , TwrBsMyt , TwrBsMzt"                               - Side-to-side bending, fore-aft bending, and yaw moments
at the base of the tower (mudline)
"TipSpdRat, GenCp"                                           - Tip speed ratio and Generator Cp
END of FAST input file (the word "END" must appear in the first 3 columns of this last line).
---------------------------------------------------------------------------
```

Because it is assumed that the wind turbine is facing directly to the wind and the wind doesn't change its direction, yaw control is not needed. Thus, in the yaw controller, there is just a constant input which is zero. For the pitch controller subsystem, it is assumed that the blade pitch can be adjusted fast enough. So the desired blade pitch obtained from the outer loop controller will be the input to the FAST subsystem directly.

In the torque controller subsystem, the speed regulation control law mentioned in Chapter 3 is used. The block diagram of this subsystem is show as Figure C.1. Vector of constants contains all the constants that the speed regulation controller needs, and a PI controller is used also for regulating the speed.

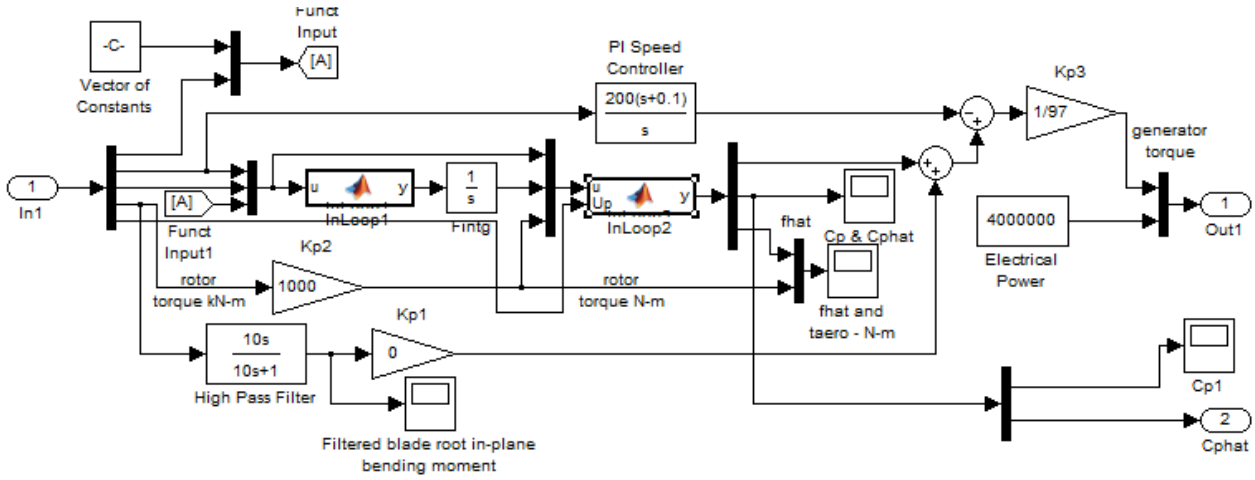


Figure C.1 Block diagram of torque controller subsystem in Figure 5.4

The MATLAB code of the embedded function Inloop1 in it are show as below.

```
function y = InLoop1(u)
    e      = -u(1) * 2*pi / 60.;    % e = -(wd-wr) -1*speed error (rad/sec)
    wr     =  u(2) * 2*pi / 60.;    % rotor speed - rad/sec.
    ks     =  u(3);
    betac  =  u(4);
    J      =  u(5);      % Rotor Inertia kg m^2
    alpha  =  u(6);
    A      =  u(7);      % Rotor swept area - m^2
    rho    =  u(8);      % density of air - kg/m^3
    vw     =  u(9);      % wind speed - m/sec.
    y      = zeros(1,1);

    Fint   = (ks+1)*alpha*e + betac * sign(e);    % F-hat integrand - N-m/s
    y      = Fint;
end
```

And the following is the MATLAB code of the embedded function InLoop2.

```
function y = InLoop2(u,Up)
    e      = -u( 1) * 2*pi / 60.;    % e = -(wd-wr) -1*speed error (rad/sec)
    wr     =  u( 2) * 2*pi / 60.;    % rotor speed - rad/sec.
    ks     =  u( 3);
    betac  =  u( 4);
```

```
    J      =  u( 5);         % Rotor Inertia kg m^2
    alpha  =  u( 6);
    A      =  u( 7);         % Rotor swept area - m^2
    rho    =  u( 8);         % density of air - kg/m^3
    vw     =  u( 9);         % wind speed - m/sec.
    Fintg  =  u(10);         % partial estimate of F-hat - N-m
    rt     =  u(11);         % rotor torque - N-m
    y      = zeros(4,1);     % output array

    wd     = wr-e;
    persistent e0 wdk1
    if isempty(e0)
        e0=e;
        wdk1=wd;
    end

    if Up==1
        e0=wd-wdk1;
        wdk1=wd;
    end

    Fhat   = Fintg + (ks+1)*(e-e0);              % aero torque estimate - N-m
    Taue   = J * alpha * e + Fhat;           % generator torque - N-m
    Cp     = rt   * wr / (0.5 * rho * A * vw^3);     % actual Cp
    Cphat  = Fhat * wr / (0.5 * rho * A * vw^3);     % estimated Cp
    y      = [Taue Cp Cphat Fhat]';          % function output vector
end
```

The outer_loop_control subsystem takes in the time signal, the estimate of Cp and the wind speed to calculate the path for the turbine to reach the maximum of Cp. The following is the block diagram.

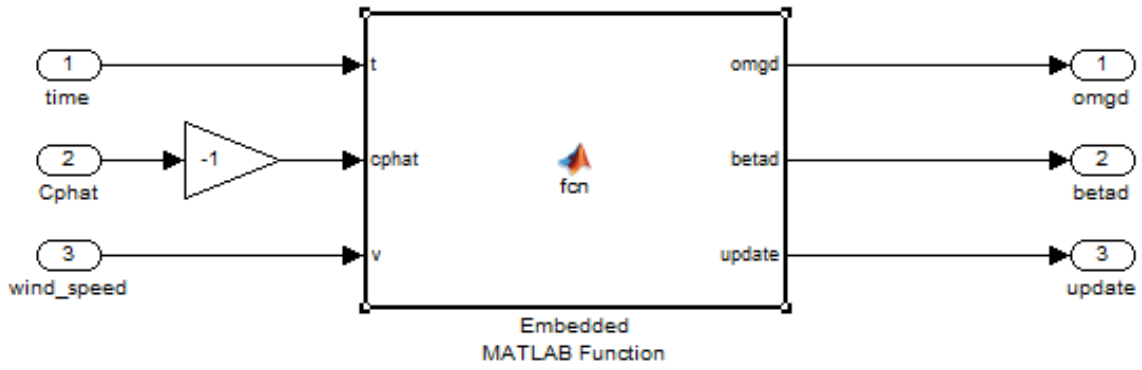


Figure C.2 Block diagram of the outer_loop_control subsystem in Figure 5.4

And the code of the embedded MATLAB function is shown as below.

```
function [omgd,betad,update]    = fcn(t,cphat,v)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Declaration of Constants
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
thetaini=[10 1]';%lambda,beta
lambdaini=thetaini(1);
betaini=thetaini(2);
n=30000;
R=63;%%
p=2;
a=10;
A=1;
c=20;%converge speed,the smaller the faster- 200
alpha=2;%%- 0.5
gamma=3;%converge speed, the larger the faster - 1
maxstep=2000;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Declaration of Persistent Variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
persistent up k theta lambdaout betaout ktk1 ki kik1 yplus yori ak ck delta
thetaplus thetaminus ghat ;
% persistent k ak ck delta Cphatplus Cphatminus up;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialization of persistent variables for time t=0
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if isempty(k)
    ki=0;
    kik1=0;
    ktk1=0;
    k=0;
    theta=thetaini;
    lambdaout=lambdaini;
    betaout=betaini;
    yplus=0;
    yori=0;
    ak=0;
    ck=0;
    delta=[0 0]';
    thetaplus=thetaini;
    thetaminus=thetaini;
    ghat=[0 0]';
    up=0;
else
    k=floor(t/100);
    ki=floor((t-100*k)/50);
    up=0;
end
if k>0 && (ktk1-k)<0
    yori=cphat;
    ak=a/(k+A)^alpha;
    ck=c/k^gamma;
    delta=2*round(rand(p,1))-1;
    thetaplus=theta+ck*delta;
    lambdaout=thetaplus(1);
    betaout=thetaplus(2);
    up=1;
end
```

```
if k>0&&ki==1&&(kik1-ki)<0
    yplus=cphat;
    ghat=(yplus-yori)./(ck*delta);
    theta=theta-ak*ghat;
    lambdaout=theta(1);
    betaout=theta(2);
    up=1;
end
ktk1=k;
kik1=ki;
betad=betaout;
if (lambdaout*v/R)*60/(2*pi)>15
    omgd=15;
else
    omgd=(lambdaout*v/R)*60/(2*pi);
end
update=up;
```

Simulation on FAST using classical gradient based method

The simulation on FAST in Figure 5.9 uses the classical gradient based extremum seeking algorithm. The general structure of the simulation is very similar to that of Figure 5.4. The inputs of the FAST subsystem are the same as that listed in the previous section. The yaw controller subsystem and the pitch controller subsystem are the same as that mentioned in last section because the assumptions of constant wind velocity and fast blade pitch adjustment still hold. Also, the block diagram of the torque controller is the same as shown in Figure C.1 and the code for embedded MATLAB function is the same.

The outer_loop_control block uses classical gradient based extremum seeking algorithm which is different from the SPSA method. The following is the code of the subsystem.

```
function y    = Find_Max(u)
    vw        = u( 1);                  % Wind speed - m/sec.
    t         = u( 2);                  % time - sec.
    count     = u( 3);                  % count for time steps
    steps     = u( 4);                  % increment for Cp points
    Cphatc    = u( 5);                  % Cp at initial point
    Cphatt    = u( 6);                  % Cp at top point
    Cphatr    = u( 7);                  % Cp at right point
    Cphatb    = u( 8);                  % Cp at bottom point
    Cphatl    = u( 9);                  % Cp at left point
    lambdad   = u(10);                  % Tip speed ratio
    betad     = u(11);                  % blade pitch - rad
    wr        = u(12) * 2*pi/60;        % rotor speed - rad/s
    Cphat     = u(13);                  % latest estimate of Cp

    y         = zeros(10,1);            % allocate space for output array

    % parameters
```

```
    fb      =   0.2;                    % step fraction - beta
    fm      =   6.0;                    % multiplying factor on step
    tstart  =  30.0;                    % time to start maximization
    tstep   =  10.0;                    % time between steps
    dlambda =   0.10;                   % purturbation in tip speed ratio
    dbeta   =   0.05;                   % purturbation in blade pitch - rad
    R       =  63.0;                    % Blade tip radius - m

    if t   < tstart
        count     = 0;                  % zero count
        steps     = -1;                 % zero steps
      else
        if t > tstart + count*tstep
            count = count + 1;          % increment count
            steps = steps + 1;          % increment steps
            if steps == 0
                lambdad   = lambdad;
                betad     = betad;
            end
            if steps == 1
                Cphatc    = Cphat;          % Initial Cp
                betad     = betad + dbeta;  % top point - beta
            end
            if steps == 2
                Cphatt    = Cphat;              % Cp at top
                lambdad   = lambdad + dlambda;  % right point - lambda
                betad     = betad - dbeta;      % right point - beta
            end
            if steps == 3
                Cphatr    = Cphat;              % Cp at right
                lambdad   = lambdad - dlambda;  % bottom point - lambda
                betad     = betad - dbeta;      % bottom point - beta
            end
            if steps == 4
                Cphatb    = Cphat;              % Cp at bottom
                lambdad   = lambdad - dlambda;  % left point - lambda
                betad     = betad + dbeta;      % left point - beta
            end
            if steps   == 5
                Cphatl  = Cphat;                          % Cp at left
                dcpdb   = (Cphatt-Cphatb)/(2*dbeta);      % grad wrt beta
                dcpdl   = (Cphatr-Cphatl)/(2*dlambda);    % grad wrt lambda
                lambdad = lambdad + dlambda;              % original lambda
                betad   = betad + dcpdb*dbeta*fm*fb;      % new beta value
                if betad < 0.0
                    betad = 0.0;
                end
                lambdad = lambdad + dcpdl*dlambda*fm;     % new lambda value
                steps   = -1;                             % reset steps
            end
        end
    end
    wd     = vw * lambdad / R;                  % omega desired - rad/s
    y      = [wd*60/(2*pi) count steps Cphatc Cphatt Cphatr Cphatb ...
              Cphatl lambdad betad]';           % output array
end                                             % wd is now in RPM
```