OPTIMIZATION OF WIND FARM LAYOUTS USING INTELLIGENTLY TUNED HARMONY SEARCH ALGORITHM

Narasimha Prasad Prabhu

Bachelor of Engineering, Manipal Institute of Technology, 2011

A PROJECT REPORT SUBMITTED

FOR THE DEGREE OF

Master of Science

DEPARTMENT OF COMPUTER AND ELECTRICAL ENGINEERING

National University of Singapore

2012

Declaration

”I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.”

_____

Narasimha Prasad Prabhu

November 29, 2012

Summary

Wind farm layouts are designed to minimize wake interactions between individual turbines to guarantee maximum power generation. The above mentioned issue has been formulated as an optimization problem with an objective to maximize the power output of each wind turbine, taking into consideration the environmental conditions prevalent in the region. The optimization algorithm has been applied to three general cases which encompass most scenarios, viz. constant wind speed and direction, constant wind speed with variable direction and variable wind speed and direction. An "Intelligently Tuned Harmony Search(ITHS)" algorithm along with the "Unrestricted Wind Flow Optimization(UWFLO)" model has been employed to predict the layout required to generate maximum power, given constraints of farm boundary, number of turbines and wind conditions. The UWFLO model takes into account the overlap area between the wake generated by an upstream turbine and the blades of a downstream turbine to predict with greater accuracy the total power generated by the wind farm. The ITHS algorithm allows for a proper balance between diversification and intensification, thus performing better than genetic and harmony search algorithms, the results of which have been compared with the present study. The document also performs a theoretical evaluation for micro-siting, during the operational phase of the wind farm based on prevalent wind conditions.The document also explores basic PD control of a model wind-turbine platform with mooring lines and thruster control. ITHS is able to perform better in two out of the three cases undertaken in this study by a margin of 4% in one and 6% in another. Micro-siting has been shown to provide an additional 3% increase in efficiency during the operational phase of the wind farm as compared to the sub-optimal placement done during the developmental phase.

Acknowledgments

If I have seen farther it is by standing on the shoulders of Giants. Sir Isaac Newton (1855)

I would like to thank A/P S.K. Panda for giving me the opportunity to work on this project and Mr. Parikshit Yadav for his continual guidance on optimization algorithms and encouragement without which, this endeavor would not have been possible.

I would also like to acknowledge the support I received from Bhuneshwar Prasad, Vinoth Viswanathan on the design and build of the prototypes and Alok Agrawal for the fluid dynamic analysis of the prototype. And last but not least I would like to thank Praveen Kumar for his assistance in modeling environmental loads.

Contents

List of Tables

List of Figures

Glossary

a Axial Induction Factor

k Entrainment Constant

HS Harmony Search

IHS Improved Harmony Search

ITHS Intelligently Tuned Harmony Search

GA Genetic Algorithm

HM Harmony Memory

MS Harmony Memory Size

HMCR Harmony Memory Consideration Rate

PAR Pitch Adjustment Rate

bw Harmony BandWidth

Chapter 1

Introduction

Climate change is no longer a debatable issue and keeping the planet's temperature at sustainable levels has become one of the major concerns of policy makers. To avoid the worst effects of climate change, global greenhouse gas emissions must peak and begin to decline before 2020[1]. With power generation being the largest contributor of greenhouse gases, the onus falls on the energy sector to either increase energy efficiency and conservation or switch from carbon based fuels to renewable energy. With such short time frames, it is essential to use current state of the art to tackle climate change. The recent failure of the of the nuclear power generation plant at Fukushima, Japan has led several countries, chiefly among the European Union, to actively pursue other sources of viable renewable energy. Chief among these are solar, hydro and wind. Germany has been particularly aggressive in this regards with plans for renewable sources supplying 35% by 2020 and 80% by 2050. With wind power providing more than 40% of the current renewable energy, Germany plans on focussing on off-shore wind farms to meet current energy requirements[2].

The trend in wind power generation is towards offshore installations as turbines installed over open seas provide several advantages as compared to those installed over land. Wind velocities over open water is significantly higher than those recorded over land. Also, wind over open seas is more predictable owing to the fact that there are zero obstacles on open sea. Second, locating wind farms offshore reduce public concerns related to noise, wildlife and aesthetics. As of 2010 , global off-shore capacity was only 3.16 GW, but projected to grow to 16 GW by the end of 2014 and 75 GW by 2020.

Power generation through wind resources is usually done through wind farms arranged in a particular pattern over a large stretch of land. Though wind farms have mainly been onshore since the emergence of the technology, the lack of real estate and other factors have driven countries to pursue off-shore wind farms. It is generally understood that the efficiency of the wind farm is not just the summation of the power produced by stand alone turbines owing primarily to wake produced by upstream wind turbines. The energy deficit produced due to the upstream turbine wake is estimated using wake models which calculate the growth of the wake and velocity deficit due to the wake itself. These models can be divided into two main categories viz.

- Analytical wake model
- Computation wake model

An analytical wake model characterizes the wake based on the velocity in a wake and the wake expansion. While a computational wake model is characterized by fluid flow equations.

Betz [3] and Lanchester [4] in the 1920, pioneered the study into the analysis of the wake produced by the wind turbines and the impact on the efficiency of the output of the turbine based on a control volume approach(wake is modelled as a volume moving with constant velocity in space). Frandsen[5] used the Betz model to study the wake in the context of wind farms and the effect the wake models had on different configurations of wind farm. But the model only accounted for regular array geometry, i.e. straight row of wind turbines and equidistant spacing between units in each row and equidistant spacing between rows. Most studies conducted on the configuration of wind turbine placement utilize the Jensen wake model [6] which models the wake as a linearly expanding region, ignoring vortices at the tip. While this negates the effectiveness of the model at close proximity, it provides a relatively accurate estimation of wake at medium to large distances.

The principal issue with wind farms is , given the number of wind turbines to be installed , to determine the optimal layout of individual wind turbine so as to maximise the energy production. This may depend on several factors such as terrain, wind speed and direction, type of turbine and installation costs. One of the early works in this field was carried out Mosetti et al.[7] This paper proposed a genetic algorithm to tackle the problem of optimal positioning of turbines in a wind

farm. Grady et al.[8] showed better results with improvements in the selection of parameters used in the genetic algorithm used. Another interesting research by Emami et al. proposed a modification of the objective function, taking into account deployment cost and efficiency of the turbines. A paper by Riquelme et al.[9] employed a variable length genetic algorithm with novel procedures of crossover to obtain the optimal positioning of wind turbines. While genetic algorithms are popular, several alternate methods have also been applied to the design of off-shore wind farms. Rivas et al.[10] used a simulated annealing algorithm to solve the problem of optimization of wind turbine placement. A study published by Saavedra et al.[11] employed a greedy heuristic algorithm while considering a detailed wind farm model including orography, shape of the wind farm and cost of installation. Alternatives approaches, not utilizing evolutionary computation have been carried out. Marmidis et al.[12] approached the problem through Monte-Carlo simulation while Mustakerov et al[13]. wherein the authors model the statement as a non-linear combinatorial optimization problem.

Harmony Search, unlike GA considers several existing vectors in the search space and as such may be better suited to find global optima for the problem of wind farm layout optimization. Also, HS has been shown to work with combinatorial problems and does not require the initialization of problem parameters, a problem present in simulated annealing. The adaptation of the HS search algorithm employed in this study offers several advantages over traditional HS by offering an innovative solution to provide both intensification and diversification.

Chapter 2

Wind Farm Model

Wind farm models can be broadly classified into two categories

- models that assume an array(row-column)[14] layout and
- models that assume a grid layout[8],[9].

The first set of models optimize the horizontal distances between turbines in a wind farm while the second set of models assign discrete grid numbers to the wind farm and optimize the pattern to provide the best optimization function. Since the array layout technique has limited freedom in terms of placement, it tends to produces sub-optimal results. In this study, the Unrestricted Wind Farm Layout Optimization(UWFLO)[15] which makes significant improvements over traditional analytical wind farm models has been utilized. The UWFLO model does take into account, the overlap between downstream turbines and the wake produced by upstream turbines. For the purposes of this study the following assumptions have been made

- constant axial induction factor of 0.32
- identical wind turbines
- wind turbine efficiency of 40%
- surface roughness of 0.3m
- wind turbine height of 60m
- uniform wind flow

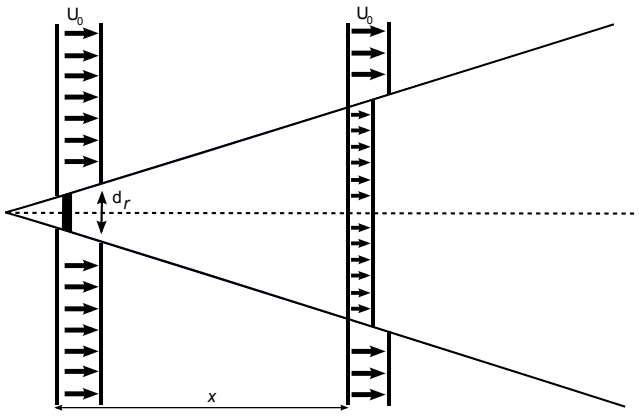


Figure 2.1: Linear Wake

- a 10x10 grid in a 2000x2000m wind farm

In the wind farm model, the growth of the wake created by an upstream turbine is determined by the Jensen wake model, which has been widely used to model wind turbines. This wake model ignores the vortex shedding which occurs very close to the turbine blades. Though this phenomenon is significant at close proximity, over large distances it can be ignored. Thus the Jensen wake model is derived by conserving momentum in a linearly expanding wake where the wind velocity in the wake is a function of the distance between the point of creation at the upstream turbine to the downstream turbine. The wake interactions modelled by the Jensen wake model are then determined by the UWFLO which estimates the resultant wind velocity as a quadratic sum of all the deficits due to individual upstream turbines.

2.1 Analytical Wake Model

Based on the Jensen wake decay model it can be shown that the power generated by a turbine is the function of the incident wind speed. The reduced wind speed, u at a distance y, in the wake of an upstream turbine is given by

$$\frac{u}{u_0} = 1 - \frac{2a}{(1+\frac{kx}{r_1})^2} \tag{2.1}$$

where

u_0 - the mean wind speed incident on wind farm

a - axial induction factor

k - entrainment constant and

r_1 - downstream rotor radius

The downstream rotor radius r_1 the turbine coeffecient C_T are a function of the axial induction factor a and rotor radius r_r as shown by

$$r_1 = r_r\sqrt{\frac{1-a}{1-2a}} \tag{2.2}$$

$$C_T = 4a(1-a) \tag{2.3}$$

The entrainment constant α is calculated using

$$\alpha = \frac{0.5}{ln\frac{z}{z_0}} \tag{2.4}$$

where z is the hub height of the turbine and z_0 is the surface roughness.

2.2 Analytical Power Generation Model

The power generated by a wind farm is a function of the co-ordinates of individual wind turbines and the current wind factors, viz speed and direction. The total power generated by the wind farm is calculated by following the algorithms proposed by Chouwdhury et al[15] as described below

Step 1 Each turbine is assignes co-ordinate(X,Y) based on a fixed co-ordinate system (X_i, Y_i). This system is then transformed into another coordinate system (x,y) such that the positive x axis is aligned with the direction of the wind and the direction of the y axis is from left to right.

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{2.5}$$

In Eqn. 2.5, θ is the angle with the positive x axis when measured clockwise. The distances between any two turbines is given by

$$\Delta x_{ij} = x_i - x_j \tag{2.6}$$

$$\Delta y_{ij} = y_i - y_j \tag{2.7}$$

Step 2 A wake identification matrix is constructed to determine if turbine *j* and turbine *i* influence each other

$$M_{ij} = \begin{cases} +1 & \text{if turbine } i \text{ influences turbine } j \\ -1 & \text{if turbine } j \text{ influences turbine } i \\ 0 & \text{if there is no mutual interaction} \end{cases} \tag{2.8}$$

where turbine-*j* is the downstream turbine in the wake of upstream turbine-*i* iff

$$\Delta x_{ij} < 0 \quad |\Delta y_{ij}| - \frac{D_j}{2} < \frac{D_{wake,ij}}{2} \tag{2.9}$$

where D_j is the rotor diameter of turbine-*j* and $D_{wake,ij}$ is the diameter of the wake created by turbine-*i* on turbine-*j*.

Step 3 The turbines are sorted in the increasing order of *x* axis values, with the first element facing the wind. In case two turbines have the same *x* value, they are then ranked on basis of their *y* axis values, in increasing order.

Step 4 The power generated by each turbine is compared sequentially in the order of their rank. Thus the turbines facing the wind are evaluated first with turbines located at the inside of the wind farm evaluated later. This methodology ensures that the wake effects of all upstream turbines are evaluated. The wakes, either partial or complete are evaluated $\forall\, M_{ij} = 1$, implying turbine *j* is in the wake of turbine *i*

- if the rotor of turbine *j* is completely in the wake of turbine *i*

$$\begin{aligned} A_{ij} &= A_j \\ A_j &= \frac{\pi D_j^2}{4} \end{aligned} \tag{2.10}$$

- if the rotor of turbine j is partially in the wake of turbine k

$$A_{kj} = r_k^2 cos^{-1}\left(\frac{d_{kj}^2 + r_k^2 - r_j^2}{2d_{kj}r_k}\right) + r_j^2 cos^{-1}\left(\frac{d_{kj}^2 + r_j^2 - r_k^2}{2d_{kj}r_k}\right) - \frac{1}{2}\sqrt{(-d_{kj}+r_k+r_j)(d_{kj}-r_k+r_j)(d_{kj}+r_k-r_j)(d_{kj}+r_k+r_j)} \tag{2.11}$$

where

A_{kj} is the effective area of influence of the wake of turbine k on turbine j

r_k area of wake front given by

$$r_k = \frac{\pi D_{wake}^2}{4} \tag{2.12}$$

r_j area of blades for turbine j

d_{kj} is the distance between the centre of the wake area and the turbine blades.

The contribution to the wake of each upwind turbine approaching turbine j is thus given by

$$p_{kj} = \frac{A_{kj}}{A_j} U_{kj}^2 \tag{2.13}$$

where U_{kj} is the velocity incident on turbine j due to the wake created by turbine k. For details regarding the calculation of incident wind velocity refer Eqn. 2.1.

The resultant velocity of the wind approaching turbine j is expressed as

$$U_j = U_0 - \sqrt{\sum_k \frac{A_{kj}}{A_j}(U_0 - U_{kj})^2} \tag{2.14}$$

The power generated by turbine j for an incident velocity u is then given by

$$P_j = \eta \frac{1}{2} \rho A u^3 \tag{2.15}$$

Assuming wind turbine efficiency η = 40% and ρ = 1.2 $\frac{kg}{m^3}$

$$P_j = \frac{40}{100} * \frac{1}{2} * 1.2 * \pi * 20^2 * u^3$$
$$= 0.3u^3 kW \tag{2.16}$$

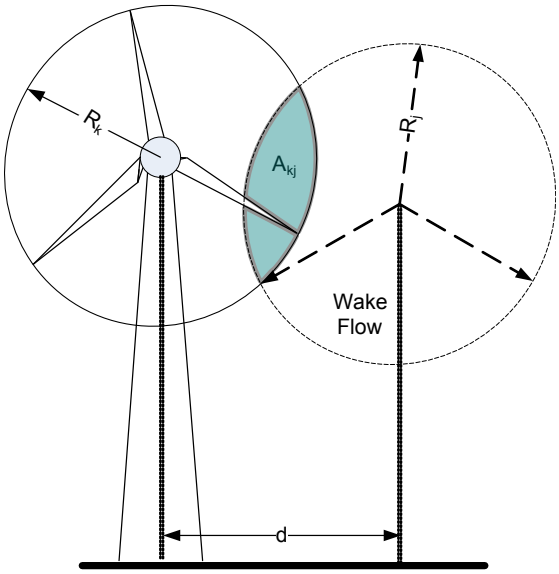


Figure 2.2: Wake Overlap

Step 5 The power generated by the entire wind farm P_{total} is a summation of power by all the individual wind turbines.

$$P_{total} = \sum_{j=1}^{N} P_j \tag{2.17}$$

and farm efficiency η_{farm} is given by

$$\eta_{farm} = \frac{P_{total}}{\sum_{j=1}^{N} P_{0j}} \tag{2.18}$$

where P_{0j} is the power a turbine would have produced if it encountered wind at full velocity.

2.3 Cost Model

To determine the cost of the wind farm, a cost model is formulated by assuming that the non-dimension cost/year of a single turbine is 1 and that a maximum cost reduction of $\frac{1}{3}$ can be obtained for each turbine if a large number of turbines are installed. The total cost/year of the whole windfarm can then be expressed by[7]

$$Cost_{tot} = N * (\frac{2}{3} + \frac{1}{3} * e^{-0.00174N^2}) \tag{2.19}$$

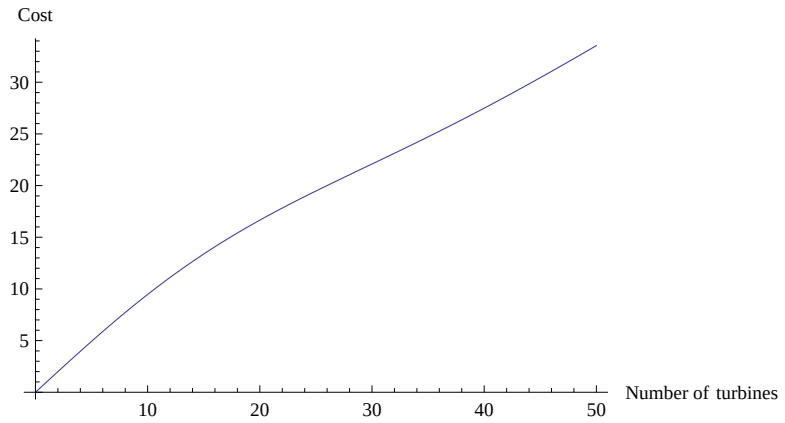


Figure 2.3: Cost Vs Number of Turbines

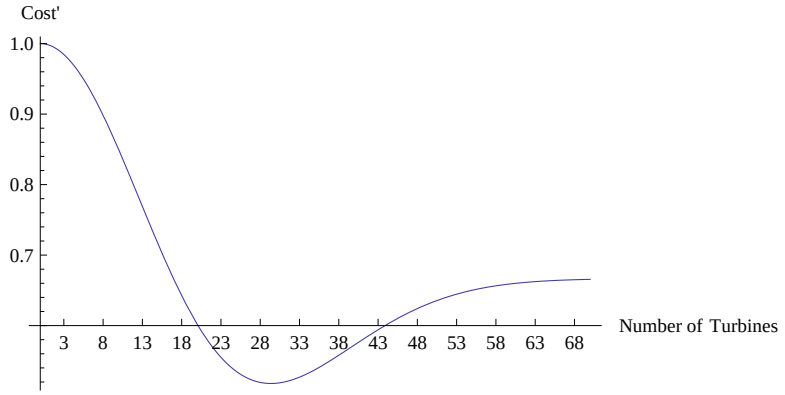


Figure 2.4: Cost' Vs Number of Turbine

Figure 2.3 shows the variation of cost with respect to the number of turbines installed. It is clear from the graph that cost as a function of the number of turbines installed is almost a linear. But Figure 2.4 is interesting as the derivative of cost with respect to the number of turbines is not a strictly decreasing function as in mass production. At around 30 the derivative of cost increases, which implies that the cost investment in installing the 31^{st} turbine would be greater

than to install the 30^{th} turbine, but the cost investment in installing the 29^{th} turbine would be less than that of installing the 28^{th} turbine, the result of which shall be explored in Chapter 5

A linear wake model(Jensen) combined with the UWFLO methodology has been employed to provide an accurate first representation of the entire wind farm. The cost model developed is a function of only the number of turbines installed and does not take into account factors dependant on the operation of the wind farm. The reason for the choice of both, the wake model and the cost model was to provide a relevant comparison between optimization algorithms to be used in this study and those used in previous studies. The UWFLO model though, does offer an accurate representation of the power generation capabilities of the wind farm, by taking into account not only the presence of a wake, but also the percentage overlap between the wake of an upstream turbine and the blades of a downstream turbine.

Chapter 3

Harmony Search Optimization algorithms

Harmony search(HS), a metaheuristic algorithm mimics the improvisation process of music players was developed by Zong Woo Geem et al.[16]. It is inspired by the process of music improvisation where a musician searches for a a harmony and continues to adjust the pitch of the instrument to improve the harmony with the pitch of the instruments being analogous to the objective function of the problem and the perfect harmony symbolizing the optimal value of parameters to get the best objective function.

HS algorithms have various advantages over other existing optimization algorithms such as considering all existing vectors rather than considering only two (parents) as in Genetic Algorithms (GA). It does not require an initialization of problem parameters and is applicable to both continuous and combinatorial problems. As the optimization is based on a stochastic process, the derivative informations is also not necessary.

3.1 Harmony Search

To overcome the drawbacks of the harmony search algorithm, mainly its inability to find local optimals for numerical applications, several variants of the HS algorithm have been proposed. In this study, an ”Intelligently Tuned Harmony Search” (ITHS) algorithm has been applied to the problem of optimizing power generation from a given wind farm. The ITHS algorithms tries to build upon the strengths of the HS algorithm. There are fewer influencing factors, viz.

HMCR and HMS and the ITHS algorithms also tries to maintain a proper balance between intensification and diversification by means of a self adaptive strategy which negates the need for more parameters. The following section gives a description of the procedure involved in the design of the HS algorithm and the subsequent implementation of the ITHS algorithm

Step 1: Initialise the optimization problem and algorithm parameters

To apply HS, the problem has to be formulated as an objective function with the following constraints

$$\text{Minimize(or Maximize)} f(\vec{x}) \tag{3.1}$$

subject to $x_i \in X_i, i = 1,2,3\ldots,N$

Where $f(\vec{x})$ is the objective function with $\vec{x}$ as the solution vector composed of decision variables x_i. X_i is a search space and is defined as an N-dimensional rectangle in R^N with all possible range of values for each decision variable x_i

$$X_i = x_i, \quad \underline{x_i} < x_i < \overline{x_i} \tag{3.2}$$

where $\lfloor x_i$ and $x_i \rceil$ are the lower and upper bounds for the i^{th} decision variable respectively. In addition, the parameters of the HS are specified in this step. These parameters are the Harmony memory size(HMS), Harmony Memory Considering Rate(HMCR), Pitch Adjusting Rate(PAR) and the number of improvisations Max_{iter}

Step 2: Initialise the Harmony Memory(HM)

The initial HM consists of a HMS number of randomly generated solutions for the formulated optimization problem. Each element of each vector in a HM is initialised with a uniformly distributed random number between the upper and lower bounds $[\underline{x_i}, \overline{x_i}]$, where $1 \leq i \leq N$. The i^{th} element of the j^{th} solution vector is then given by

$$x_i^j = \begin{cases} \underline{x_i} + (\overline{x_i} - \underline{x_i}) * rand[0,1] & \text{for continuous variables} \\ \underline{x_i} + \lfloor (\overline{x_i} - \underline{x_i}) * rand[0,1] \rfloor & \text{for discrete variables} \end{cases} \tag{3.3}$$

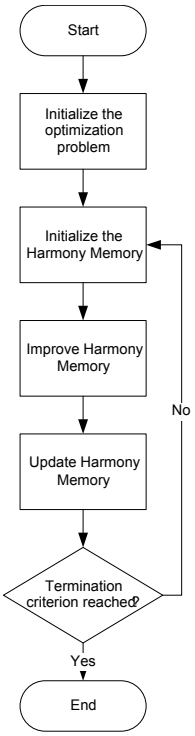


Figure 3.1: Flowchart of Harmony Search Algorithm

where $j = 1,2,3\ldots,HMS$ and *rand[0,1]* is a number drawn from a uniformly distribution on the open interval *[0,1]*. $\lfloor$ and $\rfloor$ represent the nearest integers. Each row of the HM now consists of a random solution for the optimization problem defined in *Step 1.* and the objective function value for the j^{th} solution vector is denoted by $F(\vec{x}^j)$ formed by Eqn. :3.4-3.5. The HM matrix of HMS x(N+1) is then formed by eq:3.6

$$HM(j,1:N) = \vec{x}^j \tag{3.4}$$

$$HM(j,N+1) = F(\vec{x})^j \tag{3.5}$$

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \ldots & x_N^1 & F(\vec{x}^j) \\ x_1^2 & x_2^2 & x_3^2 & \ldots & x_N^1 & F(\vec{x}^j) \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{HMS} & x_2^{HMS} & x_3^{HMS} & \ldots & x_N^1 & F(\vec{x}^{HMS}) \end{bmatrix} \tag{3.6}$$

Step 3: Improvise a new harmony from HM

After defining the HM as shown in Eqn. 3.6, the improvisation of the HM is done by generating

a new harmony vector $\vec{x}' = (x'_1 x'_2 x'_3 \ldots x'_N)$. Each component of the new harmony vector is generated using

$$x'_i \leftarrow \begin{cases} x'_i \in HM(i) & \text{with probability HMCR} \\ x'_i \in X_i & \text{with probability 1-HMCR} \end{cases} \tag{3.7}$$

where *HM(i)* is the i^{th} column of the *HM*. *HMCR* is defined as the probability of selecting a component from the *HM* members, and therefore *1-HMCR* is the probability of generating it randomly from the possible range of values. If x'_i is generated from the *HM*, then it is further modified or mutated according to the *Pith Adjustment Rate*(PAR). The PAR determines the probability of a candidate from the *HM* to be mutated and (1-*PAR*) is the probability of doing nothing. The *PAR* for the selected x'_i is given by

$$x'_i \leftarrow \begin{cases} \begin{cases} x'_i \pm rand[0,1].bw & \text{for continuous variables} \\ x'_j \pm \lfloor rand[0,1].bw \rfloor & \text{for discrete variables} \end{cases} & \text{with probability PAR} \\ x'_i & \text{with probability 1-PAR} \end{cases} \tag{3.8}$$

where *rand[0,1]* is the randomly generated number between 0 and 1 and *bw* is the pitch band width

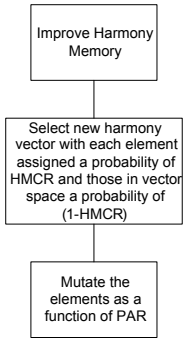


Figure 3.2: Flowchart for Improving Harmony Memory

Step 4: Update the HM

The newly generated harmony vector $\vec{x}'$ is evaluated in terms of the objective function value. If the objective value for the new harmony vector is better than the objective function value for the

worst harmony in the *HM*, then the new harmony is included in the *HM* and the existing worst harmony is excluded from the *HM*

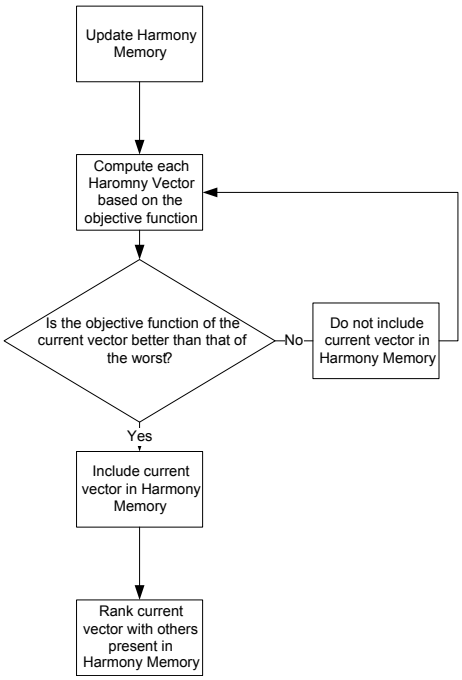


Figure 3.3: Flowchart for Updating Harmony Memory

Step 5: Go to *step 3* until termination criterion is reached

The iteration process in *steps 3 and 4* are repeated and the *HM* continuously updated with the best solution vectors $\vec{x}'$ till the algorithm reaches the termination criterion(Max_{iter}). Finally, the best solution is selected from the final *HM* and is considered the best solutions to the formulated optimization problem for the corresponding independent run.

3.2 Harmony Search Variants

The performance of the *HS* algorithm can be improved by tuning parameters *PAR* and *bw*. The parameter *bw* controls the diversification of the algorithms, i.e. it controls the search for solution vectors outside the local regions. Hence a variable *bw* with large values in initial iteration tending to smaller values in the later iterations is preferred. Mahdavi et al.[17] proposed an improvement to traditional *HS* algorithms by linearly increasing *PAR* and exponentially decreasing *bw*. The expressions for *PAR* and *bw* are given by

$$PAR_{iter} = PAR_{min} + (PAR_{max} - PAR_{min}).(\frac{iter}{Max_{iter}}) \tag{3.9}$$

$$bw_{iter} = bw_{max}.exp[\{ln(\frac{bw_{min}}{bw_{max}}\}.(\frac{iter}{Max_{iter}}) \tag{3.10}$$

where PAR_{max}, PAR_{min} are the maximum and minimum iteration rates and bw_{max}, bw_{min} are maximum and minimum bandwidth. This modification of the algorithms works well to increase the intensification of the process as the algorithm progresses, but is still dependant on the choice of the maximum and minimum boundaries of *PAR* and *bw*.

A Self Adaptive Harmony Search(SAHS) algorithm was introduced by Wang et al.[18] wherein the selection of both *PAR* and *bw* have been eliminated, thus negating the need to tune these parameters. The mutation of new harmony vectors are based on the maximum and minimum values in the *HM* given by

$$x_i' \leftarrow \begin{cases} x_i' + (max(HM)^i - x_i'.rand[0,1) & \text{with probability 0.5*PAR} \\ x_i' + (min(HM)^i - x_i'.rand[0,1) & \text{with probability 0.5*PAR} \\ x_i' & \text{with probability 1-PAR} \end{cases} \tag{3.11}$$

with required *PAR* for mutation chosen by a linearly decreasing function given by

$$PAR_{iter} = PAR_{max} - (PARmax - PAR_{min}).(\frac{iter}{Max_{iter}}) \tag{3.12}$$

In Eqn.3.11, $min(HM)^i$ and $max(HM)^i$ represent the least and greatest values of the i^{th} variable in the *HM*. Since the difference between these values are large in the initial iterations, the algorithm allows for diversification in the initial stages and as the solution vector converges on the optimal solution, decreases the *PAR*. However, if the optima lies outside the area of explorations defined by min(HM) and max(HM), the *SAHS* would completely miss it.

3.3 Intelligently Tuned Harmony Search

Yadav et al.[19] made further improvements to the *HS* algorithm by proposing an "Intelligently Tuned Harmony Search" algorithm, wherein the limiting factor of the *SAHS* algorithm was

eliminated by mutating the vectors in the *HM* based on the the best and mean values of the same.

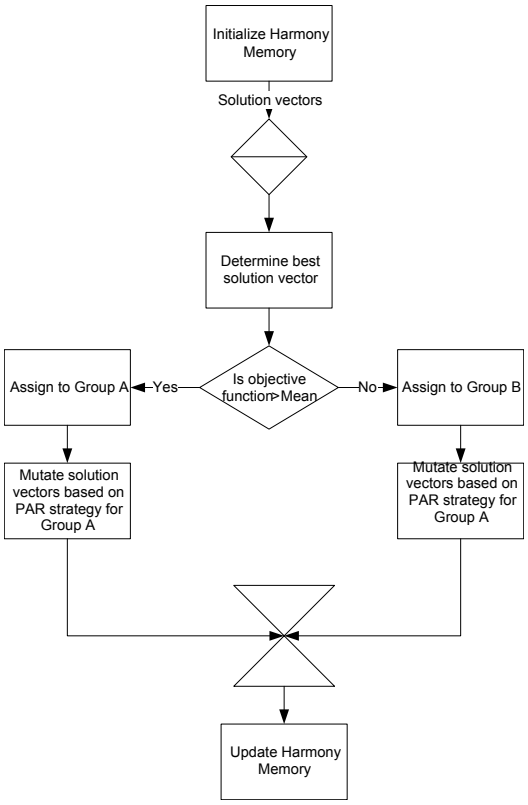


Figure 3.4: Flowchart of Key features in ITHS

The method is based on a tyrannical structure, where one dominant value drives the entire algorithm. A leader is first selected, based on the best objective function value given by

$$\vec{x}^{best} = HM(best, 1:N) \tag{3.13}$$

where *best* is the index of the best objective function in the *HM*. A *HM* is then divided into two groups, with one group containing objective functions above HM^{mean} and the other those solution vectors less than HM^{mean}. The first group($\equiv$ Group A) is responsible for both diversification and intesification while the second group($\equiv$ Group B) is responsible only for diversification. The algorithm updates the *PAR* as shown by

$$PAR_{iter} = PAR_{max} - (PAR_{max} - PAR_{min}).\frac{iter}{Max_{iter}} \tag{3.14}$$

which is similar to the method proposed by *SAHS*. The minimum and mximum values of *PAR* are fixed at 0 and 1. The *HM* improvisation for the selected x_i' is determined by the group to which it belongs given by

$$x_i' \leftarrow \begin{cases} x_i^{Best} - (x_i^{Best} - x_i').rand[0,1] & \text{with probability 0.5PAR} \\ x_i^{Best} + (x_i^{Worst} - x_i').rand[0,1] & \text{with probability 0.5PAR} \\ x_i' & \text{with probability (1-PAR)} \end{cases} \tag{3.15}$$

where x_i^{Best} and x_i^{Worst} denote the i^{th} variable of the best and the worst solution vectors. During initial iterations, a balance need to be struck between intesifcation and diversification. The term $x_i^{Best} - (x_i^{Best} - x_i').rand[0,1]$ is responsible for the intensification of the algorithm while $x_i^{Best} + (x_i^{Worst} - x_i').rand[0,1]$ is responsible for diversification, albeit governed by the *PAR* strategy. But this alone would fall into the same local optima problem faced by *SAHS* if the optimal solution were to lie beyond the areas defined by x_i^{Best} and $x_i Worst$. To address this problem Group B performs the role of a rebellion, with the mutation of solution vectors in this group given by

$$x_i' \leftarrow \{x_i' + (x_m^{Best} - x_i').rand[0,1] \quad \text{where m=int(1+(N-1)*rand)} \tag{3.16}$$

The choice of the rebellion mutation, though has been indicated as a debatable issue by Yadav et al. and have proposed a method to modify x_m^{Best} that ensures that each decision variable x_i is in the range defined by $[{}_L x_i, {}_U x_i]$ expressed as a function of Δ_m between 0 and 1. The new variable $(x_m^{Best})'$ is given by

$$x_M^{Best} = {}_L x_m + ({}_U x_m - {}_L x_m)\Delta_m \tag{3.17}$$

$$(x_M^{Best})' = {}_L x_i + ({}_U x_i - {}_L x_i)\Delta_m \tag{3.18}$$

$$= {}_L x_i + ({}_U x_i - {}_L x_i)\frac{x_m^{Best} - {}_L x_m}{{}_U x_m - {}_L x_m} \tag{3.19}$$

An adaptation of the HS algorithm has been implemented in the following study. The ITHS algorithm offers significant advantages with respect to the consideration of search space involved while retaining the advantages provided by traditional HS, i.e. selection by vector mu-

tation. The ITHS also proposes a novel method to provide an ideal compromise between diversification and intensification, thus increasing the probability of finding the true global optima. ITHS also assuages the problem involved in parameter definition by adapting these values as it progresses through. As HS has been proven to work well with combinatorial problem, ITHS should be able better results with respect to wind farm layout optimization, especially when searching in larger search spaces.

Chapter 4

Problem Definition

The problem of optimization of wind farm layout contains two issues: a) the number of turbines to place and b) the layout in which they are to be placed. To approach this problem, the wind conditions prevalent in the region are simulated followed by the estimation of the number of times and the configuration to achieve a high degree of desirability. This report is restricted to the one-dimensional optimization problem of determining the configuration of individual turbines in a wind farm.

4.1 Wind Scenarios

For any chosen geographical location, wind direction and speed are not constant factors. Wind distribution is characterised by its speed, direction and the probability of occurrence at a particular angle. But in cases of such study, three cases are considered to test the effectiveness of the optimization viz.

- Case 1: Constant wind speed and fixed direction
- Case 2: Constant wind speed with variable direction
- Case 3: Variable wind speed and direction

Case 1 For the first scenario wind is assumed to be flowing at a constant speed of 12m/s at 0° and it is the simplest model considered.

Case 2 The second scenario considers uniform wind from all 360°at a constant velocity of 12m/s.

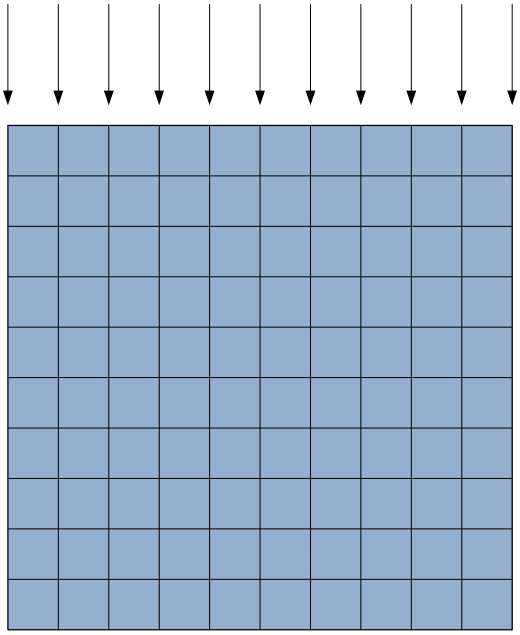


Figure 4.1: Wind Distribution for Case 1

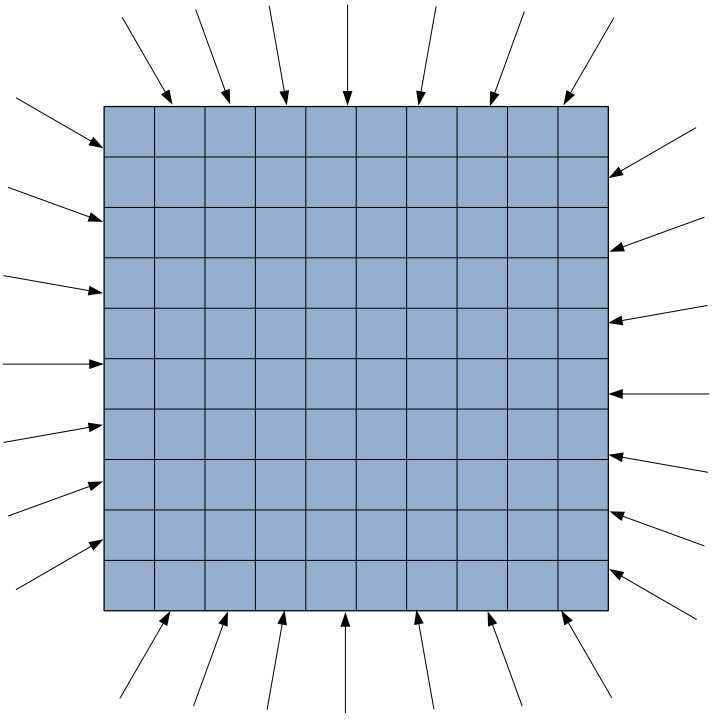


Figure 4.2: Wind Distribution for Case 2

Case 3 The third scenario is similar to the second , except for the fact that the probability of wind speed at a given angle is considered as shown by Fig. 4.3

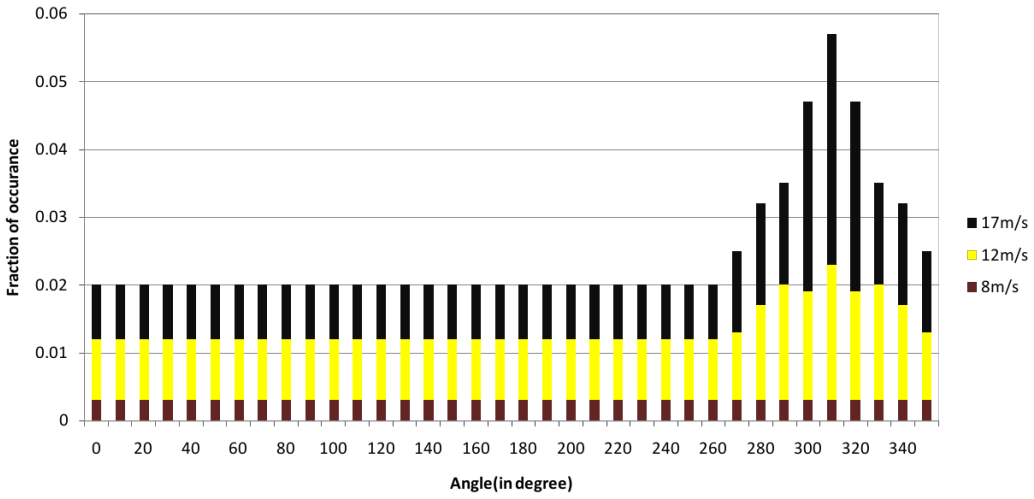


Figure 4.3: Distribution of wind speed

4.2 Objective Function

The goals of this study are

- Explore the effectiveness of the optimization algorithm,
- Obtain the optimized layout for a wind farm with boundary constraints and
- Explore methodologies to increase efficiency of off-shore wind farms

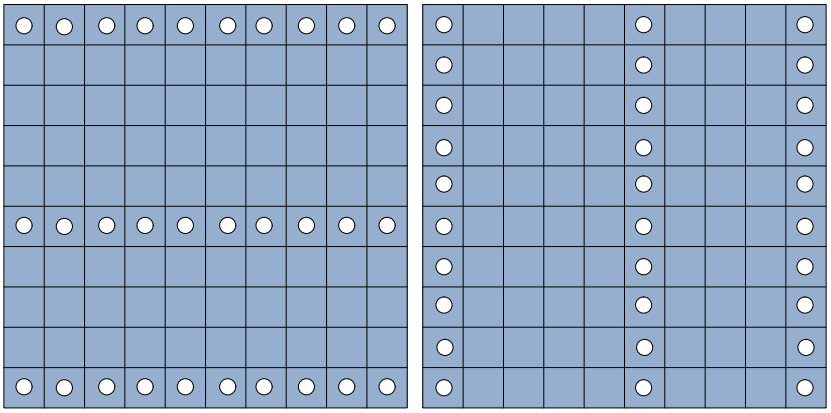
Since this is an initial study into the optimization of wind farm layouts using the *ITHS* algorithm, the same objective function as those considered in previous studies has been utilised . The objective function for the study has been chosen to minimize the cost per unit power generated given by

$$\text{Objective Function} = \frac{Cost}{Power} \quad \text{(Cost per unit power generated)} \tag{4.1}$$

where Power,P_{total} is the total power generated by the farm given the prevailing wind conditions given by Eqn. 2.17 , and cost, $Cost_{total}$ is cost of the wind farm given by Eqn. 2.19

4.3 Micro-sitting

In a general scenario, a wind farm is designed for Case 3. This however presents a unique problem during the operation phase of the wind farm. While the cost per unit power has been optimised over 360°with varying probabilities of wind speeds, it does not however guarantee maximum power generation for a particular wind speed and direction.



(a) Wind farm optimised for 0° **(b)** Effective Layout at 90°

Figure 4.4: Case for Micro-siting

From Figure 4.4a and Figure 4.4b it is clear that an optimal layout at one wind direction, may not necessarily provide optimal power at another. Extending this line of reasoning, the layout optimised over 360°is not optimal over every angle of wind but over the entire spectrum. This results in the wind farm operating at sub-optimal efficiency for most of the year. As a result, this study explores the options of micro-siting the wind-farms once they have already been anchored in a region by means of altering the mooring cables.

Micro-siting involves a system wherein the current wind characteristics(speed and direction) are monitored through sensors and the optimization algorithm running on a low Max_{iter} is used to suggest small variations in position, within the turbines individual grid to get back some of the efficiency lost due to sub-optimal placement for the current wind characteristics. Now since mooring cables offer a limited range of motion and also taking into consideration that large displacements will increase the strain on the mooring cables, the maximum displacement achievable has been set at 20m. The system is as shown in Fig. 4.5

Since the wind farm has already been established, the objective function of cost per unit power utilised becomes obsolete. For the purposes of this study, the objective function has been

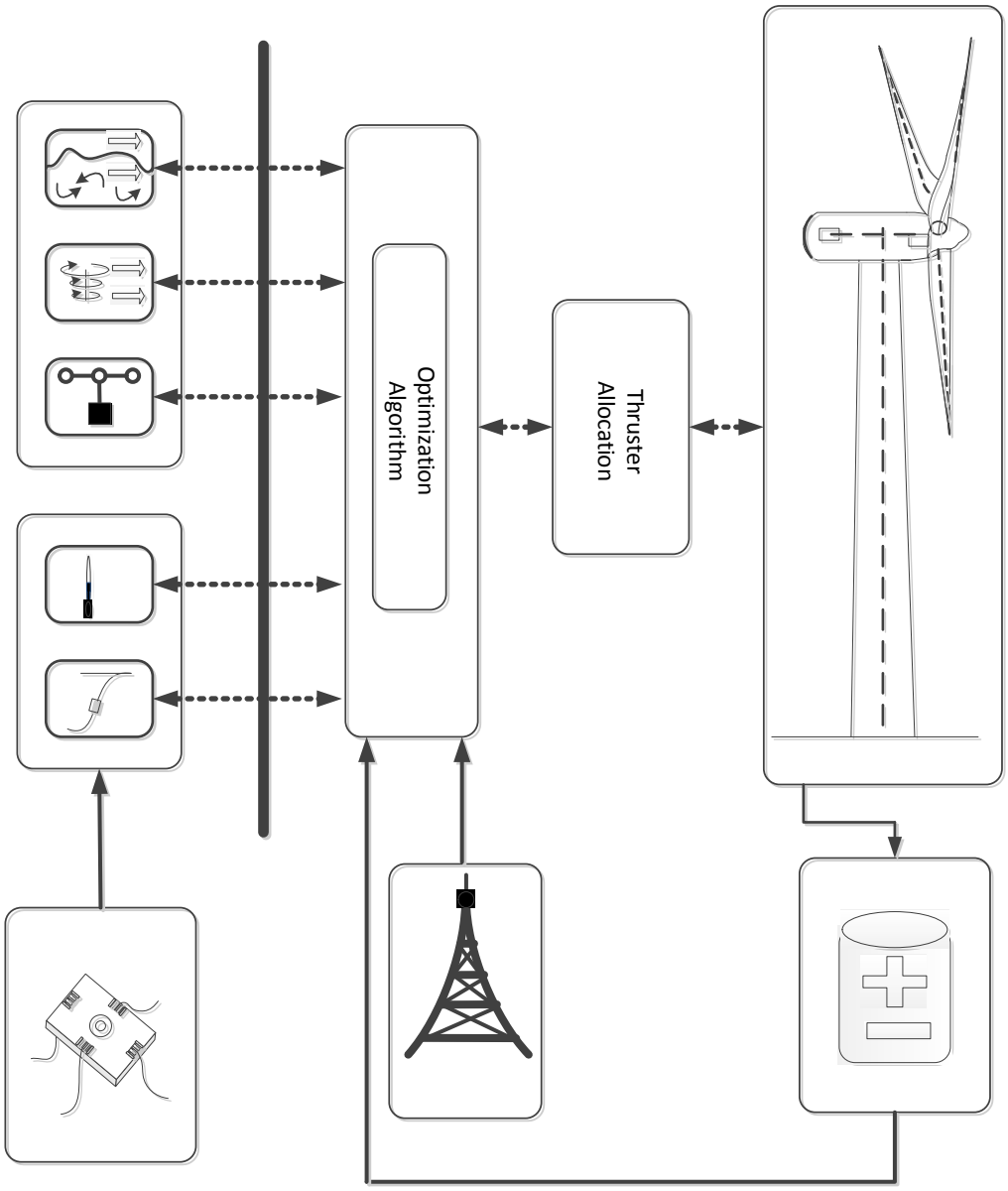


Figure 4.5: Micro-siting System

chosen as

$$\text{Objective function} = \frac{1}{\text{Power Generation capability} + \Delta\text{Power}} \tag{4.2}$$

where ΔPower = Old power generation capability - New power generation capability.

Also, the *ITHS* algorithm for the micro-siting process contains one significant change with respect to the one employed to determine the optimal layouts in the previous section; the initialization is not entire stochastic. During the operation phase of the wind farm, the micro-siting

optimization process is dependant on the current co-ordinates of individual wind turbines in the wind farm. The objective of the optimization is to re-gain some of the lost power with minimal movement in the position of the individual wind turbines themselves. As such, the initial *HM* contains the current co-ordinates of individual wind turbines in the wind farm. The rest of the solution vectors generated in the initialisation of the optimization problems are still stochastic within the allowable search space. The upper and lower limits of the search space are then defined by

$$[{}_{L}x_i, {}_{U}x_i] = [\{N,E\} \pm \{N_l, E_l\}] \tag{4.3}$$

where {N,E} are the North and East co-ordinates of the individual wind turbine and $\{N_l, E_l\}$ are the limits placed on the movements of the wind turbine platform in each direction.

This study defines the problem of wind farm layout optimization in much the same way as Grady et al. As with the choice of analytical models, the choice of the objective function for the initial placement of the wind turbines has been kept much the same as suggested by Grady et al. to provide a relevant comparison between the two optimization algorithms.The optimization problem for micro-siting has been designed to increase the power generation capabilities of the wind farm taking into consideration prevailing wind factors. Both the problems have been defined as a 1-dimensional, optimizing only the cost per unit power during the initial siting and the increase in power generation capabilities during micro-siting.

Chapter 5

Results

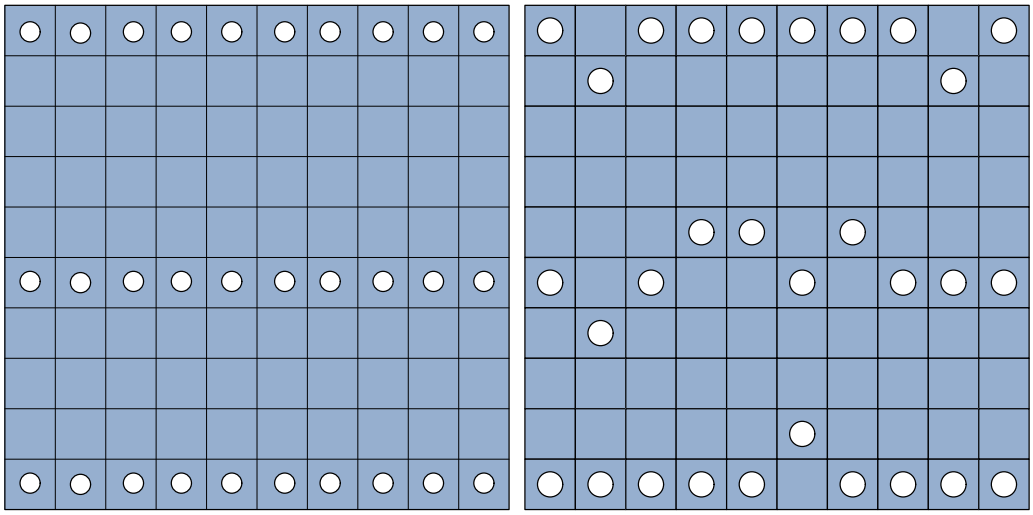
As discussed in Chapter 3, the objective of the optimization algorithm is to minimize the cost per unit power generated. But an analysis of the derivative of cost vs the number of turbines showcased that at $N > 30$, the cost of installation of turbines, increases with the increase in the number of turbines. Thus an analysis is performed for all three cases at N = 30 and at N = 30 and 39 for cases 2 and 3 as suggested by a previous study conducted by Grady et al.[8] to draw a comparison between the optimization algorithm employed in that study, GA, and the one employed in this study, ITHS.

5.1 Wind Farm Layouts

5.1.1 Case 1

In Case 1, with constant wind flow at 12 m/s at 0°. The analysis has been divided into two sub-divisions, local optimization and global optimization. Grady et al. assumed that, since wind direction wake disturbances would not propagate to neighbouring columns and that it would be sufficient to be optimize a single column of the 10x10 grid and the result applied across the grid. And this is a fair assumption if the distance between individual columns is comparatively large. Without making the above assumption this study ran the *ITHS* algorithm to find the solution for optimal placement. The results are shown in Figure 5.1a and Figure 5.1b. It is to be noted that for the same number of turbines, the previous study was able to perform better with a 1.3% margin in the objective function.

Under the assumptions made by Grady et al, the results obtained through ITHS is identical



(a) Grady **(b)** ITHS(Globally Optimised)

Figure 5.1: Case 1 Wind Farm Layout

and does not offer any advantage to either study. The results shown in Table 5.1 compares the performance of ITHS,HS and IHS for globally optimised solutions. The first case may also be chosen as a validation of the optimization algorithm as it is able to optimize to expected results for Case 1, and Case 2 and 3 are only extrapolations over 360°with constant and varying wind speeds.

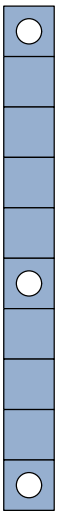


Figure 5.2: Single Column Optimised ITHS

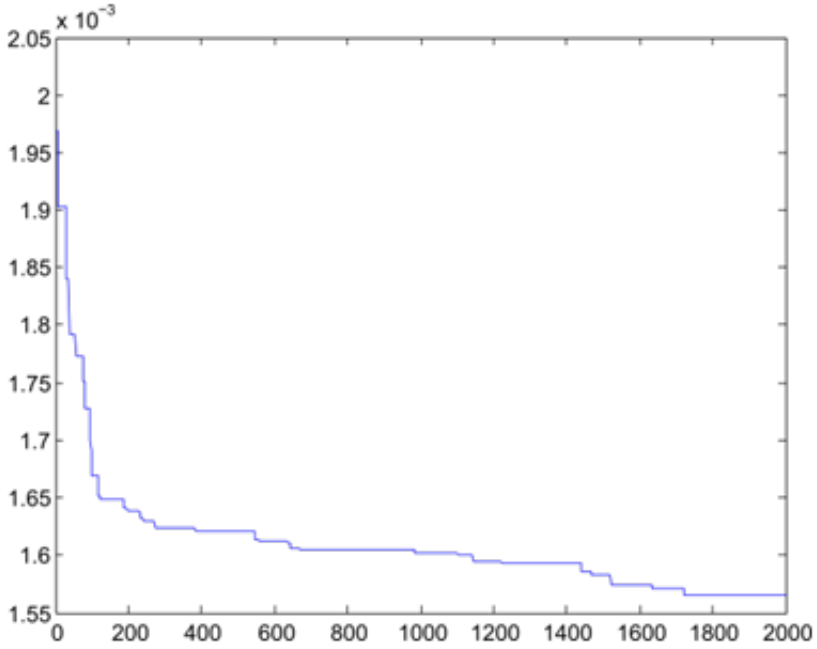


Figure 5.3: Objective Function Convergence for Case 1

	ITHS	GA	HS	IHS
Number of turbine	30	30	30	30
Objective function	0.001565	0.0015436	0.001581	0.001581
Power generated	14113	14310	13967	14094
Efficiency(%)	90.75	92.015	89.81	90.62
Increase in efficiency(%)		1.395876143	-1.03451	-0.13463

Table 5.1: Comparison of Case 1 Result

5.1.2 Case 2

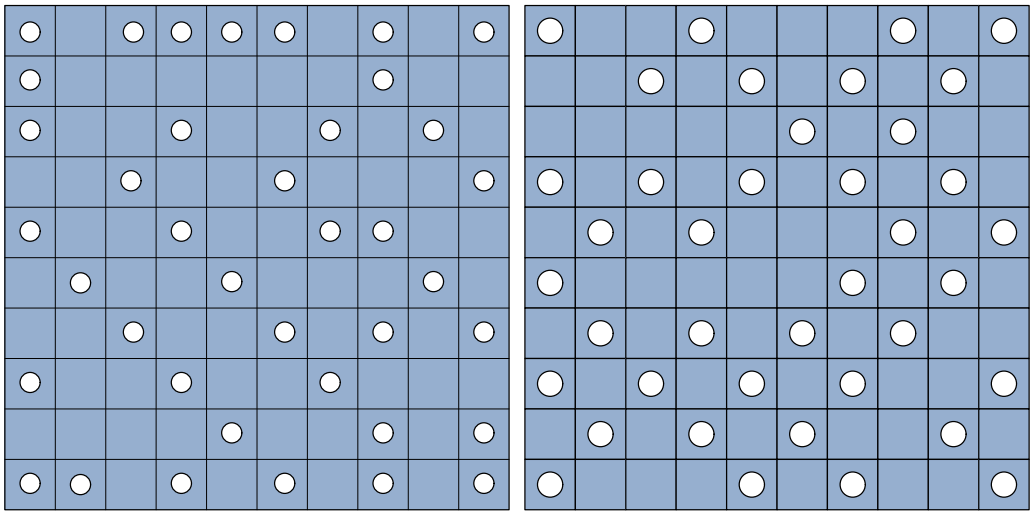
Since the derivative of cost increases after N = 30(Please refer Figure. 2.4), the optimization algorithm was first run at N = 30 to estimate the optimal objective function for the same. Since this is the first phase of the study, this report shall not determine the optimum number of turbines to be placed but move on N = 39, to draw a comparison between the performance of ITHS and GA.

	N=30	N=39
Objective Function	0.001579	0.001488
Power(kW)	14539	18094
Efficiency(%)	93.49	89.49

Table 5.2: Objective Function at N=30 and N=39 for Case 2

From Table 5.2,it is evident that even though the farm efficiency is higher in case of N =

30, the objective function and thus, cost per unit power is more desirable at N = 39. What this implies is, that the rate of change for cost of installation of a single turbine is less than the increase in power obtained by the installation of an extra turbine. This also brings forward the need to formulate the placement of wind turbines as a multi-dimensional optimization problem. The result of Case 2 is a symmetrical pattern, as a result of wind flowing with equal speed from all directions. The current study performs much better than the previous one, and improves the objective function by about 5%. The difficulty in the structure proposed by Grady becomes evident at many angles, mainly at 90°intervals where the presence of rows of wind turbines placed right behind each other in close intervals might result in those turbines getting almost no wind and thus reduce their capability to generate any power. The structure suggested by ITHS on the other hand at these angles provides adequate spacing between turbines. Another feature evident from the Grady's study is from -90°to +90°, there is a wall of turbines encountering the onward wind flow without the presence of any channels, whereas the ITHS offers channels at all orientation, thus increasing the capability to produce power. The ITHS though seems to fail at positive 45 and 135 degress as it does line up a large number of turbines in a straight line, a feature present in Grady's study too. But, the present study offers a large empty region in the centre, symmetrical from all angles wherein wind speeds may increase again.



(a) Grady **(b)** ITHS

Figure 5.4: Case 2 Wind Farm Layout

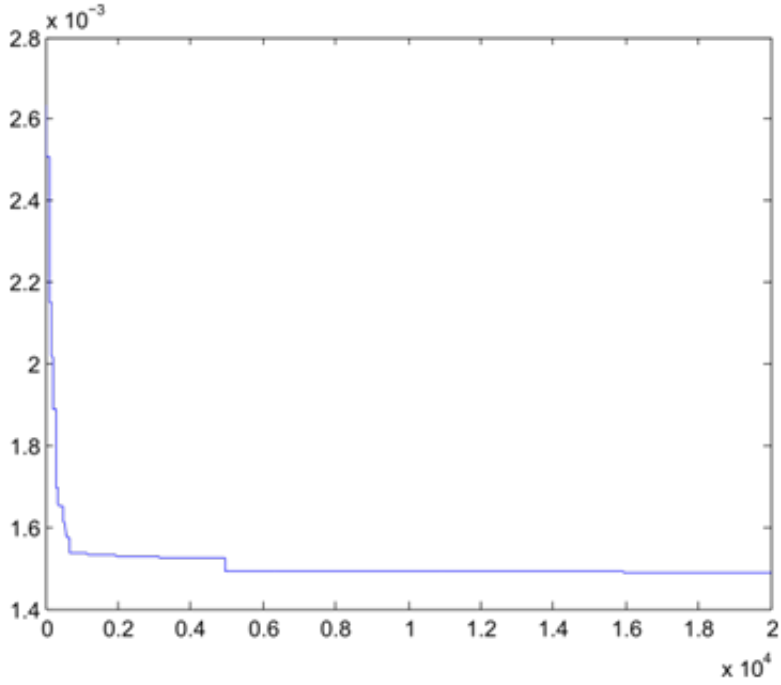


Figure 5.5: Objective Function Convergence for Case 2

	ITHS	GA	HS	IHS
Number of turbine	39	39	39	39
Objective function	0.001488	0.001566	0.0015	0.0014916
Power generated	18094	17220	17919	18049
Efficiency(%)	89.49	85.174	88.63	89.27
Increase in efficiency(%)		-4.83033	-0.96717	-0.24870123

Table 5.3: Comparison of Case 2 Result

5.1.3 Case 3

As in the previous case, the algorithm is run first at N = 30 to determine the optimal layout at the point where the cost of installation increases with an increase in the number of turbines.

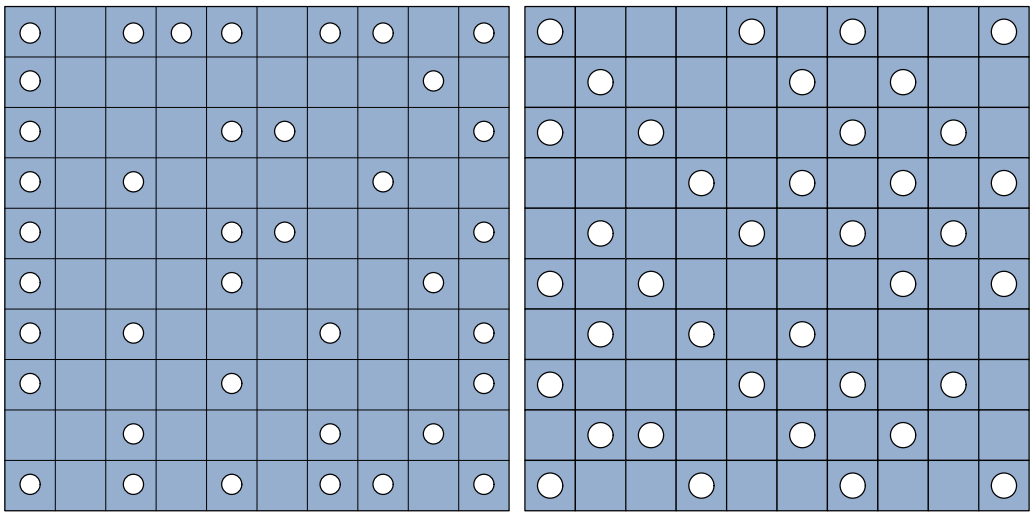
	N=30	N=39
Objective Function	0.0008044	0.000786
Power(kW)	27446	34227
Efficiency(%)	90.45	89.49

Table 5.4: Objective function at N=30 and N=39 for Case 3

Again, it is evident that even though the farm is operating at a higher efficiency, it does not necessarily guarantee optimal objective function. When N is chosen to be 30, the best result was obtained with a farm efficiency of 90.45% but its objective function is comparable to previous

studies and has been improved significantly by employing the ITHS algorithm.

In Case 3,the structure is similar to Case 2 but with preference to areas with high probability of winds with high speeds. To get a perspective of this, notice from Figure 4.3 that the probability of encountering high wind speeds is maximum at 310°, and in accordance with that there are large empty areas in which the wind speeds can be revived at approximately 310°. In contrast, the layout for Case 2 had a central empty region symmetrical from all angles. The layout suggested by Grady et al. encounters the same problem as in Case 2, wherein a wall of turbine encounters the oncoming wind, without much consideration for turbines in the inner regions.



(a) Grady **(b)** ITHS

Figure 5.6: Case 3 Wind Farm Layout

	ITHS	GA	HS	IHS
Number of turbine	39	39	39	39
Objective function	0.000786	0.000803	0.000895	0.000826
Power generated	34227	32038	30739	32590
Efficiency(%)	86.67	81.13	77.87	82.53
Increase in efficiency(%)		-6.3955357	-10.1908	-4.78277

Table 5.5: Comparison of Case 3 Result

5.2 Micrositing

This section details the improvements in power generation capabilities of the achieved by micro-siting. Since the optimization process for micro-siting is in the operational phase of a wind farm, it is quintessential for the optimization problem to be able to converge on its optima in a

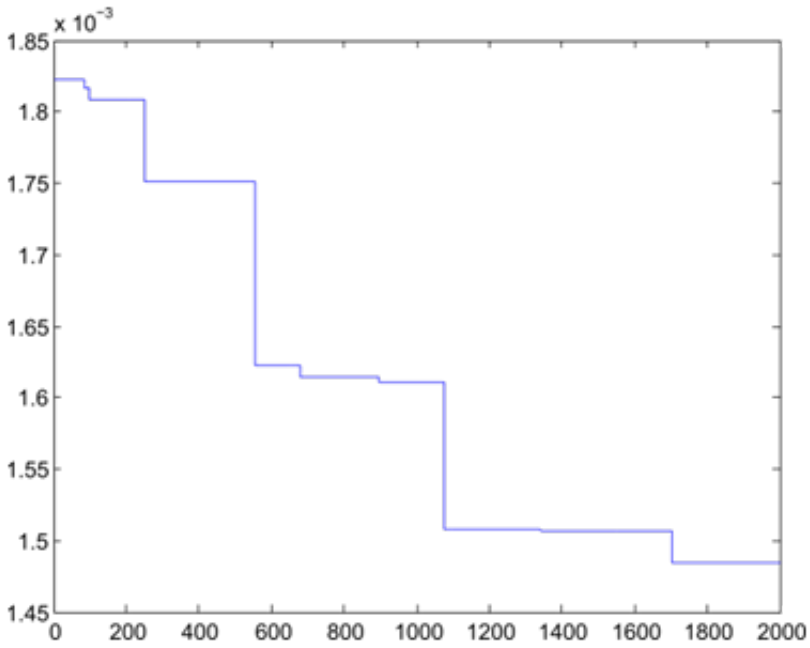


Figure 5.7: Objective Function Convergence for Case 3

relatively short time span. The search space of the optimization algorithm is restricted to $\pm$20m from the center of the grid to take into account

- Strain on the mooring cables
- Small area of movement allowed by fixed mooring cables.

The optimzation process was run for four scenarios

- Harmony Memory =100 and Max_{Iter}=300
- Harmony Memory =100 and Max_{Iter}=200
- Harmony Memory =10 and Max_{Iter}=200
- Harmony Memory =10 and Max_{Iter}=300

Table 5.6 shows the results achieved for the cases listed above. The initial placement considered for this scenario is the optimal placement obtained by employing the ITHS algorithm in Case 3. The optimization process is a variation of the Case 1, where in the algorithm optimises the power generation capabilities of the wind farm at a fixed angle but unlike the previous section, the the angle of incident wind and its speed is determined by combination of wind angle

measurement and the speed determined by Figure 4.3[1].

Angle(°)	MS[2]=100,I[3]=300	MS=100,I=200	MS=10,Iter=200	MS=10,I=300
10	2.702702703	2.702702703	2.702702703	2.702702703
20	2.702702703	2.702702703	2.702702703	2.702702703
30	2.702702703	2.702702703	2.702702703	2.702702703
40	0	0	0	0
50	0	0	0	0
60	2.702702703	2.702702703	2.702702703	2.702702703
70	2.702702703	2.702702703	2.702702703	2.702702703
80	6.27778141	4.482155463	8.442460685	6.313851069
90	4.807009962	9.251024192	2.250661815	0.014985527
100	6.530192192	4.579911662	6.699098358	9.879040848
110	4.152531254	3.309012607	4.143255755	3.610414689
120	0.045532885	0	0.104802563	0.380359461
130	2.631715798	2.14317702	1.143255004	3.037935465
140	2.065016902	1.620870839	4.67019436	3.524610481
150	0.121108653	0.03913001	0.294568692	0.342632774
160	3.364631803	3.514958514	3.345524281	3.342352805
170	9.600973843	9.954429	12.08841111	12.26247605
180	0	0	0.177144537	0.155281523
190	9.863746143	8.955013656	11.65142071	13.86773737
200	4.558326152	3.761243929	3.733090726	3.969608829
210	0.125672891	0.275519837	0.179408729	0.195772945
220	1.595490674	2.294628478	4.980145841	3.273265113
230	2.076931575	4.193928404	3.125802832	3.235521616
240	0.173217343	0.136007214	0.205891111	0.260378713
250	4.873149055	4.787458357	5.741627824	5.300264043
260	3.795580546	3.85433095	3.347229415	4.193652594
270	0	0	0	0
280	2.702702703	2.702702703	2.702702703	2.702702703
290	2.702702703	2.702702703	2.702702703	2.702702703
300	2.702702703	2.702702703	2.702702703	2.702702703
310	0	0	0	2.702702703
320	0	0	0	0
330	2.702702703	2.702702703	2.702702703	0
340	2.702702703	2.702702703	2.702702703	2.702702703
350	2.702702703	2.702702703	2.702702703	2.702702703
360	0	0	0	0
Average Improvement(%)[4]	2.677453856	2.691181385	2.94593678	2.969163101
Time(s)[5]	221	167	126	178

Table 5.6: Micro siting Results

[1] In a real case scenario the wind speed would not be a function of the graph, but a measurement of actual wind speed

[2] Memory Size

[3] Max_{iter}

An average improvement of 2.7% was observed in the power generation capabilities over all the 360 degrees. But more importantly an increase of 13% was achieved in the 150-200°region and in some cases an increase in power generation of more than 3% was also achievable. Translated into the a real world model, of excluding the scenario of 0% increase, the average increase in percentage is 3.5% over 360°.

In micro-siting optimization, another critical factor to be considered, is the computational time required to obtain optimal layouts. For the chosen scenario, the optimization algorithms were able to optimize the layout between 125-225 seconds over all 36 sectors[6]. The average computational time over a single angle is approximately 3.47-6.25s. Since wind speed and angle at wind farm scales are variables with very large time constants, the case to utilize micro-siting optimization is very strong.

ITHS was able to provide significantly better solutions when encountering large search spaces and generic wind conditions. ITHS was also able to perform quite well when trying to find the global optima in Case 1 when compared to the the layout extrapolated from local optima in the previous study. Micro-siting has also shown to increase the efficiency of the wind farm by about 3%, though this is largely dependant on the freedom of movement available to the individual turbine. The study was able to show better results when compared to Grady et al. and has also provided a convincing case to pursue micro-siting to increase wind farm efficiencies.

[4]Including 0 increase scenario

[5]Over 360°

[6]These values indicate time taken on a quad core second generation i5 processor with 4GB ram operating the Windows 7 OS. Computation times may vary over different computers and operating systems

Chapter 6

Simulation and Hardware Implementation of Prototype

This study has also begun the implementation of a scaled down prototype modelled along with modelling on the Simulink platform. Hardware implementation has gone through two revisions. The second version of the prototype has been equipped with position mooring control capabilities along with wireless communication. The objective of building was to model a small scale wind farm to test the validity of ”micro-siting”. To this end, two avenues of position control are being explored currently

- Mooring Lines
- Thruster Control

A model off-shore wind turbine platform was constructed according to the dimensions given in Table 6.1. Please refer to the Appendix A.5 for further details regarding the structure of the prototype.

Turbine Height	300mm
Platform Dimentions	255x255x90mm
No of Motors	5
Weight of Platform	1500 gms
Weight of platform[1]	2500 gms

Table 6.1: Prototype Dimensions

[1] With Battery

Each prototype contains

- 4 motors to simulate mooring winches
- 1 motor to orient the wind turbine in the direction of the wind
- 1 Xbee transceiver for communication
- 2 Ultrasonic sensors for distance measurements in N and E
- 1 Magnetic Encoder to determine the direction of wind flow

The structure of the proposed prototype wind farm is as shown in Figure 6.1. Each wind turbine sends the wind characteristics it is currently experiencing along with its position co-ordinates to the base module, by means of the Xbee module. Each platform sends a data stream containing the current wind angle and its current position co-ordinates. The wind speed is measured for the entire wind farm and the angle is averaged over all the individual wind turbines. The base module, then stores the data in an external storage module, to be analysed by the base station running the optimization algorithm. If the optimization algorithm deems the change in power output worth the energy investment, the individual wind turbines are instructed to move to the new co-ordinates. The flow chart for which is given by

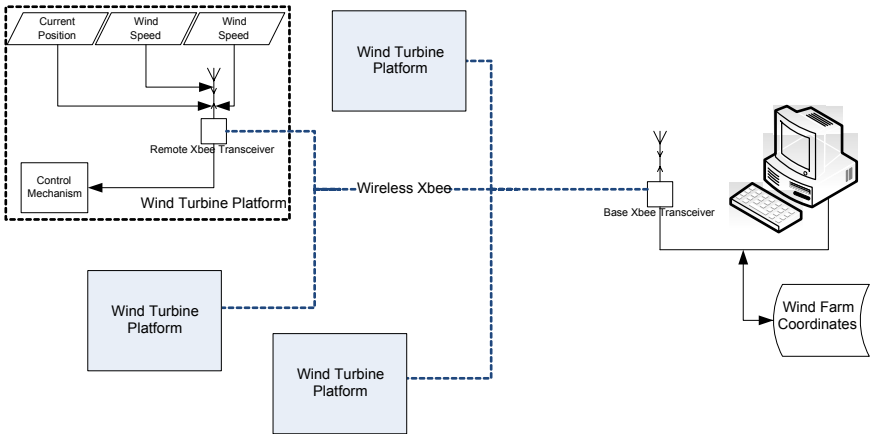


Figure 6.1: Prototype Wind Farm Architecture

The discussed steps have been implemented through a simple 4-line mooring system with a proportional controller(Please refer to Appendix A.2 for details regarding the implementation regarding the 4-line proportional controller). The next section discuses the implementation of thruster control with PD control to allow for greater freedom in movement of the platform.

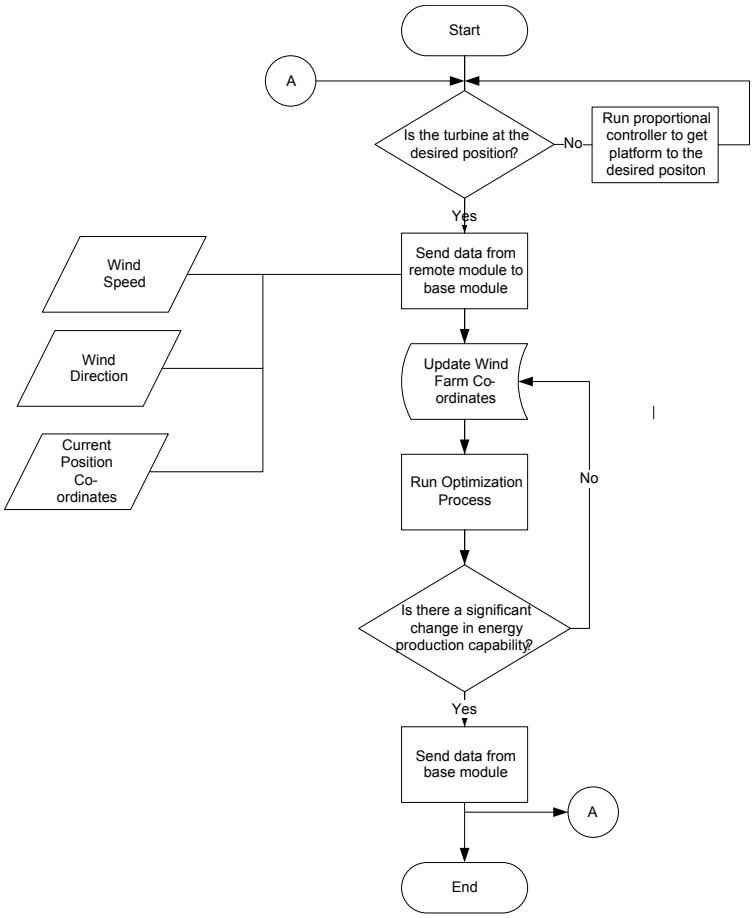


Figure 6.2: Flowchart for Micro-siting

6.1 Simulation Modelling

6.1.1 Dynamic Model

To describe the motion of an off-shore mobile platform, two references frames are used, a local geographical earth-fixed frame and a body fixed frame. The position of the platform is given by three components viz. the north-east position and the platform's heading relative to the local geographical reference frame. This is given by a [3x1] matrix defined by

$$\eta = \begin{bmatrix} n \\ e \\ \psi \end{bmatrix} \tag{6.1}$$

where η is the position matrix of the platform. The corresponding velocity vector is given by $\nu = [u, v, r]^T$ with surge and sway velocities(u,v) and yaw rate r being the three components.

The body fixed co-ordinates u and v are the time derivatives of the position of the platform and yaw-rate is the time derivative of the heading of the platform.

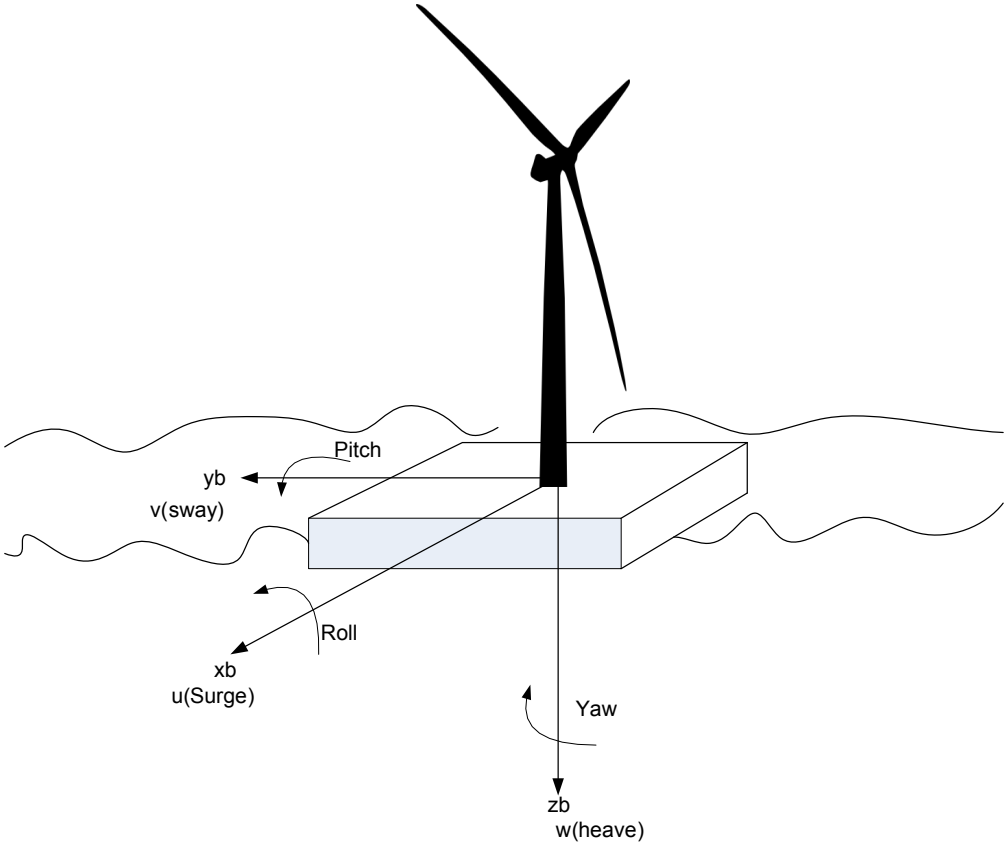


Figure 6.3: Co-ordinate System for Position Control

For the purposes of this study, the motion of the platform is confined to the horizontal plane and the original 6-DOF motions suggested by Fossen[20] is reduced to 3-DOF. The model dynamics of a platform is given by

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \tag{6.2}$$

$$(M_{rb} + M_A)\dot{v} + C_{RB}(v)v + d(V_{rc}, \gamma_c) = \tau_{control} + \tau_{environment} \tag{6.3}$$

$$\dot{\eta} = J(\eta)v \tag{6.4}$$

The terms on the right hand side of Eqn. 6.3 represent the force vectors acting on the platform and is given by $\tau = [X, Y, N]^T$, where X is the surge force, Y is the sway force, and N is the yaw moment. For a platform with the origin of the body frame lies at the center of gravity

such that $y_G = 0$, the associated rigid body matrices reduce to

$$M_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & mx_G \\ 0 & mx_G & Iz \end{bmatrix} \quad C_{RB}(v) = \begin{bmatrix} 0 & 0 & -m(x_G + v) \\ 0 & 0 & mu \\ m(x_G r + v) & -mu & 0 \end{bmatrix} \tag{6.5}$$

and the added mass matrix reduce to

$$MA = \begin{bmatrix} -X_{\dot{u}} & 0 & 0 \\ 0 & -Y_{\dot{v}} & -Y_{\dot{r}} \\ 0 & -Y_{\dot{r}} & -N_{\dot{r}} \end{bmatrix} \tag{6.6}$$

The C_{RB} relates to the Coriolis-Centripetal term which appears as a consequence of expressing the equations of motion in body-fixed co-ordinates. Since the velocities the platform will be operating at are small, the matrix C_{RB} can be ignored for control design[21]. The matrices for the prototype model were obtained from flow modelling simulation in ANSYS for a wave period of 1Hz with a max wave height of 2 cm Please refer to Appendix A.3 for further details regarding the rigid body and damping matrices.

6.1.2 Control

Most existing systems use either a series of PID controls, with each controller controlling a single DOF or simple P or PI controllers for automatic heading and depth control as it is very difficult to estimate the velocity vector. By using vehicle kinematics along with gravity compensation, a set point regulation in terms of Lyapunov stability was suggested by Fossen[20] if input $\tau_{control}$ is transformed according to

$$\tau = J^T(\eta)\tau_{PID} + g(\eta); \tag{6.7}$$

The control law can then be given by

$$u = B^{-1}\tau \tag{6.8}$$

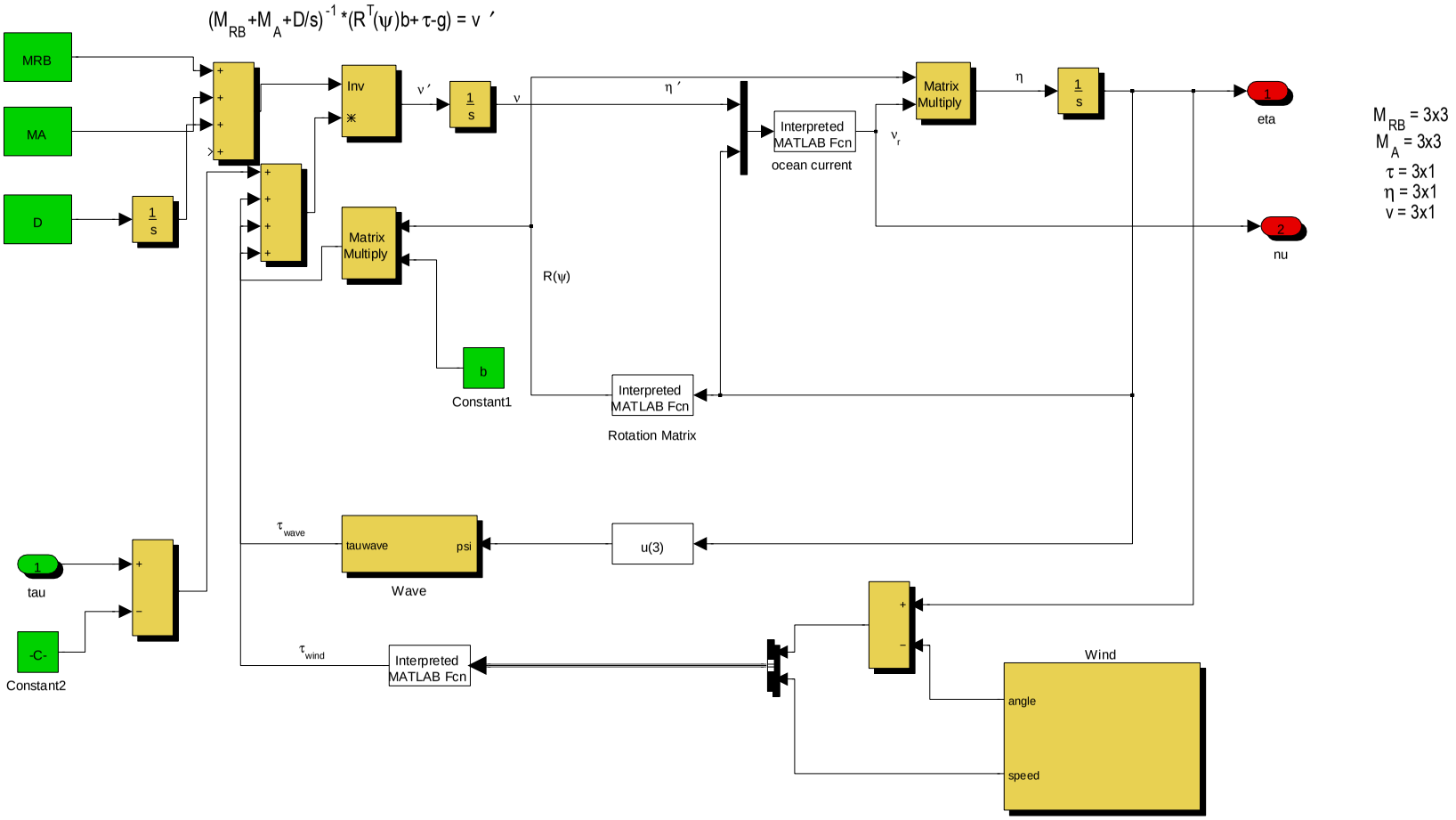


Figure 6.4: Simulation Model of the prototype

The model under consideration here is

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = Bu \tag{6.9}$$

where $\eta \in \Re^n, v \in \Re^n$ and $u \in \Re^r$. In the model, input matrices B and gravitational forces $g(\eta)$ are known whereas M,C and D are unknown. Assuming B to be invertible and choosing a PD control law with error $e = \eta_{desired} - \eta$ and the inclusion of the term $g(\eta)$ to compensate for gravity and buoyancy, the control input is given as

$$u = B^{-1}[J^T(\eta)K_p e - K_d^* v + g(\eta)] \tag{6.10}$$

The control law is motivated from time differentiation of a Lyapunov function candidate

$$V(v,e) = \frac{1}{2}(v^T M v + e^T K_p e) \tag{6.11}$$

Differentiating the above Lyanpunov function yields

$$\dot{V} = v^T[M\dot{v} - J^T(\eta)K_p e] \tag{6.12}$$

because $\dot{e} = -\dot{\eta} = -J(\eta)v$. Substituting the dynamic equation of the model into the Eqn.6.12 we have

$$\dot{V} = v^T[Bu - D(v)v - g(\eta) - J^T(\eta)K_p e] \tag{6.13}$$

On choosing $K_p = K_p^T > 0$ and $K_d^* > 0$

$$\dot{V} = -v^T[D(v) + K_d^*]v \leq 0 \tag{6.14}$$

The above equation suggests that rate of attaining set-point is achieved passively by Damping matrix D and actively by virtual damping matrix K_d^*.

The next step is to verify $\dot{V}$ does not get stuck at 0. When $\dot{V}$ is 0, Eqn. 6.14 implies $v = 0$

$$\dot{v} = M^{-1}J^T(\eta)K_p e \tag{6.15}$$

Consequently $\dot{v}$ will always be non-zero whenever error is not 0 and $\dot{V} = 0$ only if e=0. Thus the system cannot get stuck and the system will always converge towards to the desired position vector.

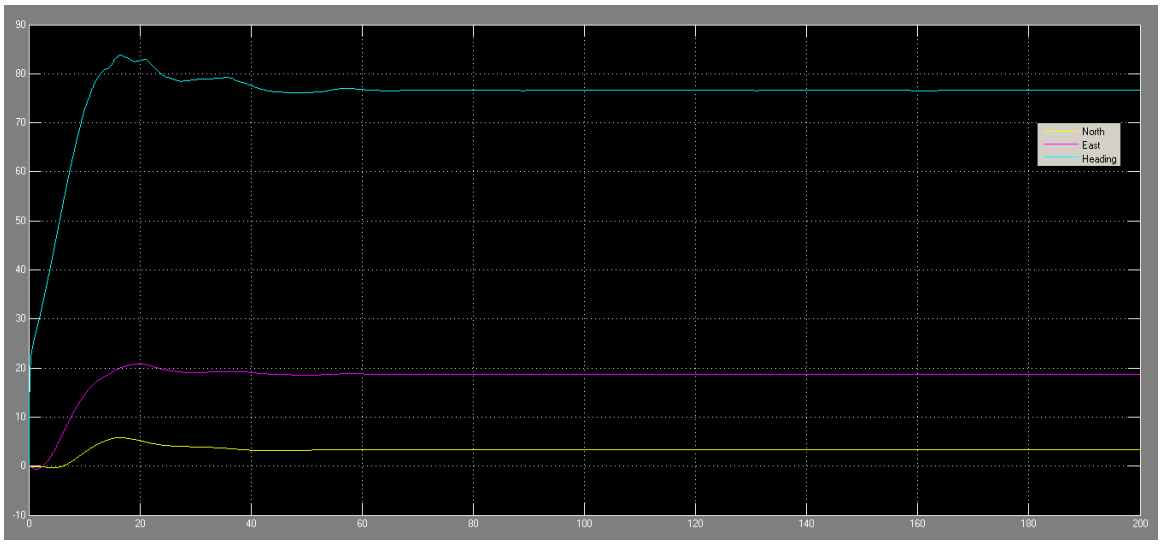


Figure 6.5: Platform response for given setpoint point(n,e,ψ)

Simulation trials were run for the designed wave period and amplitude of 0.983s and 2cm with additional wind and current loads. Wind was scaled down by a hundred to 0.12m/s and current between 0.1m/s and 0.7m/s Figure 6.5 and Figure 6.6 show the output of the system $[n, e, \psi]$ for a setpoint of [5,10,80] and the forces required to get the system to the position respectively

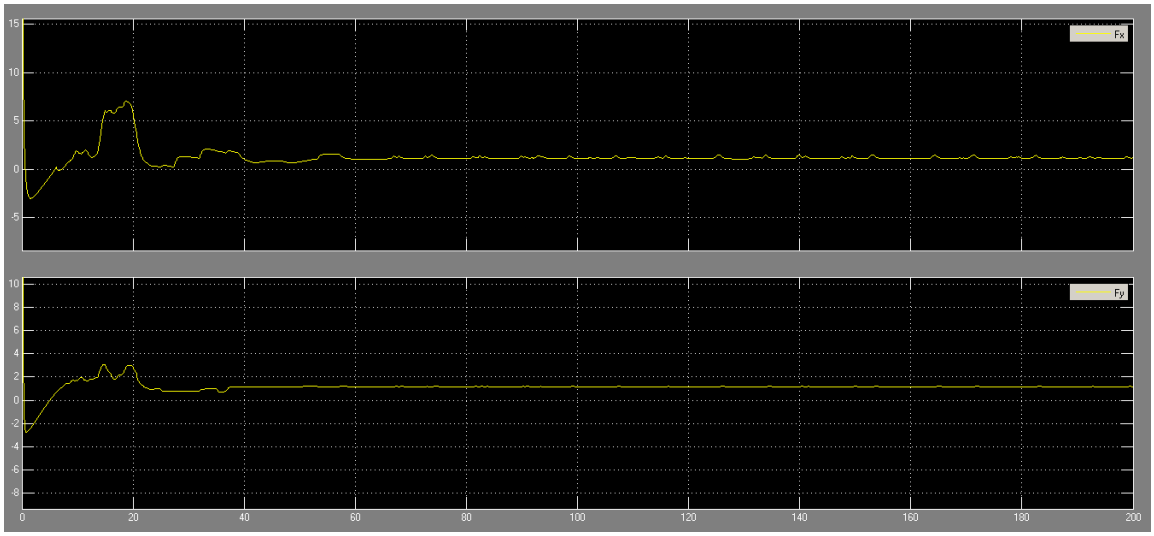


Figure 6.6: Forces required for Setpoint Regulation in X and Y planes

Forces required for position control through thrusters/propellers do not go above 5N, achievable realistically through two thrusters/propellers. With a 4 thruster/propeller system, the prototype system should theoretically be able to achieve respectable performance of attaining its setpoint in about a 60-90s.

An architecture for the wind farm micro-siting system has been developed and prototype development is being carried out by this study to verify results obtained in the previous section. The architecture for the wind farm system along with position mooring control has already been implemented and found to be performing well .Simulation results for thruster control have shown to be promising and as such, thruster assisted mooring control is currently being pursued by the study as a means of station-keeping and position control for the micro-siting. This combination of the two offers both freedom of movement through thrusters and reduction in energy consumptions through mooring lines.

Chapter 7

Future Work

Wind farm layout, unlike what has been explored here, is a multi-dimensional optimization problem where both the number of turbines and their placement need to be optimised. This study has only explored the effectiveness of the *ITHS* algorithm at three points for all cases. The true global optima for the multi-dimensional problem of optimizing both the number of turbines and their power output may lie in the search space unexplored by this study. Also this study assumes a linear wake flow model suggested by N.O.Jensen for two reasons

- To make an accurate comparison between previous studies
- To simplify the nature of study

Though, an accurate power generation model was considered, the wake flow model considered was analogous to a "first approximation". A more comprehensive model may be considered to provide a more accurate estimation of the global optima for both, the number of turbines and its power generation capabilities for the given environmental conditions.

The micro-siting problem may also be formulated as a multi-objective optimization problem by taking in to consideration not just the increase in power generation capabilities but also the energy investment required to achieve it. There also exists the possibility to form the objective function as a function of the stresses involved in moving the platform to a new positions and maintaining its position there, thus increasing the power generation capability without compromising the life span of the wind farm itself. Though position control through thrusters has been explored in this study, it may not be completely feasible owing to continuous power demands

required for station-keeping. A possible alternative could be thruster assisted position control, with mooring lines and thrusters working in conjunction with each other to achieve position control. Implementation of the above features will enable wind-farms to achieve increased degrees of efficiency and also helps increase the feasibility/economic viability of both installation and operation of the same.

Bibliography

[1] Intergovernmental Panel on Climate Change. Climate change 2007, the fourth assessment report. Technical report, 2007.

[2] Federal Ministry of Ecnomics, Federal Ministry for Environment Conservation Technology, and Nucelear Safety. Energy concepts for an environmentally friendly, reliable and affordable energy supplies. Technical report, 2010.

[3] Albert Betz. Der maximum der theoretisch mlichen ausntzung des windes durch windmotoren. *Zeitschrift fr das Gesamte Turbinenwesen*, 26:307–309, 1920.

[4] Frederick W. Lanchester. Contribution to the theory of propulsion and the screw propeller. *Transactions of the Institution of Naval Architects*, LVII:98–116, 1915.

[5] Pryor S Rathman O Larsen S Hojstrup J "Frandsen S, Barthelmie R and Thgerse M". Analytical modelling of wind speed deficit in large offshore wind farms. *Wind Energy*, 9:39–53, 2006.

[6] NO Jensen. A note on wind generator interaction. Riso National Laboratory, Roskilde, Denmark, 1983.

[7] G. Mosetti, C. Poloni, and B. Diviacco. Optimization of wind turbine positioning in large windfarms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*, 51(1):105 – 116, 1994.

[8] S.A. Grady, M.Y. Hussaini, and M.M. Abdullah. Placement of wind turbines using genetic algorithms. *Renewable Energy*, 30(2):259 – 270, 2005.

[9] Javier Serrano Gonzalez, Angel G. Gonzalez Rodriguez, Jose Castro Mora, Jesus Riquelme Santos, and Manuel Burgos Payan. Optimization of wind farm turbines layout using an evolutive algorithm. *Renewable Energy*, 35(8):1671 – 1681, 2010.

[10] Rajai Aghabi Rivas, Jens Clausen, Kurt Schaldemose Hansen, and Leo E. Jensen. Solving the turbine positioning problem for large offshore wind farms by simulated annealing. *Wind Engineering*, 33(3):287–297, 2009.

[11] B. Saavedra-Moreno, S. Salcedo-Sanz, A. Paniagua-Tineo, L. Prieto, and A. Portilla-Figueras. Seeding evolutionary algorithms with heuristics for optimal wind turbines positioning in wind farms. *Renewable Energy*, 36(11):2838 – 2844, 2011.

[12] Grigorios Marmidis, Stavros Lazarou, and Eleftheria Pyrgioti. Optimal placement of wind turbines in a wind park using monte carlo simulation. *Renewable Energy*, 33(7):1455 – 1460, 2008.

[13] Ivan Mustakerov and Daniela Borissova. Wind turbines type and number choice using combinatorial optimization. *Renewable Energy*, 35(9):1887 – 1894, 2010.

[14] Nielsen T Soresen P. Recalibrating wind turbine wake model parameterse validating the wake model performance for large offshore wind farms. 2006.

[15] Souma Chowdhury, Jie Zhang, Achille Messac, and Luciano Castillo. Unrestricted wind farm layout optimization (uwflo): Investigating key factors influencing the maximum power generation. *Renewable Energy*, 38(1):16 – 30, 2012.

[16] Joong Hoon Kim Zong Woo Geem and G.V. Loganathan. *Simulation*, 76(2):60–68.

[17] M. Mahdavi, M. Fesanghary, and E. Damangir. An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188(2):1567 – 1579, 2007.

[18] Chia-Ming Wang and Yin-Fu Huang. Self-adaptive harmony search algorithm for optimization. *Expert Systems with Applications*, 37(4):2826 – 2837, 2010.

[19] Parikshit Yadav, Rajesh Kumar, S.K. Panda, and C.S. Chang. An intelligent tuned harmony search algorithm for optimisation. *Information Sciences*, 196(0):47 – 72, 2012.

[20] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. 1994.

[21] Thor I. Fossen and Tristian Perez. Kalman filtering for positioning and heading control of ships and offshore rigs. 2009.

Appendix A

Appendix

A.1 Optimization

This section contains all the Matlab scripts utilized to determine the wind farm layout with the exception of the optimization algorithm.

Listing A.1: Power Estimation Program

```
function f = uwflo(wf)
% Power generation estimation program
% The program estimates the power generated by the wind farm for three
% cases as applicable to the study
% Please refer "Unrestricted wind farm layout optimization (UWFLO): Investigating key factors
% influencing the maximum power generation" – Souma Chowdhury, Jie Zhang,
% Achille Messac, Luciano Castillo

%% Global variables initialisation
global windfarm
global Dj
global cost
global alpha
global U0
global choice
global winddata

switch choice
    case 1
        %% Variables
        theta = 0; %Angle of wind
        N = numel(wf); % Number of turbines
```

```
M = zeros(N,N);
chk = zeros(N-1,N);
power = zeros(1,N);

%% Wind Farm co-ordinate
coord = gridnumber(wf); %grid co-ordinates of turbines
Yi = coord(:,2);
Xi = coord(:,1);
x = cos(theta)*Xi - sin(theta)*Yi;
y = sin(theta)*Xi + cos(theta)*Yi;
windfarm = [x y];

%% Wake matrix
for i = 1:1:N-1
    for j = i+1:1:N
        deltax = x(i) - x(j);
        deltay = y(i) - y(j); %#ok<*IJCL>

        if abs(deltax) < 199 && abs(deltay) < 199 == 1
            chk(i,j) = 1; %Grid Check
        else
            chk(i,j) = 0;
        end

        Dwake = Dj + 2*alpha* deltax;

        if (deltax > 0 && abs(deltay)-Dj/2 < Dwake/2)
            M(i,j) = 1;   %Wake Matrix
            M(j,i) = -1;   %Wake Matrix
        else
            M(i,j) = 0;
        end
    end
end

%% Power Generation calculation
for l = 1:1:N
    r = find(M(:,l)>0);
    s = find(chk(:,l)>0, 1);
    if isempty(s) == 0
        U = 0;
    else
        if isempty(r) == 1
```

```
                U = U0;
            else
                U = calcvel(r,l);
            end
        end

        power(l) = 0.3*(U^3);
    end

    %% Objective function
    totalpower = sum(power);
    f = cost/totalpower;
case 2
    %% Variables
    N = numel(wf); % Number of turbines
    M = zeros(N,N);
    chk = zeros(N-1,N);
    power = zeros(10,N);

    %% Wind Farm co-ordinate
    coord = gridnumber(wf); %grid co-ordinates of turbines
    Yi = coord(:,2);
    Xi = coord(:,1);
    k=1;
    for theta = 0.1745329:0.1745329:6.28
        x = cos(theta)*Xi - sin(theta)*Yi;
        y = sin(theta)*Xi + cos(theta)*Yi;
        windfarm = [x y];

        %% Wake matrix
        for i = 1:1:N-1
            for j = i+1:1:N
                deltax = x(i) - x(j);
                deltay = y(i) - y(j); %#ok<*IJCL>

                if abs(deltax) < 199 && abs(deltay) < 199 == 1
                    chk(i,j) = 1; %Grid Check
                else
                    chk(i,j) = 0;
                end

                Dwake = Dj + 2*alpha* deltax;
```

```
                if (deltax > 0 && abs(deltay)-Dj/2 < Dwake/2)
                    M(i,j) = 1;   %Wake Matrix
                    M(j,i) = -1;   %Wake Matrix
                else
                    M(i,j) = 0;
                end
            end
        end

        %% Power Generation calculation
        for l = 1:1:N
            r = find(M(:,l)>0);
            s = find(chk(:,l)>0, 1);
            if isempty(s) == 0
                U = 0;
            else
                if isempty(r) == 1
                    U = U0;
                else
                    U = calcvel(r,l);
                end
            end

            power(k,l) = 0.3*(U^3);
        end
        k=k+1;
    end

    %% Objective function
    totalpower = sum(power);
    totalpower = sum(totalpower)/(k-1);
    f = cost/totalpower;



case 3
    %% Variables

    N = numel(wf); % Number of turbines
    M = zeros(N,N);
    chk = zeros(N-1,N);
    power = zeros(10,N);
```

```
%% Wind Farm co-ordinate
coord = gridnumber(wf); %grid co-ordinates of turbines
Yi = coord(:,2);
Xi = coord(:,1);


k1=1;
for runno = 1:1:numel(winddata(1,:));
    k=1;
    for theta = 0.1745329:0.1745329:6.28
        x = cos(theta)*Xi - sin(theta)*Yi;
        y = sin(theta)*Xi + cos(theta)*Yi;
        windfarm = [x y];
        U0 = winddata(k,runno);

        %% Wake matrix
        for i = 1:1:N-1
            for j = i+1:1:N
                deltax = x(i) - x(j);
                deltay = y(i) - y(j); %#ok<*IJCL>

                if abs(deltax) < 199 && abs(deltay) < 199 == 1
                    chk(i,j) = 1; %Grid Check
                else
                    chk(i,j) = 0;
                end

                Dwake = Dj + 2*alpha* deltax;

                if (deltax > 0 && abs(deltay)-Dj/2 < Dwake/2)
                    M(i,j) = 1;   %Wake Matrix
                    M(j,i) = -1;   %Wake Matrix
                else
                    M(i,j) = 0;
                end
            end
        end

        %% Power Generation calculation
        for l = 1:1:N
            r = find(M(:,l)>0);
            s = find(chk(:,l)>0, 1);
            if isempty(s) == 0
```

```
                    U = 0;
                else
                    if isempty(r) == 1
                        U = U0;
                    else
                        U = calcvel(r,l);
                    end
                end

                power(k,l,runno) = 0.3*(U^3);
            end
            k=k+1;
        end
        k1 = k1+1;
    end

    %% Objective function
    totalpower = sum(power);
    totalpower = sum(totalpower);
    totalpower = sum(totalpower)/((k-1)*(k1-1));
    f = cost/totalpower;


end
```

Listing A.2: Grid assignment

```
function cord = gridnumber(p)
% Calculate x-y co-ordinates of the turbine in a given grid.

length = numel(p);
x=zeros(1,length);
y=zeros(1,length);

for i=1:1:length
    x(i)= mod(p(i),10);
    if x(i) == 0;
        x(i)=1900;
    else
        x(i)= ((2*(x(i)-1))+1)*100;
    end
    y(i) = ceil(p(i)/10);
    y(i) = ((2*(y(i)-1))+1)*100;
```

```
end

cord = [x' y'];
cord  = sortrows(cord,[-2,1]);
end
```

Listing A.3: Velocity Calculation

```
function f = calcvel(i,j)
% Calculate incident velocity on any turbine

%% Initialise global variables
global windfarm
global alpha
global a
global U0
global Aj
global Dj

%% Wake model calculation
pwrcent = 0;
dwind = j;
dcord = windfarm(dwind,:);
for k = 1:1:numel(i)

    upwind = i(k);
    ucord = windfarm(upwind,:);
    xdistance = ucord(1)-dcord(1);
    Dwake = Dj + (2*alpha*abs(xdistance)); % Wake radius
    [xout,yout] = circcirc(dcord(2),dcord(1),Dj/2,ucord(2),ucord(1),Dwake/2);
    if isequalwithequalnans(xout, [NaN,NaN])==1 && isequalwithequalnans(yout, [NaN,NaN])==1
        Akj = Aj; % Complete overlap
    else
        rk = pi*Dwake^2/4;
        Ak = (rk^2*acos((xdistance^2+rk^2-Aj^2)/(2*xdistance*rk))+...
        Aj^2*acos((xdistance^2+Aj^2-rk^2)/(2*xdistance*Aj))...
        -0.5*sqrt((-xdistance+rk+Aj)*(xdistance-rk+Aj)*(xdistance+rk-Aj)*(xdistance+rk+Aj)));
        Akj= Ak; % Partial overlap
    end
    Ukj = U0*(1 - (2*a/(1+alpha*xdistance/27.87)^2));
    pwrcent = pwrcent+Akj*(U0-Ukj)^2/Aj;
end
```

```
%% Total wake energy loss
sigma = pwrcent;
f = U0 - sqrt(sigma);
```

A.2 Arduino and xbee

This section contains all the Arduino files required to setup the prototype architecture discussed in the study.

Listing A.4: Remote Module

```
#include <Servo.h>

#define encoder0PinA 2
#define encoder0PinB 3
#define anPin 8
#define anPin0 9

//Encoder Initialisation
volatile unsigned int encoder0Pos = 0;
static boolean rotating=false;
double radPos;

// Xbee Remote Module

char buff[24]; //Serial Buffer
char xpos[3]; // Desired x co-ordinates received
char ypos[3]; // Desired y co-ordinates received
Servo ser1; // Servo 1
Servo ser2; // Servo 2
Servo ser3; // Servo 3
Servo ser4; //Servo 4
int x,y; // Current x and y co-ordinates
int xtarget,ytarget;  // Desired x and y co-ordinates extracted

// Ultrasonic Initialization
int arraysize = 5;
int rangevalue[] = {0, 0, 0, 0, 0};
int rangevalue0[] = {0,0,0,0,0};
```

```
void setup(){
  Serial.begin(9600);
  // Encoder setup
  pinMode(encoder0PinA, INPUT);
  digitalWrite(encoder0PinA, HIGH);
  pinMode(encoder0PinB, INPUT);
  digitalWrite(encoder0PinB, HIGH);
  attachInterrupt(0, rotEncoder, CHANGE);

  // Ultrasonic and servo system setup
  pinMode(anPin, INPUT);
  pinMode(anPin0,INPUT);
  ser1.attach(9);
  ser2.attach(11);
  ser3.attach(10);
  ser4.attach(12);
  ser1.writeMicroseconds(1450);
  ser2.writeMicroseconds(1450);
  ser3.writeMicroseconds(1450);
  ser4.writeMicroseconds(1450);
  delay(2000);
}

void loop()
{
  while(Serial.available()>23){
    for(int c=0;c<24;c++){
      buff[c] = Serial.read(); // Gather datastream
    }
    char* indx = strstr(buff,"x2");
    for(int i=0;i<3;i++){
      xpos[i]=buff[indx-buff+(i-2)]; // Extract x desired position and convert into integer
      x = atoi(xpos);
    }
    char* indy = strstr(buff,"y2");
    for(int j=0;j<3;j++){
      ypos[j]=buff[indy-buff+(j-2)]; // Extract y desired position and convert into integer
      y = atoi(ypos);
      xtarget = x;
      ytarget = y;
    }
  }

```

```
for(int i = 0; i < arraysize; i++)
{
  rangevalue[i] = analogRead(anPin);
  rangevalue0[i] = analogRead(anPin0);
  delay(5);   //wait between analog samples
}

isort(rangevalue, arraysize);
isort(rangevalue0,arraysize);
int midpoint = arraysize/2;
int xd  =  rangevalue[midpoint];
int yd = rangevalue0[midpoint];
if (((xd <= xtarget+10)&&(xd >= xtarget-10))&&((yd <= ytarget+10)&&(y >= ytarget-10)))
{// xtarget is within the defined error region and do nothing
  ser1.writeMicroseconds(1450);
  ser2.writeMicroseconds(1450);
  ser3.writeMicroseconds(1450);
  ser4.writeMicroseconds(1450);
}

while ((xd > xtarget+10)||(xd < xtarget -10))
{// x target outside defined error region
  if (xd > xtarget+10)
  {
    ser1.writeMicroseconds(1500);
    ser2.writeMicroseconds(1500);
    ser3.writeMicroseconds(1450);
    ser4.writeMicroseconds(1450);
  }
  if (xd < xtarget -10)
  {
    ser1.writeMicroseconds(1400);
    ser2.writeMicroseconds(1400);
    ser3.writeMicroseconds(1450);
    ser4.writeMicroseconds(1450);
  }
  for(int i = 0; i < arraysize; i++)
  {// collect x distance data over 50ms
    rangevalue[i] = analogRead(anPin);
    delay(10);
  }
  isort(rangevalue, arraysize);
  xd  =  rangevalue[midpoint]; // current position
```

```
    delay(500);
  }

  while ((yd > ytarget+10)||(yd < ytarget-10))
  {
    if (yd > ytarget+10)
122 {// ytarget within defined error region
      ser1.writeMicroseconds(1450);
      ser2.writeMicroseconds(1450);
      ser3.writeMicroseconds(1400);
      ser4.writeMicroseconds(1400);
    }
    if (yd < ytarget-10)
    {
      ser1.writeMicroseconds(1450);
      ser2.writeMicroseconds(1450);
132   ser3.writeMicroseconds(1500);
      ser4.writeMicroseconds(1500);
    }

    for(int i = 0; i < arraysize; i++)
    {//collect y distance data over 50ms
      rangevalue0[i] = analogRead(anPin0);
      delay(10);
    }

142 isort(rangevalue0, arraysize);
    yd = rangevalue0[midpoint]; // current position
    delay(500);
  }

  while(rotating) {
    delay(4);
    // When signal changes we wait 4 milliseconds for it to
    // stabilise before reading (increase this value if there
    // still bounce issues)
152 if (digitalRead(encoder0PinA) == digitalRead(encoder0PinB))
    {
      encoder0Pos++;
    }
    else {
      encoder0Pos--;
    }
```

```
        radPos = encoder0Pos*0.0174; //convert to radian
      rotating=false; // Reset the flag back to false
    }
    Serial.print("a");
    Serial.print(radPos);
    Serial.print("xa2");
    Serial.print(x);
    Serial.print("ya2");
    Serial.print(y);
}

void isort(int *a, int n)
{
  for (int i = 1; i < n; ++i)
  {
    int j = a[i];
    int k;
    for (k = i - 1; (k >= 0) && (j < a[k]); k--)
    {
      a[k + 1] = a[k];
    }
    a[k + 1] = j;
  }
}

void rotEncoder(){
  rotating=true;
  // If a signal change (noise or otherwise) is detected
  // in the rotary encoder, the flag is set to true
}
```

Listing A.5: Base Module

```
char x[12];
char y[12];
char buffer[24],buff[15];

char xpos[3],ypos[3];
int xa1,xa2,ya1,ya2;


void setup(){
  Serial.begin(9600);
```

```
  delay(2000);
  while(Serial.available()>0)
  {
    while(Serial.available()>23)
    {
      for(int i=0;i<24;i++){
17      buffer[i] = Serial.read();
      }
      for(int i =0;i<24;i++){
        Serial.print(buffer[i]);
        delay(10);
      }
    }
  }
}

27 void loop(){
  while(Serial.available()>14)
  {
    for(int i=0;i<15;i++){
      buffer[i] = Serial.read();
    }
    char* indx = strstr(buff,"xa1");
    char* indy = strstr(buff,"ya1");
    if(indx!=0&&indy!=0)
    {
37    for(int i=0;i<3;i++){
        xpos[i]=buff[indx-buff+(i-3)]; // Extract x1 current position and convert into integer
        xa1 = atoi(xpos);
      }

      for(int j=0;j<3;j++){
        ypos[j]=buff[indy-buff+(j-3)]; // Extract y1 current position and convert into integer
        ya1 = atoi(ypos);
      }
    }
47  indx = strstr(buff,"xa2");
    indy = strstr(buff,"ya2");
    if(indx!=0&&indy!=0)
    {
      for(int i=0;i<3;i++){
        xpos[i]=buff[indx-buff+(i-3)]; // Extract x2 current position and convert into integer
        xa2 = atoi(xpos);
```

```
      }

      for(int j=0;j<3;j++){
        ypos[j]=buff[indy-buff+(j-3)]; // Extract y2 current position and convert into integer
        ya2 = atoi(ypos);
      }
    }
  }
}
```

A.3 Prototype Fuild Dynamic Analysis

This section contains the results of the analysis performed on the platform through ANSYS. Please note the analysis has been run at a wave amplitude of 2cm and a wave period of 0.938s

Rigid-Body Matrices

$$M_{RB} = \begin{bmatrix} 2500 & 0 & 0 \\ 0 & 2500 & 0 \\ 0 & 2500 & 0.04683 \end{bmatrix} \quad M_A = \begin{bmatrix} 1.65 & 0 & 0 \\ 0 & 1.98 & -1.22e^{-04} \\ 0 & -3.8933e^{-05} & 1.75e^{-02} \end{bmatrix} \tag{A.1}$$

Damping matrix

$$D = \begin{bmatrix} 6.551e^{-0.4} & 0 & 0 & \\ 0 & 7.1768e^{-04} & -6.3413e^{-08} & -2.6495e^{-10} \end{bmatrix} \tag{A.2}$$

A.4 Simulink Modelling

This section contains all the simulink models for 3-DOF wind platform with environmental disturbances.

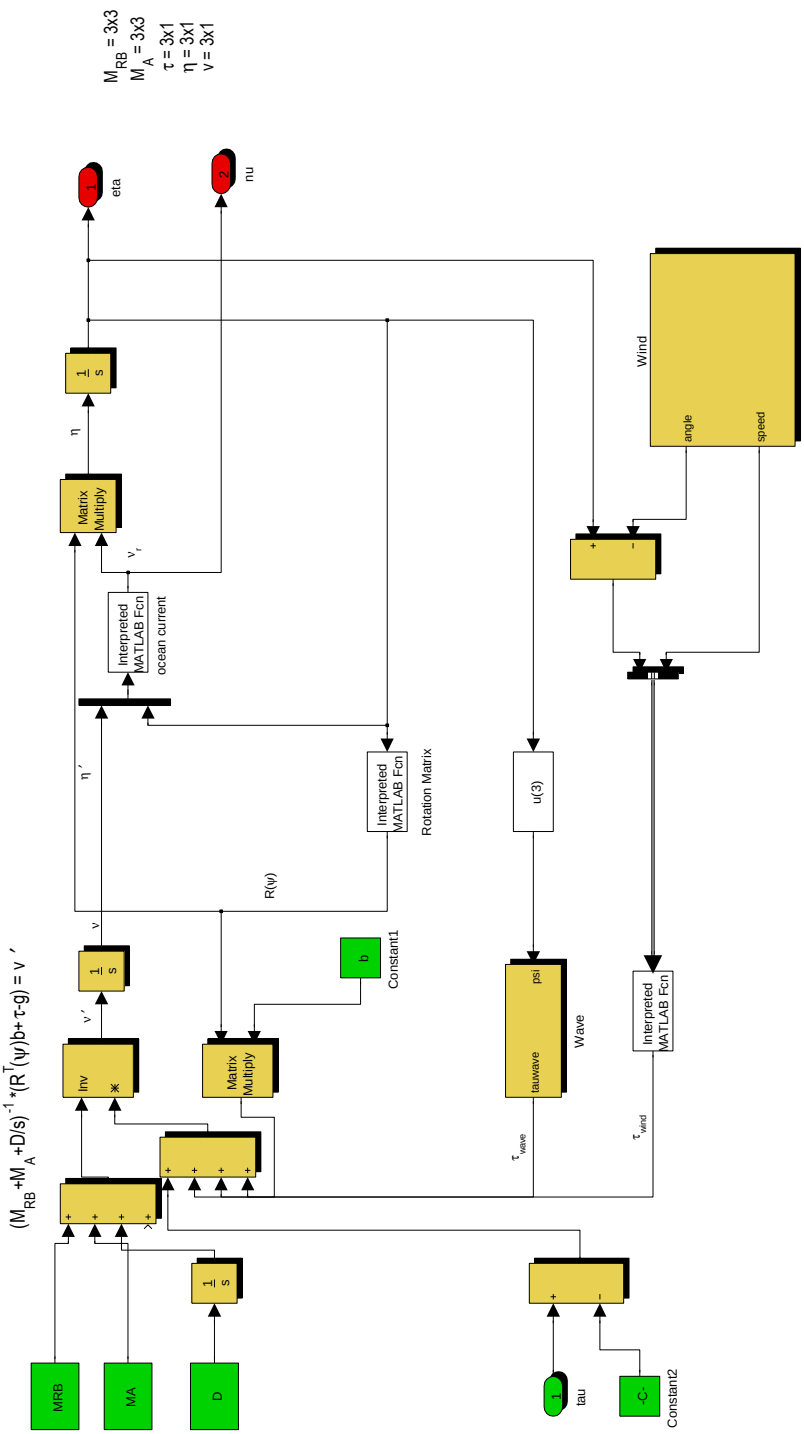


Figure A.1: Simulation Model of Wind Platform

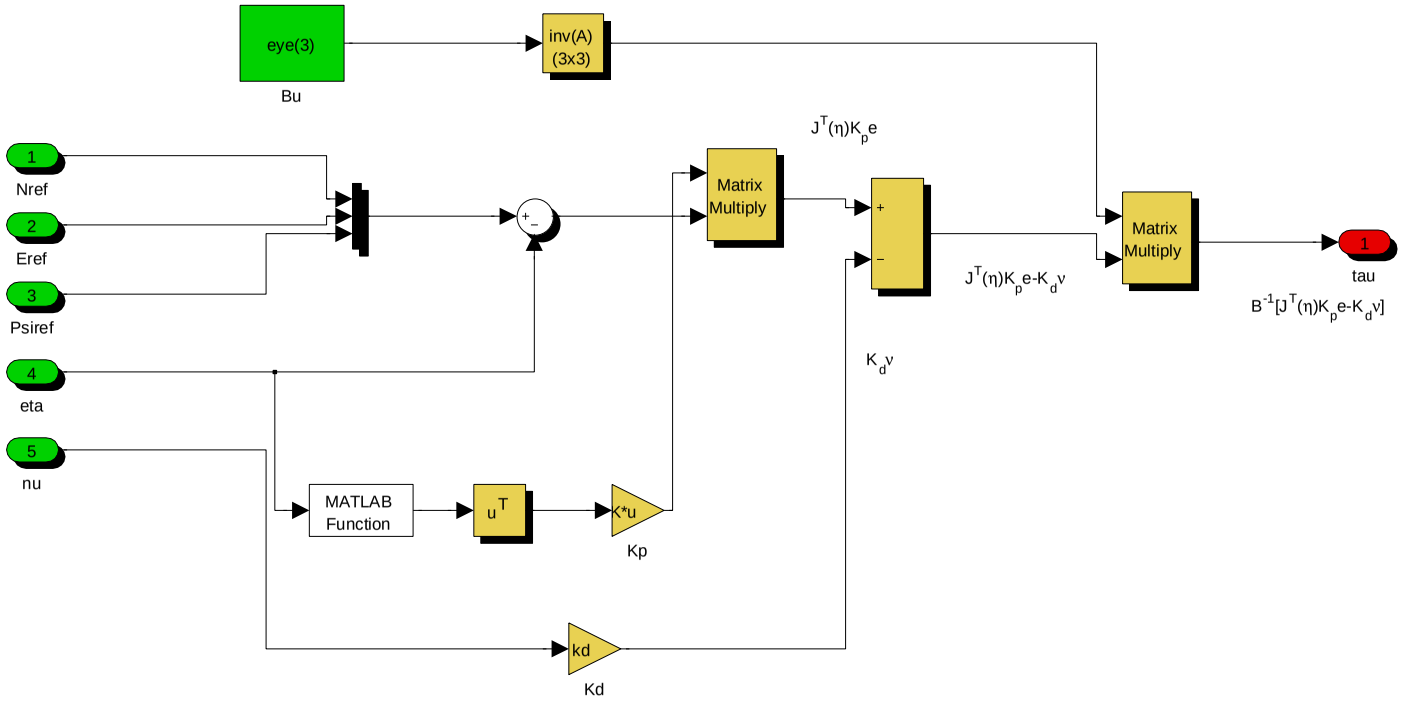


Figure A.2: Simulation Model of Wind Platform

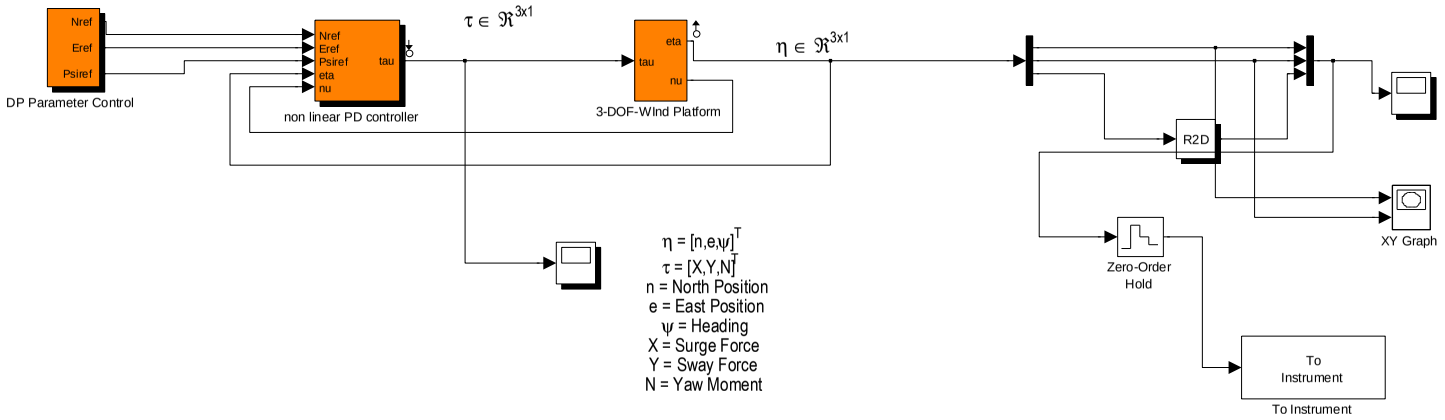
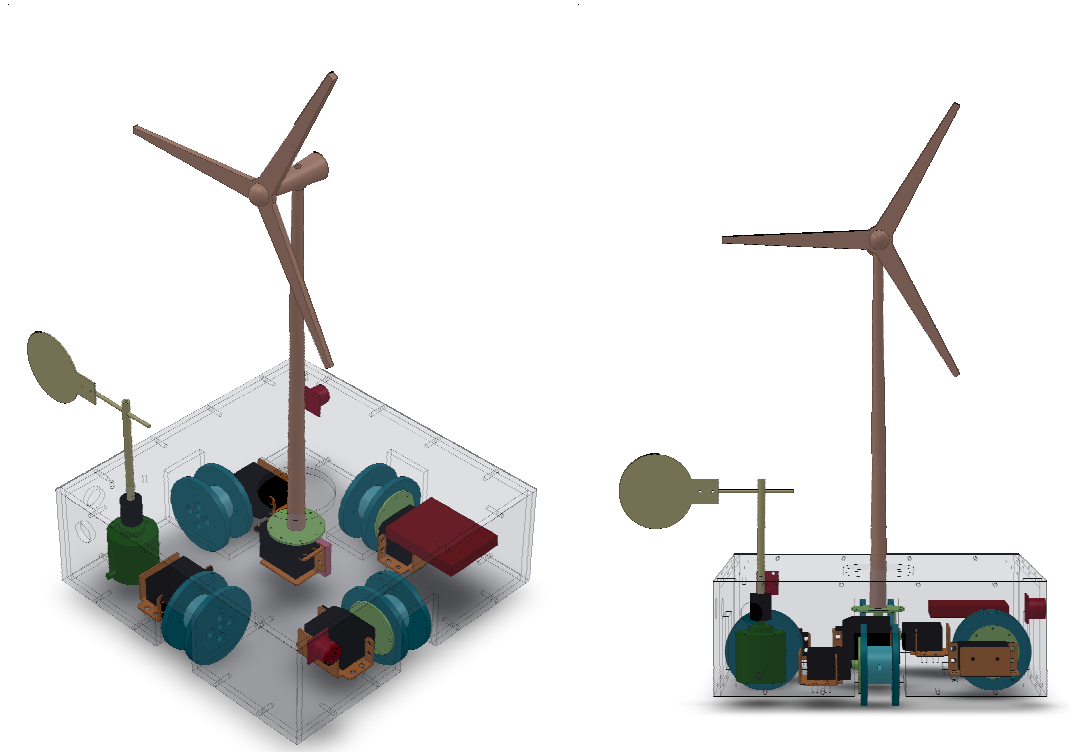


Figure A.3: Simulation Model of Wind Platform

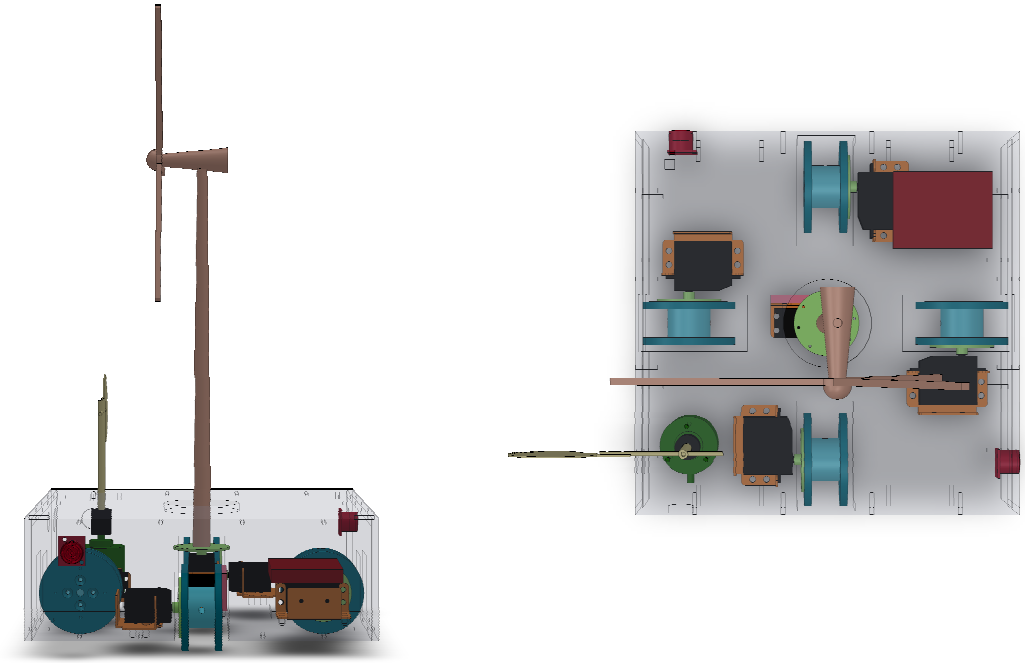
A.5 Prototype Design

This section contains all the CAD files utilized in designing the prototypes for the architecture suggested in the study



(a) Isometric View **(b)** Front View

Figure A.4: Prototype Design



(a) Side View **(b)** Top View

Figure A.5: Prototype Design contd.