DEPARTMENT OF ENGINEERING SCIENCE, UNIVERSITY OF OXFORD

Stochastic Kite Control

4th Year Project: 1004

Joshua Sutherland, St Catherine's College

24/05/2011

Supervisor: Dr. Mark Cannon

Abstract: A linearized continuous-time wind-powered kite model, computed numerically from a nonlinear model, is discretised and used to develop linear feedback and predictive controllers. All controllers are derived mathematically and used to control both the linearized and nonlinear kite models which are subject to stochastic disturbance. Based on simulation results a control strategy is selected for implementation.

Contents

Acknowledgements 4
Nomenclature 5
Acronyms 5
Mathematical notation 5
Introduction 6
Why is the control of kites being investigated? 6
Why is this a non-trivial control problem? 6
Report organisation 7
Literature review 8
Initial kite model 8
Methods of controlling systems under stochastic disturbances 9
1. Creation of the discrete kite and wind models used 10
1.1. Formation of the linear discrete-time-varying periodic kite model 10
1.1.1. Theory 10
1.1.2. Implementation 12
1.2. Wind description 12
1.2.1. Simple wind description 12
1.2.2. Truncating the wind distribution 13
2. Controller design 14
2.1. Discrete-time-varying periodic linear feedback quadratic regulator optimal control (LQOpt) 14
2.1.1. Mathematical theory 14
2.1.2. Limitations and inherent assumptions 16
2.1.3. Implementation considerations 17
2.2. Discrete-time-varying periodic H_∞ auxiliary linear feedback control (H_∞) 17
2.2.1. Mathematical theory 17
2.2.2. Limitations and inherent assumptions 19
2.2.3. Implementation considerations 19
2.3. Pre-stabilized time-varying periodic deterministic linear model predictive control (DMPC) 19
2.3.1. Mathematical theory 19
2.3.2. Limitations and inherent assumptions 25
2.3.3. Implementation considerations 25
2.4. Pre-stabilized time-varying stochastic model predictive control (SMPC) 26
2.4.1. Mathematical theory 26
2.4.2. Limitations and inherent assumptions 34
2.4.3. Implementation considerations 34
3. Preliminary data gathering needed for performance evaluation 35
3.1. Determining model sampling rate ($\boldsymbol{T}$) 35
3.2. Determining input constraints 35

4. Controller comparison 36
4.1. Performance evaluation methods and test descriptions 36
4.1.1. Testing methodology and results gathered 36
4.1.2. Systems, controllers and constraint combinations evaluated 37
4.2. Test results 39
4.2.1. Preventing kite crashing performance 39
4.2.2. Accuracy of linear model approximation 41
4.2.3. Level of infeasibility in MPC optimizations 42
4.3. Controller selection 43
Conclusions and summary 44
Report summary: Activities carried out 44
Report summary: Results 44
Recommendations for future work: Model improvements 44
Recommendations for future work: Controller improvements 45
Recommendations for future work: Testing improvements 45
Appendix A: Algorithm verification/Checks performed 46
Note on Feasible Stochastic MPC (FSMPC) code verification 47
Appendix B: MATLAB/Simulink modelling environment 48
Tips / Problems to avoid with Simulink modelling 48
Computing resources needed for implementation 48
Appendix C: Project DVD 49
Appendix D: Risk assessment 50
References 52

This project report was written by Joshua Sutherland of St Catherine's College Oxford. Help received is acknowledged below and a list of references used is provided on page 52.

Acknowledgements

I thank my supervisor Dr. Mark Cannon of the University of Oxford for proposing this project and providing excellent and intensive tuition and advice on the control of time-varying dynamic systems which are subject to stochastic uncertainty. It is very much appreciated.

I thank Prof. Moritz Diehl of Katholieke Universiteit Leuven whose paper (5) and PHD thesis (4) provided the original continuous-time kite model which is the basis of this project.

Nomenclature

Symbol	Meaning
I	The identity matrix of correct dimension for the context
$\mathbf{0}$	Zero filled square matrix or column vector of correct dimension for the context
T	Time period between model samples
N_{models}	Number of samples in a discrete periodic model
g_e	eth column of an identity matrix of the same dimension as the 4 element state vector

Table 1: Some symbols commonly used in the report (not exhaustive).

Acronyms

Acronym	Meaning
FSMPC	Feasible stochastic model predictive control (SMPC designed to remain feasible at all times)
LHS	Left hand side of an equation
MPC	Linear model predictive control
NMPC	Nonlinear model predictive control
PSMPC	Probabilistic stochastic model predictive control (SMPC with no guarantee on feasibility)
PSD	Positive semi definite matrix
RHS	Right hand side of an equation
SMPC	Stochastic model predictive control
QP solver	Class of algorithm used to solve quadratic cost function optimizations subject to linear constraints

Table 2: Some acronyms commonly used in the report (not exhaustive).

Mathematical notation

- Many of the time-varying matrices repeat every N_{models}. Notation introduced in Equation 1 (PI stands for Periodic Index) enables simulations and predictions to extend beyond a single kite loop without excessive notation.

$A_{PI(0)} = A_{PI(0+bN_{models})} \; \forall \; b \in \mathbb{Z}$	Equation 1

- The subscript $row\ n$ indicates the n^{th} block row of the matrix considered. Other subscripts identify a matrix from similar ones. The derivations clarify the differences being represented.
- Notation for time-varying block matrix generation (Equation 2) is such that if the *initial* index variable value is greater than the final index variable then the product sequence is taken to be 1.

$\prod_{\mathrm{i}}^{\mathrm{N}} \Phi_{PI(k+N-i)} = \begin{cases} \prod_{\mathrm{i}}^{\mathrm{N}} \Phi_{PI(k+N-i)}, & i \leq N \\ 1, & i > N \end{cases}$	Equation 2

- $pr()$ is used to state the probability of the expression in the brackets being true, $\mathbb{E}()$ evaluates the expected value of the bracket.

Introduction

Why is the control of kites being investigated?

Conventional wind turbines are only most efficient on the blade tips; the mast and the rest of the blade are not contributing to extracting power from the wind (8). A computer controlled kite however could be flown in a trajectory of that of the blade tip of a conventional turbine (Figure 1) or in another more optimal trajectory (with regard to power generation and fatigue) (8). A further advantage is the ability to access less variable and stronger high altitude air flows (heights of 1000m compared to 150m (3)).

Figure 1: A kite tracking the trajectory of the blade tip of a conventional wind turbine (8)

By developing a method to transfer kinetic energy (contained in the periodic motion) into electrical, an alternative to the traditional turbine could be developed. Electrical power generation methods could involve a generator being mounted on a train which is dragged by a kite of fixed line length (9), fixing small turbines onto the kite (16) or by placing a generator on the ground and over a periodic loop produce net electricity by letting the kite pull the line out turning the generator and then, when most optimal retract the kite in (via a motor) (9).

The first of such systems was presented by (15) and due to ever-increasing computing performance and advances in control algorithms such systems are now a viable control problem.

As development has continued an 850kg kite has been proposed to generate 5MW (9)) an output greater than that of GE's largest offshore wind turbine (6). However current experimental systems are much smaller; Makani Power, a US company formed around this technology, has now tested a 10kW system (16).

Why is this a non-trivial control problem?

The nonlinear nature of models describing kites, the limited number of control inputs and non-deterministic nature of the wind make the task of controlling such craft substantial. Even in linearized form it is a time-varying system.

Report organisation

This report describes work carried out to control a continuous-time nonlinear kite model programmed in MATLAB and run as a Simulink model. As noted in the Literature review (page 7) this is a simple kite model which makes no attempt to model power generation. However it does make a good starting point for initial controller design work as different classes of control algorithms can be compared quickly such that the ones offering best performance can be selected for further investigation on more complicated kite models with extra design requirements (such as optimal electrical power generation).

The report is organised into the following topics:

- Presentation of the key features of the continuous nonlinear kite model (Literature review, page 7).
- Presentation of the method used to discretise the continuous kite model which had been linearized about a reference trajectory (Section 1.1, page 10).
- Presentation of a finitely supported probability distribution to act as the stochastic wind model (Section 1.2, page 12).
- Mathematical derivations are presented of various discrete control methodologies used to control linear state space models (Section 2, page 14):
    - Discrete-time-varying periodic linear feedback quadratic regulator optimal control (LQOpt).
    - Discrete-time-varying periodic H_∞ auxiliary linear feedback control (H_∞).
    - Pre-stabilized time-varying periodic deterministic linear model predictive control (DMPC).
    - Pre-stabilized time-varying stochastic model predictive control (SMPC). With soft constraints in the form of:
        - Probabilistic constraints (PSMPC).
        - Feasibility constraints (FSMPC).
- Performance evaluation of the controllers controlling both linear and nonlinear kites (Section 4.2, page 39).
- Conclusions to be drawn from the project (page 44).
- Appendices offering further clarification of points and insight into how the project was implemented in MATLAB (page 46 to 52).

Literature review

Initial kite model

A continuous-time kite model is presented in (5) and derived in detail in (4). This is a simple small two-lined kite; the physical parameters (note different notation to (5) and (4)) used for this model are described in Table 3 (this experimental kite is significantly smaller than the 850kg kite proposed to generate 5MW (9)).

Name	Symbol	Value
Line length	r	50m
Kite mass	m	1kg
Mean wind velocity	w^0	$[6,0,0]^T$ ms^{-1}
Characteristic area	A	0.5 m^2

Table 3: Kite model parameters.

The kite's position as a particle in 3D space is described by spherical polar coordinates (r, θ, ϕ). Newton's 2nd law can be used to analyse the system in continuous-time on a fixed length line ($\ddot{r} = \dot{r} = 0$). Lift and drag from the wind (force of gravity is ignored) are analysed by finding the effective wind speed acting on the kite, through various coordinate transformations; resulting in a nonlinear continuous-time differential equation (Equation 3) with a 4 element state vector ($x_C^{total}(t)$), single element input vector ($u_C^{total}(t)$) and a 3 element disturbance vector ($w_C^{total}(t)$) specifying the absolute wind velocity.

$\dot{x}_C^{total}(t) = f\left(x_C^{total}(t), u_C^{total}(t), w_c^{total}(t)\right) = f\left(\left(\theta, \dot{\theta}, \phi, \dot{\phi}\right)^T, u_C^{total}(t), w_c^{total}(t)\right)$	Equation 3

(5) use this model to generate a periodic reference input $u_C^0(t)$, for constant wind w^0 and an initial state condition to control the kite in a periodic state loop $x_C^0(t)$ of 8 seconds (however on Simulink implementation (5) found that due to numerical errors the reference input is not enough to control the kite and some control feedback is always required, indicating this is a highly unstable system).

Exit conditions for the model (given the title 'kite crashing') based on the physical assumptions are:

1. The kite must not crash into the ground ($|\theta| \leq 90°$).
2. The effective wind needs to be greater than $1x10^{-8}$ in order that the kite is blown into the same orientation as the component of effective wind tangential to the kite's surface.
3. Page 143 of (4) describes a check (Equation 4) to ensure the side of the kite facing the earth is the side with the lines extending from it. With α being a ratio of effective wind projected onto two

directions local to the kite's current orientation. A discussion on how an input constraint was produced from this is presented in Section 3.2 page 35.

$\lvert\alpha \tan(u_c^{total}(t))\rvert > 1$	Equation 4

Further (5) describes a nonlinear model predictive controller (NMPC) to control the kite, where for feedback it is assumed that all states are observable at all times and there is no delay in control input. Robustness to disturbance tests are performed by randomly 'kicking' the state at an average of once per period by a maximum of ±[5°;5°;5°s^{-1};5°s^{-1}] or ±[20°;0°;0°s^{-1};0°s^{-1}].

It is recognised that significantly more complicated kite models have been created now which include modelling power generation such as those presented by (3) and (9), where again NMPC is used for control.

Methods of controlling systems under stochastic disturbances

Additive model uncertainty with a finitely supported probability distribution is present in many processes. Methods identified to deal with additive uncertainty when constraints are present are as follows:

Most predictive controllers aiming to offer robustness to disturbance do so by tightening the constraints such that if the worst-case disturbance is present constraints are still met. This is the basis of (18), (12) and (2). While (18) only considers the worst-case disturbance, (12) extends the approach to include the probabilistic information about the disturbance to offer soft constraints as well as hard ones. (2) takes an alternative approach to generating robust constraints. A finite random sample of disturbances is taken and constraints are generated from these, instead of considering the worst case explicitly. The closed loop optimization method (13) offers a completely different method. An input is created to exactly cancel (for single variable constraints) the disturbance of previous time steps; with the remaining disturbance at the current time used to introduce a soft constraint. While (2) has the benefit of being computed on any control problem ((18), (12) and (13) are based on a linear state space model) it has the shortcoming of offering robustness to the majority of disturbances, but not all of them. (13) has the limitation that only one constrained variable can have all disturbance completely removed from it. Given the control problem presented in this report is of linear state space form subject to a probabilistic disturbance with multi variable soft constraints, the method described in (12) seemed most appropriate for further investigation.

1. Creation of the discrete kite and wind models used

1.1.Formation of the linear discrete-time-varying periodic kite model

At the start of the project the following models were available: A simulation model (Simulink) implementing the nonlinear continuous-time model of Equation 3, and samples of a continuous-time linear model, which were obtained by linearizing the nonlinear model about a pre-determined reference loop. Rather than use the nonlinear continuous-time model for the kite, the samples of the linearized model were used to construct a linear discrete-time model. Linear models were used as controller design is simpler and subsequent controllers have a lower computational burden.

1.1.1. Theory

1.1.1.1. Linear models

Linear models assume a dynamic system state is the sum of a reference value ($x_c^0(t)$ which has corresponding $u_c^0(t)$ and w^0 values) and a perturbation from that value $\left(x_C(t)\right)$ as described by Equation 5. The perturbation is then approximated (Equation 7) using 1st order derivatives (Equation 8) of the dynamic system model about an operating point $(x_c^0(t), u_c^0(t), w^0)$ described by Equation 5 and Equation 6. The approximation occurs due to the ignoring of higher-order terms. However, far from the operating point the approximation degrades, the rate of degradation with distance is nonlinear function specific.

$x_C^{Total}(t) = x_C^0(t) + x_C(t)$	Equation 5
$u_C^{Total}(t) = u_C^0(t) + u_C(t); w_C^{Total}(t) = w^0 + w_C(t)$	Equation 6
$\dot{x}_C(t) \approx A_C(t)x_C(t) + B_C(t)u_C(t) + D_C(t)w_C(t)$	Equation 7
$A_C(t) = \left.\frac{\partial f}{\partial x}\right\|_{(x_c^0(t),u_c^0(t),w_c^0(t))}; B_C(t) = \left.\frac{\partial f}{\partial u}\right\|_{(x_c^0(t),u_c^0(t),w_c^0(t))}; D_C(t) = \left.\frac{\partial f}{\partial w}\right\|_{(x_c^0(t),u_c^0(t),w_c^0(t))}$	Equation 8

1.1.1.2. Converting a continuous-time-varying model to a discrete-time-varying model

To convert a continuous-time linearized system to a discrete-time linearized system, Equation 7 is solved. Over the time interval $kT \le t \le (k+1)T$ the input $u_C(t)$ is held constant by the use of a zero order hold (ZOH). For computational ease (rather than a physical basis) the wind $w_C(t)$ is also held constant over the interval. Multiplying Equation 7 by the integrating factor $e^{-\int_{kT}^{t} A_C(t')dt'}$ yields Equation 9.

$e^{-\int_{kT}^{t} A_C(t')dt'}\dot{x}_C(t) - e^{-\int_{kT}^{t} A_C(t')dt'}A_C(t)x_C(t)$ $= e^{-\int_{kT}^{t} A_C(t')dt'}B_C(t)u_C(kT) + e^{-\int_{kT}^{t} A_C(t')dt'}D_C(t)w_C(kT)$	Equation 9

By differentiating the integrating factor Equation 10 can be stated. Letting $t' = t$ in Equation 10 enables Equation 11 to be stated and used via the chain rule in Equation 12 which is equal to the LHS of Equation 9.

$\frac{d}{dt}(e^{-\int_{kT}^{t} A(t')dt'}) = \frac{d}{dt'}(e^{-\int_{kT}^{t} A(t')dt'})\frac{dt'}{dt}$ $= \frac{d}{dt'}\left(-\int_{kT}^{t} A(t')dt'\right)\left(e^{-\int_{kT}^{t} A(t')dt'}\right)\frac{dt'}{dt} = A(t')(e^{-\int_{kT}^{t} A(t')dt'})\frac{dt'}{dt}$	Equation 10
$\frac{d}{dt}(e^{-\int_{kT}^{t} A(t')dt'}) = A(t)e^{-\int_{kT}^{t} A(t')dt'}$	Equation 11
$\frac{d}{dt}\left(e^{-\int_{kT}^{t} A_C(t')dt'}x_C(t)\right) = e^{-\int_{kT}^{t} A_C(t')dt'}\dot{x}_C(t) - e^{-\int_{kT}^{t} A_C(t')dt'}A_C(t)x_C(t)$	Equation 12

Integrating Equation 12 and equating to the RHS of the integral of Equation 9 forms Equation 13 which is rearranged to Equation 14 and Equation 15.

$\left[e^{-\int_{kT}^{t} A_C(t')dt'}x_C(t)\right]_{t=kT}^{t=(k+1)T}$ $= \int_{kT}^{(k+1)T} e^{-\int_{kT}^{t} A_C(t')dt'}B_C(t)u_C(kT)dt + \int_{kT}^{(k+1)T} e^{-\int_{kT}^{t} A_C(t')dt'}D_C(t)w_C(kT)dt$	Equation 13
$x_C((k+1)T) = e^{\int_{kT}^{(k+1)T} A_C(t')dt'}x_C(kT)$ $+ e^{\int_{kT}^{(k+1)T} A_C(t')dt'}\left(\int_{kT}^{(k+1)T} e^{-\int_{kT}^{t} A_C(t')dt'}B_C(t)u_C(kT)dt\right)$ $+ e^{\int_{kT}^{(k+1)T} A_C(t')dt'}\left(\int_{kT}^{(k+1)T} e^{-\int_{kT}^{t} A_C(t')dt'}D_C(t)w_C(kT)dt\right)$	Equation 14
$x_C((k+1)T) = e^{\int_{kT}^{T} A_C(t')dt'}x_C(kT)$ $+ \int_{kT}^{(k+1)T} e^{-\int_{t}^{(k+1)T} A_C(t')dt'}B_C(t)dt\, u_C(kT)$ $+ \int_{kT}^{(k+1)T} e^{-\int_{t}^{(k+1)T} A_C(t')dt'}D_C(t)dt\, w_C(kT)$	Equation 15

Grouping terms in Equation 15 enables the discrete-time state space model (Equation 16) with matrices (Equation 17, Equation 18 and Equation 19) to be defined.

$x_C((k+1)T) = A_k x_C(kT) + B_k u_C(kT) + D_k w_C(kT)$	Equation 16
$A_k = e^{\int_{kT}^{(k+1)T} A_C(t')dt'}$	Equation 17
$B_k = \int_{kT}^{(k+1)T} e^{\int_{t}^{(k+1)T} A_C(t')dt'}B_C(t)dt$	Equation 18
$D_k = \int_{kT}^{(k+1)T} e^{\int_{t}^{(k+1)T} A_C(t')dt'}D_C(t)dt$	Equation 19

1.1.1.3. Linear discrete-time-varying model used for controller design

As the kite flies on a periodic loop the state space matrices and reference values repeat themselves once a loop. Therefore Equation 16 is modified to the periodic loop notation introduced on page 5. Using this notation Equation 16 is rewritten as Equation 20. Those which are not periodic have a subscript which is not within PI brackets. At a given time step k the total state, input and wind are described by Equation 21.

$x_{k+1} = A_{PI(k)}x_k + B_{PI(k)}u_k + D_{PI(k)}w_k$	Equation 20
$x_k^{Total} = x_{PI(k)}^0 + x_k;\ u_k^{Total} = u_{PI(k)}^0 + u_k;\ w_k^{Total} = \mathrm{w}^0 + w_k$	Equation 21

1.1.2. Implementation

Values of $x_C^0(t)$, $u_C^0(t)$ and the time-varying matrices of Equation 8 were provided at a set of sampling instances (sampled at 2 kHz to give a good approximation to continuous-time), these were evaluated by Simulink (explanation is provided in (17)) while the kite was controlled by continuous-time linear feedback.

1.1.2.1. Calculation of linear discrete-time-varying model

Numerical approximations to the integrations in Equation 17, Equation 18 and Equation 19 were performed by Simpson's rule (14). Selection of the sampling rate (T) is discussed in Section 3.1 page 35.

1.1.2.2. Formulating state and input reference vectors

The continuous reference input $(u_C^0(t))$ and state $(x_C^0(t))$ trajectories were sampled to provide a discrete reference input (u_k^0) and state (x_k^0). It was found that best tracking of the continuous reference came from sampling the input continuous data at the midpoint between the model step times of the continuous data (as would be expected); however for the state reference best performance was obtained by sampling at the start of the interval (Equation 22). Poor results from sampling of state at the midpoint may be as a result of changing the initial state which the kite is to be tracked to, however further investigation is required. A visual record of this decision is provided on the project DVD (page 49).

$u_k^0 = u_C^0\left(kT + \frac{T}{2}\right);\ x_k^0 = x_C^0(kT)$	Equation 22

1.1.2.3. Simulink modelling environment

Two Simulink models were produced: (i) The original (5) modified so the continuous reference trajectories were replaced by discrete versions. (ii) A fully linear time-varying periodic discrete system.

1.2.Wind description

1.2.1. Simple wind description

The wind in 3 dimensional Cartesian coordinates was assumed to have a mean of $w^0 = [6,0,0]^T$ ms^{-1} with each element being independently normally distributed and stationary (covariance matrix; Equation 23) and each vector being independent of any other (Equation 24). The wind vectors are generated at the same frequency as the linear model for computational ease (i.e. D of Equation 20 could be generated at same rate as the other matrices), rather than a physical basis.

$\Omega_w = \begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix} = 0.1\mathrm{I} = \mathbb{E}\left(w_{k+i}\mathrm{w}_{k+i}^{\mathrm{T}}\right) \forall\, i \in \mathbb{N}^0$	Equation 23
$\mathbb{E}\left((w_i)(w_j)^T\right) = \mathbf{0} \forall\, i \neq j$	Equation 24

1.2.2. Truncating the wind distribution

The $\chi^2(n)$ distribution describes the sum of the squares of n zero mean independent normal random variables each with a variance of 1 (Equation 25). The $\chi^2(n)$ distribution therefore determines the probability distribution of the sum of the squares of elements of vectors of normally distributed elements.

Standard tables (such as (11)) list, for a range of numbers of degrees of freedom, n, the maximum values (r^2 of Equation 26) that this sum of squares can take with a given probability: $\overline{p}$. With modification the $\chi^2(3)$ distribution is appropriate for the simple kite system as it is built using normal distributions which have simple mathematical properties (discussed further in Section 2.4.1.1). The wind model however was truncated so that the most extreme events are bounded to a known level (and therefore can be designed for); 1% of the probability density function of $\chi^2(3)$ was discarded, corresponding to the most extreme wind events. By a pre and post matrix multiplication trick (Equation 27) the wind random vector can be transferred to a new vector z (Equation 28) with identity covariance so that $z^T z = \chi^2(n)$.

$z^T z = \left(\sum_{\mathrm{i}=1}^{\mathrm{n}} z_i^2\right) \sim \chi^2(n)\ if\ \left(\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathrm{I})\right)$	Equation 25
$pr(z^T z \leq r^2) = \overline{p}$	Equation 26
$\Omega_w^{-\frac{1}{2}}\Omega_w\Omega_w^{-\frac{1}{2}} = \mathrm{I} = \Omega_w^{-\frac{1}{2}}\mathbb{E}(ww^T)\Omega_w^{-\frac{1}{2}} = \mathbb{E}(zz^T)$	Equation 27
$z = \Omega_w^{-\frac{1}{2}} w$	Equation 28
$pr(w^T\Omega_w^{-1}w \leq r^2) = \overline{p}$	Equation 29

Substituting Equation 28 into Equation 26 results in Equation 29. If 1% of the mass is to be removed, a value of $\overline{p} = 0.99$ is looked up, giving the corresponding r^2 value of 11.34 (11). Therefore when a new random wind vector is generated the inequality ($w^T\Omega_w^{-1}w \leq r^2$) must be checked that it is satisfied. If it is not then the wind vector must be discarded and a new one generated. This generate, check, discard or keep cycle removes 1% of the probability distribution from the extremities of the wind and truncates the wind vector down to a sphere of radius $\sqrt{11.34}$. This truncated distribution is considered as having a greater bearing on reality than truncating each wind component individually and forcing the wind vector to lie in a polytope.

2. Controller design

In this section the mathematical derivations of various linear control strategies are presented. These were all implemented in MATLAB and are included in the attached DVD (page 49). A description of the measures taken to verify the implementation in code of the maths described in this section is presented in Appendix A: Algorithm verification/Checks performed (page 46).

2.1. Discrete-time-varying periodic linear feedback quadratic regulator optimal control (LQOpt)

2.1.1. Mathematical theory

Optimal control is a method whereby the control input is selected to minimize a predetermined cost function, $J(x,u)$. The discrete linear model in Equation 20 is of a form where the linear quadratic regulator (LQR) cost function (Equation 30, where Q and R are PSD (positive semi definite) symmetric matrix weights which punish deviation of the state and control from zero respectively) can be minimized analytically (10). The minimum is denoted J^*(Equation 31).

$J = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_k^T R u_k)$	Equation 30
$J^* = \min(J)$	Equation 31
$J_N^* = x_N^T P_N x_N$	Equation 32

Rather than evaluate an infinity of terms explicitly an alternative method is derived by first speculating the solution form and then showing it holds by induction (10). First assume that the optimal cost from time N to infinity (J_N^* Equation 32) is a function of state at time step N (x_N) and a matrix (terminal cost matrix (P_N)).

The cost from $N-1$ to infinity (J_{N-1} Equation 33) is then the cost from N to infinity (J_N^* Equation 32) plus terms for the cost associated with the time $N-1$ (which if u_{N-1} is equal to the optimal value, u_{N-1}^*, results in the optimal cost J_{N-1}^*). Figure 2 displays graphically the horizons over which these costs are evaluated for additional clarity.

$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_N^T R u_N + x_N^T P_N x_N$	Equation 33

Substituting for x_N using the linear model with no disturbances (Equation 34) into Equation 33 results in a cost (J_{N-1} Equation 35) which is only a function of state and input at time step $N-1$, which on rearrangement making the assumption P_N is symmetric results in Equation 36.

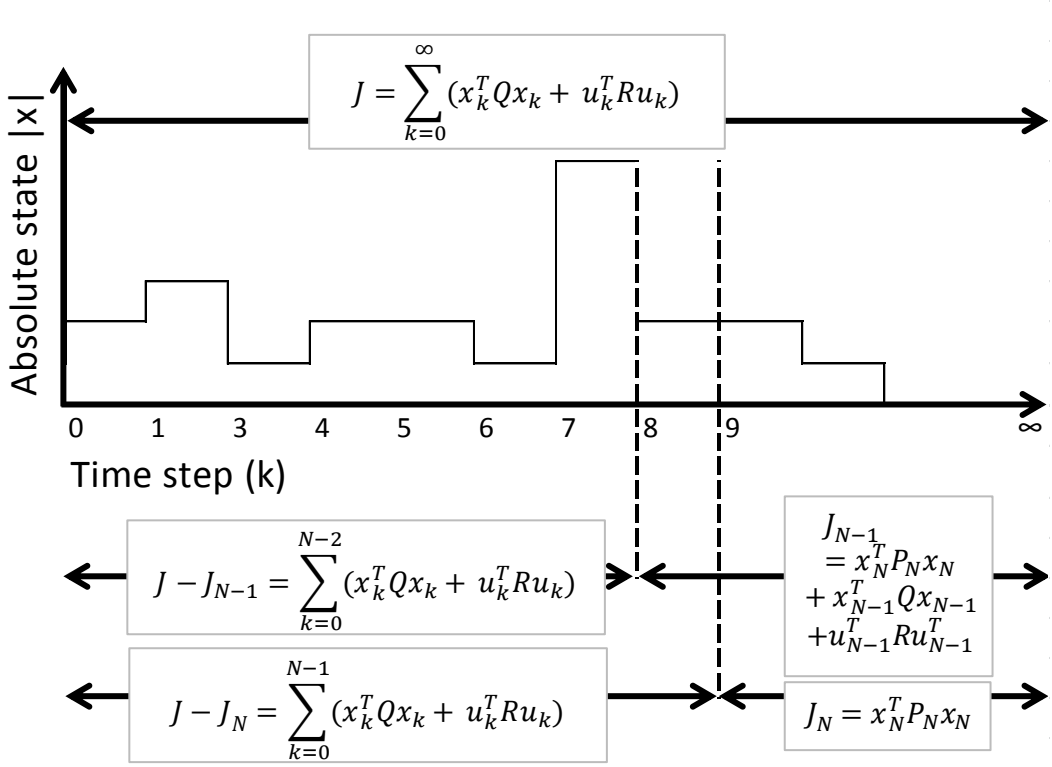

Figure 2: For an arbitrary 1 dimension state trajectory, the cost functions (Equation 30, Equation 32 and Equation 33) are shown for $N = 9$. Note: Due to the lack of * notation, these are not necessarily optimal costs.

$x_N = A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}$	Equation 34
$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)^T P_N (A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1})$	Equation 35
$J_{N-1} = x_{N-1}^T \left(Q + A_{PI(N-1)}^T P_N A_{PI(N-1)}\right)x_{N-1} + u_{N-1}^T \left(R + B_{PI(N-1)}^T P_N B_{PI(N-1)}\right)u_{N-1} + 2u_{N-1}^T B_{PI(N-1)}^T P_N A_{PI(N-1)} x_{N-1}$	Equation 36

Differentiating Equation 36 with respect to u_{N-1} and setting the result to zero to find the stationary point results in Equation 37. On rearrangement, the optimal input (u_{N-1}^*) and gain matrix (K_{N-1}) to move from time step $N-1$ to N is then found (Equation 38). Substituting this optimal input (u_{N-1}^*) into Equation 36 results in the optimal cost (J_{N-1}^* Equation 39) which is only dependent on the entry state and a new terminal cost matrix (P_{N-1}). However this process only works if the matrix in Equation 38, which requires inverting, is non-singular (i.e. it must be positive definite)

$\left.\frac{\partial J_{N-1}}{\partial u_{N-1}}\right\|_{u_{N-1}=u_{N-1}^*} = 2\left(R + B_{PI(N-1)}^T P_N B_{PI(N-1)}\right)u_{N-1}^* + 2B_{PI(N-1)}^T P_N A_{PI(N-1)} x_{N-1} = 0$	Equation 37

$u_{N-1}^{*} = -\left(R + B_{PI(N-1)}^{T} P_N B_{PI(N-1)}\right)^{-1} (B_{PI(N-1)}^{T} P_N A_{PI(N-1)}) x_{N-1} = K_{N-1} x_{N-1}$ $= \arg\min_{u_{N-1}} J_{N-1}$	Equation 38
$J_{N-1}^{*} = x_{N-1}^{T}[Q + K_{N-1}^{T} R K_{N-1}] x_{N-1}$ $+ x_{N-1}^{T}[\left(A_{PI(N-1)} + B_{PI(N-1)} K_{N-1}\right)^{T} P_N (A_{PI(N-1)} + B_{PI(N-1)} K_{N-1})] x_{N-1}$ $= x_{N-1}^{T} P_{N-1} x_{N-1}$	Equation 39

The same process that generated Equation 33 is applied to Equation 39 to generate Equation 40. This could clearly have the same analysis applied to it to generate another optimal gain matrix and a terminal cost matrix which includes the stage cost corresponding to time $N - 2$.

$J_{N-2} = x_{N-2}^{T} Q x_{N-2} + u_{N-2}^{T} R u_{N-2} + x_{N-1}^{T} P_{N-1} x_{N-1}$	Equation 40

Assuming that the cost to infinity is bounded (so values will converge) a recursive formula can be formed to generate optimal gains (K_{PI}) and terminal cost matrices (P_{PI}) which make use of the periodic index notation. This implementation is shown in Figure 3.

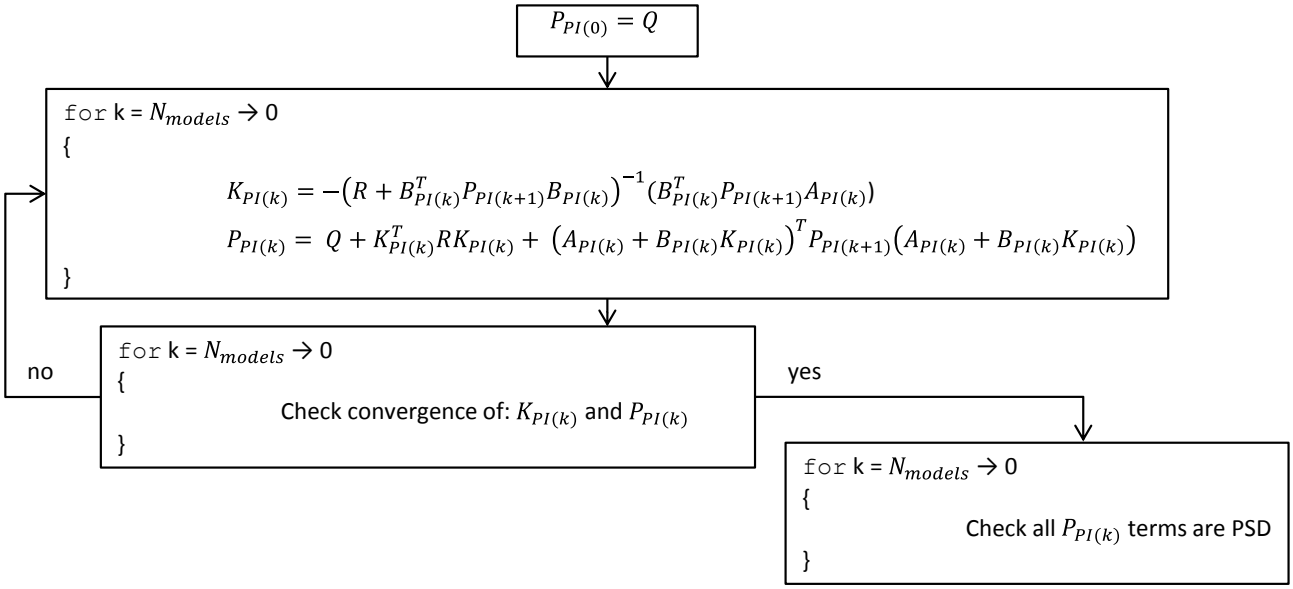

Figure 3 LQOpt control terminal cost matrices (P_{PI}) and gain matrices (K_{PI}) generation algorithm.

Once this iteration has converged, the terminal cost matrices P_{PI} and the gain matrices K_{PI} are solutions of the time-varying periodic Algebraic Riccati Equation (10).

2.1.2. Limitations and inherent assumptions

The optimal controller feedback will minimize the cost function. However it takes no consideration for constraints on input or state. Therefore careful choice of both Q and R is required to give performance on

implementation which is satisfactory and performance may be improved by incorporating time-varying periodic Q and R. Further, nowhere in the design has any knowledge of the disturbance been included.

2.1.3. Implementation considerations

Once the optimal gain matrices are calculated offline the online input calculation is a single matrix-vector multiplication which can be performed at very little computational cost.

2.2.Discrete-time-varying periodic H∞ auxiliary linear feedback control (H∞)

An alternative linear feedback controller is derived which attempts to optimize the cost of a cost function which has been maximized with respect to disturbance, (19) explains this method but for models with even more uncertainty terms than considered here.

2.2.1. Mathematical theory

It is possible to modify the cost function (Equation 30) to include a term for the disturbance as shown in Equation 41, with $\gamma_{H_\infty PI(k)}$ representing a time-varying periodic weighting on that disturbance. Here satisfaction of Equation 42 ensures the cost has a finite sum.

$J = \sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k - w_k^T \left(\gamma_{H_\infty PI(k)}^2 I\right) w_k\right)$	Equation 41
$\sum_{k=0}^{\infty} \left(x_k^T Q x_k + u_k^T R u_k\right) \leq \sum_{k=0}^{\infty} w_k^T \left(\gamma_{H_\infty PI(k)}^2 I\right) w_k$	Equation 42

Therefore taking Equation 33 and modifying it to include the extra cost term results in Equation 43, which, if x_N is substituted, results in Equation 44, where the disturbance term is included. In this case the optimal control is computed for the disturbance which produces the maximum cost. Under the assumption that P_N is symmetric Equation 44 can be rearranged to Equation 45.

$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} - w_{N-1}^T \left(\gamma_{H_\infty PI(N-1)}^2 I\right) w_{N-1} + x_N^T P_N x_N$	Equation 43
$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} - w_{N-1}^T \left(\gamma_{H_\infty PI(N-1)}^2 I\right) w_{N-1} + \left(A_{PI(N-1)} x_{N-1} + B_{PI(N-1)} u_{N-1} + D_{PI(N-1)} w_{N-1}\right)^T P_N (A_{PI(N-1)} x_{N-1} + B_{PI(N-1)} u_{N-1} + D_{PI(N-1)} w_{N-1})$	Equation 44
$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + \left(A_{PI(N-1)} x_{N-1} + B_{PI(N-1)}\right)^T P_N \left(A_{PI(N-1)} x_{N-1} + B_{PI(N-1)} u_{N-1}\right) + w_{N-1}^T \left(D_{PI(N-1)}^T P_N D_{PI(N-1)} - \gamma_{H_\infty PI(N-1)}^2 I\right) w_{N-1} + 2 w_N^T D_{PI(N-1)}^T P_N \left(A_{PI(N-1)} x_{N-1} + B_{PI(N-1)} u_{N-1}\right)$	Equation 45

By completing the square, Equation 45 can be rearranged to Equation 46. Here $L_{N-1}$and S_{N-1} are defined in Equation 47 and Equation 48 respectively. There is a unique global maximum if Equation 49 is satisfied, i.e. the maximum eigenvalue of $D_{PI(N-1)}^T P_N D_{PI(N-1)}$ must be less than $\gamma_{H_\infty PI(N-1)}^2$, ensuring the Hessian of the cost function is negative definite with respect to w_{N-1}.

$J_{N-1} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}\right)^T P_N \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right) + (w_{N-1} + S_{N-1})^T L_{N-1}(w_{N-1} + S_{N-1}) + \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)^T P_N^T D_{PI(N-1)} S_{N-1}$	Equation 46
$L_{N-1} = D_{PI(N-1)}^T P_N D_{PI(N-1)} - \gamma_{H_\infty PI(N-1)}^2 I$	Equation 47
$S_{N-1} = (L_{N-1})^{-1} D_{PI(N-1)}^T P_N \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)$	Equation 48
$\gamma_{H_\infty PI(N-1)}^2 I > D_{PI(N-1)}^T P_N D_{PI(N-1)}$	Equation 49

Equation 46 can then be maximized by differentiating with respect to w_{N-1}.The result of differentiating Equation 46 results in Equation 50. Wind disturbance (w_{N-1}^*) at max cost is displayed in Equation 51.

$0 = \frac{\partial J_{N-1}}{\partial w_{N-1}}\Big\|_{w_{N-1} = w_{N-1}^*} = L_{N-1} w_{N-1}^* + D_{PI(N-1)}^T P_N \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)$	Equation 50
$w_{N-1}^* = -(L_{N-1})^{-1} D_{PI(N-1)}^T P_N \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)$	Equation 51

Now that maximum cost has been defined only in terms of input (u_{N-1}) and state (x_{N-1}), the input required (u'_{N-1}) to minimize this cost (Equation 52) can then be determined in a similar way as Section 2.1. Therefore, analogously to Equation 35 grouping the quadratic terms of $A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}$ in Equation 53 and labelling them P'_N, (to be used in the place of the terminal weighting matrix P_N in Section 2.1 when calculating controller gains iteratively) yields Equation 54.

$u'_{N-1} = \underset{u_{N-1}}{\arg\min}\left(\underset{w_{N-1}}{\max} J_{N-1}\right) = \underset{u_{N-1}}{\arg\min}(J_{N-1}(w_{N-1}^*(x_{N-1}, u_{N-1}), x_{N-1}, u_{N-1}))$	Equation 52
$J_{N-1}(w_{N-1}^*(x_{N-1}, u_{N-1}), x_{N-1}, u_{N-1}) = J_{N-1}\|_{w_{N-1} = w_{N-1}^*} = x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}\right)^T P_N \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right) + (w_{N-1}^* + S_{N-1})^T T_{N-1}(w_{N-1}^* + S_{N-1}) + \left(A_{PI(N-1)}x_{N-1} + B_{PI(N-1)}u_{N-1}\right)^T P_N^T D_{PI(N-1)} S_{N-1}$	Equation 53
$P'_N = P_N + P_N D_{PI(N-1)}\left(D_{PI(N-1)}^T P_N D_{PI(N-1)} - \gamma_{H_\infty PI(N-1)}^2 I\right)^{-1} D_{PI(N-1)}^T P_N$	Equation 54

Figure 4 presents a modified Figure 3 as a method of calculating linear feedback gains. Note that this procedure determines a sequence of $\gamma_{H_\infty PI}$ values satisfying Equation 49, and, due to the complex recursive nature of the algorithm the starting values of $\gamma_{H_\infty PI}$ ($\gamma_{H_\infty Inital}$) will likely have an impact on the

final feedback gain matrices (K_{PI}). The algorithm (Figure 4) was used for its simplicity; other methods (perhaps calculating for many initial values, $\gamma_{H_\infty Inital}$) may produce better controllers.

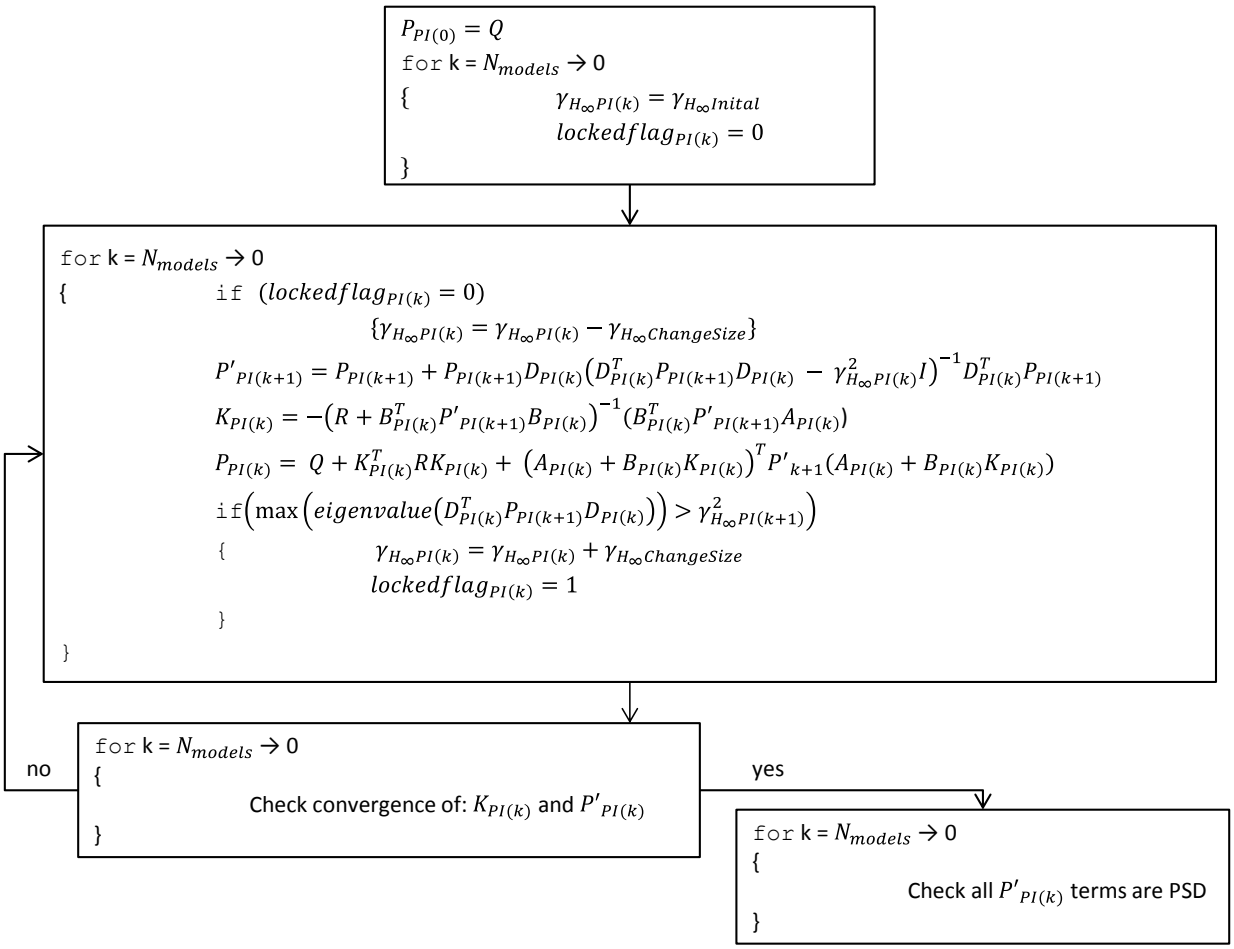

Figure 4: Algorithm to generate H∞ auxiliary feedback control terminal cost (P'_{PI}) matrices and gain (K_{PI}) matrices.

2.2.2. Limitations and inherent assumptions

The H∞ auxiliary feedback controller analysis attempted to minimize the cost function which has been maximized over disturbance, but no information about the wind disturbance was included.

2.2.3. Implementation considerations

Once the H∞ gain matrices are calculated offline implementation is the same as LQOpt.

2.3. Pre-stabilized time-varying periodic deterministic linear model predictive control (DMPC)

Linear model predictive control (MPC) explicitly addresses inability of linear feedback control schemes to control linear systems to respect constraints. However in doing so the amount of computation which is required both online and offline increases significantly.

2.3.1. Mathematical theory

2.3.1.1. Pre-stabilized dual mode

MPC is based around predicting how a system will respond to an input sequence and then optimizing that input sequence subject to constraints. The (Mode 1) input generated by the controller (Equation 55) is taken to be the summation of a linear feedback control law with an additional input term (c_k which is a free variable in an online MPC optimization). The linear feedback controller provides stabilization (needed for recursive feasibility) of the linear system while the c_k modifies the control input to prevent constraints from being violated. The optimization can only provide a solution to problems of a finite size, therefore after N_{M1} prediction steps the input is purely linear feedback ($c_k = 0$) known as Mode 2.

$u_k = K_{PI(k)}x_k + c_k$	Equation 55

2.3.1.2. Linear predictions

The state (ignoring stochastic disturbance; $w_k = \mathbf{0}$) at the next time step (x_{k+1} Equation 57, $\Phi_{PI(k)}$ defined in Equation 58) can be predicted by feeding the Mode 1 input (Equation 55) into the model (Equation 56). Repeating this process N_{M1} times and grouping the results, it is possible to form a vector of N_{M1} state predictions (Equation 59, $M_{PI(k)}$, $C_{PI(k)}$ and $\boldsymbol{c}$ defined by Equation 62, Equation 64 and Equation 61 respectively) and input predictions (Equation 60, $M'_{PI(k)}$, $C'_{PI(k)}$ and $\boldsymbol{c}$ defined by Equation 63, Equation 65 and Equation 61 respectively). These predictions are then only a function of current state (x_k) and a vector of MPC inputs $\boldsymbol{c}$ (Equation 61, which are to be determined via optimization).

$x_{k+1} = A_{PI(k)}x_k + B_{PI(k)}u_k$	Equation 56
$x_{k+1} = \Phi_{PI(k)}x_k + B_{PI(k)}c_k$	Equation 57
$\Phi_{PI(k)} = A_{PI(k)} + B_{PI(k)}K_{PI(k)}$	Equation 58
$X = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+N_{M1}-1} \\ x_{k+N_{M1}} \end{bmatrix} = M_{PI(k)}x_k + C_{PI(k)}\boldsymbol{c}$	Equation 59
$U = \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N_{M1}-2} \\ u_{k+N_{M1}-1} \end{bmatrix} = M'_{PI(k)}x_k + C'_{PI(k)}\boldsymbol{c}$	Equation 60
$\boldsymbol{c} = \begin{bmatrix} c_k \\ c_{k+1} \\ c_{k+2} \\ \vdots \\ c_{k+N_{M1}-2} \\ c_{k+N_{M1}-1} \end{bmatrix}$	Equation 61

$$M_{PI(k)} = \begin{bmatrix} \Phi_{PI(k)} \\ \Phi_{PI(k+1)}\Phi_{PI(k)} \\ \Phi_{PI(k+2)}\Phi_{PI(k+1)}\Phi_{PI(k)} \\ \vdots \\ \prod_{i=2}^{N_{M1}} \Phi_{PI(k+N_{M1}-i)} \\ \prod_{i=1}^{N_{M1}} \Phi_{PI(k+N_{M1}-i)} \end{bmatrix}$$	Equation 62
$$M'_{PI(k)} = \begin{bmatrix} K_{PI(k)} \\ K_{PI(k+1)}\Phi_{PI(k)} \\ \mathrm{K}_{PI(k+2)}\Phi_{PI(k+1)}\Phi_{PI(k)} \\ \vdots \\ K_{PI(k+N_{M1}-2)}\left(\prod_{i=3}^{N_{M1}} \Phi_{PI(k+N_{M1}-i)}\right) \\ K_{PI(k+N_{M1}-1)}\left(\prod_{i=2}^{N_{M1}} \Phi_{PI(k+N_{M1}-i)}\right) \end{bmatrix}$$	Equation 63

$$C_{PI(k)} = \begin{bmatrix} B_{PI(k)} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \Phi_{PI(k+1)}B_{PI(k)} & B_{PI(k+1)} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \Phi_{PI(k+2)}\Phi_{PI(k+1)}B_{PI(k)} & \Phi_{PI(k+2)}B_{PI(k+1)} & B_{PI(k+2)} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \left(\prod_{i=2}^{N_{M1}-1} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k)} & \left(\prod_{i=2}^{N_{M1}-2} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+1)} & \left(\prod_{i=2}^{N_{M1}-3} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+2)} & \dots & B_{PI(k+N_{M1}-2)} & \mathbf{0} \\ \left(\prod_{i=1}^{N_{M1}-1} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k)} & \left(\prod_{i=1}^{N_{M1}-2} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+1)} & \left(\prod_{i=1}^{N_{M1}-3} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+2)} & \dots & \left(\prod_{i=1}^{N_{M1}-j} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+N_{M1}-2)} & B_{PI(k+N_{M1}-1)} \end{bmatrix}$$

Equation 64

$$C'_{PI(k)} = \begin{bmatrix} \mathrm{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathrm{K}_{PI(k+1)}B_{PI(k)} & \mathrm{I} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathrm{K}_{PI(k+2)}\Phi_{PI(k+1)}B_{PI(k)} & \mathrm{K}_{PI(k+2)}B_{PI(k+1)} & \mathrm{I} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ K_{PI(k+N_{M1}-2)}\left(\prod_{i=3}^{N_{M1}-1} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k)} & K_{PI(k+N_{M1}-2)}\left(\prod_{i=3}^{N_{M1}-2} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+1)} & K_{PI(k+N_{M1}-2)}\left(\prod_{i=3}^{N_{M1}-3} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+2)} & \dots & \mathrm{I} & \mathbf{0} \\ K_{PI(k+N_{M1}-1)}\left(\prod_{i=2}^{N_{M1}-1} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k)} & K_{PI(k+N_{M1}-1)}\left(\prod_{i=2}^{N_{M1}-2} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+1)} & K_{PI(k+N_{M1}-1)}\left(\prod_{i=2}^{N_{M1}-3} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+2)} & \dots & K_{PI(k+N_{M1}-1)}\left(\prod_{i=2}^{N_{M1}-j} \Phi_{PI(k+N_{M1}-i)}\right)B_{PI(k+N_{M1}-2)} & \mathrm{I} \end{bmatrix}$$

Equation 65

2.3.1.3. Cost for optimization to generate c

By predicting how the system would respond to input it is possible to rearrange the cost function (Equation 30) to a form where predicted state and input can be used to predict the cost; Equation 66 (block diagonal matrices $\overline{Q}$ and $\overline{R}$ defined in Equation 67 and Equation 68 respectfully, having dimension N_{M1} blocks by N_{M1} blocks).

$J = X^T \overline{Q}_{PI(k+N_{M1})} X + U^T \overline{R} U + x_k^T Q x_k$	Equation 66
$\overline{Q}_{PI(k+N_{M1})} = \begin{bmatrix} Q & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & Q & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & P_{PI(k+N_{M1})} \end{bmatrix}$	Equation 67
$\overline{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & R \end{bmatrix}$	Equation 68

Substituting for state (Equation 59) and input (Equation 60) predictions into the cost function (Equation 66) results in a new cost function (Equation 69) which is quadratic in the MPC input sequence vector ($\boldsymbol{c}$). The cost function can then be minimized analytically by finding its stationary point in much the same way as was performed on the optimal control case (Section 2.1). However if no constraints are included, this would render a result equal to the LQOpt case with $\boldsymbol{c} = \mathbf{0}$. Instead optimization is performed subject to constraints.

$J = \left(M_{PI(k)} x_k + C_{PI(k)} \boldsymbol{c}\right)^T \overline{Q}_{PI(k+N)} \left(M_{PI(k)} x_k + C_{PI(k)} \boldsymbol{c}\right) + \left(M'_{PI(k)} x_k + C'_{PI(k)} \boldsymbol{c}\right)^T \overline{R} \left(M'_{PI(k)} x_k + C'_{PI(k)} \boldsymbol{c}\right) + x_k^T Q x_k$	Equation 69

Minimizing quadratic cost functions subject to equality and inequality constraints is a standard problem and therefore the MATLAB optimization toolbox provides a QP solver (quadprog) to do such calculation efficiently, providing a solution to Equation 70. Equation 69 can be rearranged to the form (Equation 71) used by MATLAB where $H_{PI(k)}$, $f_{PI(k)}$ and $h_{PI(k)}$ are defined by Equation 72, Equation 73 and Equation 74 respectively). These are dependent on the current position on the loop that the kite is at $\left(PI(k)\right)$ and the current state (x_k). Note that $\overline{Q}_{PI(k+N_{M1})}$ and $\overline{R}$ are both symmetric (under the assumption $P_{PI(k+N_{M1})}$ is symmetric) enabling cross terms of the quadratic to be grouped together $\left(2\boldsymbol{c}^T\left(f_{PI(k)}\right)\right)$.

$\boldsymbol{c}^* = arg\min_{\boldsymbol{c}}(J)\ s.t. constraints$	Equation 70
$J = \boldsymbol{c}^T H_{PI(k)} \boldsymbol{c} + 2\boldsymbol{c}^T \left(f_{PI(k)}\right) + x_k^T (h_{PI(k)}) x_k$	Equation 71
$H_{PI(k)} = C_{PI(k)}^T \overline{Q}_{PI(k+N_{M1})} C_{PI(k)} + C'^T_{PI(k)} \overline{R} C'_{PI(k)}$	Equation 72
$f_{PI(k)} = C_{PI(k)}^T \overline{Q}_{PI(k+N_{M1})} M_{PI(k)} x_k + C'^T_{PI(k)} \overline{R} M'_{PI(k)} x_k$	Equation 73
$h_{PI(k)} = x_k^T (M_{PI(k)}^T \overline{Q}_{PI(k+N_{M1})} M_{PI(k)} + M'^T_{PI(k)} \overline{R} M'_{PI(k)}) x_k$	Equation 74

2.3.1.4. Receding horizon implementation

Rather than use the full $\boldsymbol{c}$ vector, only the first value c_k is applied to the plant at time k. This is due to another optimization (Equation 70) being performed at the next time step making use of the actual state at time step $k+1$ rather than a prediction. Further the receding horizon allows the prediction horizon to step forward and make a prediction using the next step of the model at a new further step in the future at the end of the Mode 1 horizon.

2.3.1.5. Constraints

Two types of constraint were implemented: (i) Limits on the max deviation from the reference, in which the components of the state were constrained (Equation 75) not to deviate beyond a value (Δx) from the reference so as to remain tight to the predefined trajectory; (ii) limits on maximum/minimum value, in which both the input (Equation 76) and state (Equation 77) were limited to a maximum ($\overline{u}$ and $\overline{x}$) and minimum ($\underline{u}$ and $\underline{x}$) values at all points in the loop to prevent actuator saturation and kite crashing. Equation 78 defines the various constraint vectors which are of appropriate dimension as to act on the entire prediction horizon. Equation 79 defines vectors for the input and state reference of correct dimension.

$-\Delta X \leq X \leq \Delta X$	Equation 75
$\underline{U} \leq U^0 + U \leq \overline{U}$	Equation 76
$\underline{X} \leq X^0 + X \leq \overline{X}$	Equation 77
$\Delta X = \begin{bmatrix} \Delta x \\ \vdots \\ \Delta x \end{bmatrix}; \underline{U} = \begin{bmatrix} \underline{u} \\ \vdots \\ \underline{u} \end{bmatrix}; \overline{U} = \begin{bmatrix} \overline{u} \\ \vdots \\ \overline{u} \end{bmatrix}; \underline{X} = \begin{bmatrix} \underline{x} \\ \vdots \\ \underline{x} \end{bmatrix}; \overline{X} = \begin{bmatrix} \overline{x} \\ \vdots \\ \overline{x} \end{bmatrix};$	Equation 78
$U_{M1}^0 = \begin{bmatrix} u_{PI(k)}^0 \\ \vdots \\ u_{PI(N_{M1}+k)}^0 \end{bmatrix};\ X_{M1}^0 = \begin{bmatrix} x_{PI(k+1)}^0 \\ \vdots \\ x_{PI(k+N_{M1}+1)}^0 \end{bmatrix};$	Equation 79

The constraints can be included in the MATLAB optimization by splitting the individual combined inequalities of Equation 75, Equation 76 and Equation 77 and then substituting for the predicted state (Equation 59) and input (Equation 60). This process forms the constraints: Equation 80, state lower

deviation from reference; Equation 81, state upper deviation from reference; Equation 82, input global minimum value; Equation 83, input global maximum value; Equation 84, state global minimum value and Equation 85 state global maximum value. Equation 80 to Equation 85 may be stacked into a matrix and vector inequality of the form $A_{PI(k)}^{constraint}\boldsymbol{c} \leq b_{PI(k)}^{constraint}$. Note that $b_{PI(k)}^{constraint}$ is a function of current state (x_k).

$-\Delta X \leq X \Rightarrow -C_{PI(k)}\underline{c} \leq M_{PI(k)}x_k + \Delta X$	Equation 80
$X \leq \Delta X \Rightarrow C_{PI(k)}\underline{c} \leq -M_{PI(k)}x_k + \Delta X$	Equation 81
$\underline{U} \leq U^0 + U \Rightarrow -C'_{PI(k)}\underline{c} \leq M'_{PI(k)}x_k + U_{M1}^0 - \underline{U}$	Equation 82
$U^0 + U \leq \overline{U} \Rightarrow C'_{PI(k)}\underline{c} \leq -M'_{PI(k)}x_k - U_{M1}^0 + \overline{U}$	Equation 83
$\underline{X} \leq X^0 + X \Rightarrow -C_{PI(k)}\underline{c} \leq M_{PI(k)}x_k + X_{M1}^0 - \underline{X}$	Equation 84
$X^0 + X \leq \overline{X} \Rightarrow C_{PI(k)}\underline{c} \leq -M_{PI(k)}x_k - X_{M1}^0 + \overline{X}$	Equation 85

A feasible solution to the optimization given an initial state and these constraints implies that for no wind disturbance the kite system will not break any constraints over the N_{M1} predicted time steps accounted for in Equation 80 to Equation 85.

2.3.1.6. Recursive feasibility

Further constraints are added to deal with kite control after the end of Mode 1 i.e. Mode 2, where $c_k = 0$ (input and state predictions are defined by Equation 86 and Equation 87 respectively). To ensure that the next minimization is feasible, the QP solver must be able to return a feasible solution corresponding to the term at $N_{M1} + 1$ in the current optimization. An initial approach could be to extend the constraints and predictions (using linear feedback) from the end of the Mode 1 predictions over the infinite Mode 2 horizon, and as such if the QP solver found a feasible solution (given an initial state) the kite could be deemed (with no wind disturbance) to never break constraints. Clearly constraints cannot be enforced explicitly over an infinite horizon. Due to pre-stabilization it is asserted that once the kite (in predictions) travels around a full loop under linear state feedback control and does not break any constraints at any time step in the loop with no disturbances, the kite will remain within these constraints. This can be proven using the approach presented in (7). Thus Mode 2 constraints are calculated offline for a sufficient number of predictions to reach this for the worst-case state on entry to Mode 2, the kite on meeting these constraints (with no further disturbance) will remain within constraints over the rest of its trajectory.

$$U_{M2} = \begin{bmatrix} K_{PI(k+N_{M1})} \\ \vdots \\ K_{PI(k+N_{M1}+N_{M2})}\left(\prod_{\mathrm{i}=2}^{N_{M2}} \Phi_{PI(k+N_{M1}+N_{M2}-i)}\right) \\ K_{PI(k+N_{M1}+N_{M2})}\left(\prod_{\mathrm{i}=1}^{N_{M2}} \Phi_{PI(k+N_{M1}+N_{M2}-i)}\right) \end{bmatrix} x_{k+N_{M1}}$$	Equation 86
$$X_{M2} = \begin{bmatrix} \mathrm{I} \\ \vdots \\ \left(\prod_{\mathrm{i}=2}^{N_{M2}} \Phi_{PI(k+N_{M1}+N_{M2}-i)}\right) \\ \left(\prod_{\mathrm{i}=1}^{N_{M2}} \Phi_{PI(k+N_{M1}+N_{M2}-i)}\right) \end{bmatrix} x_{k+N_{M1}}$$	Equation 87

Generation of the finite Mode 2 constraints is handled via the algorithm displayed in Figure 5, where maximum and minimum of the functions are linear programming problems subject to constraints. With U_{M2}^0 and X_{M2}^0 defined analogously to Equation 79 but both start offset by N_{M1} and are of block height N_{M2}.

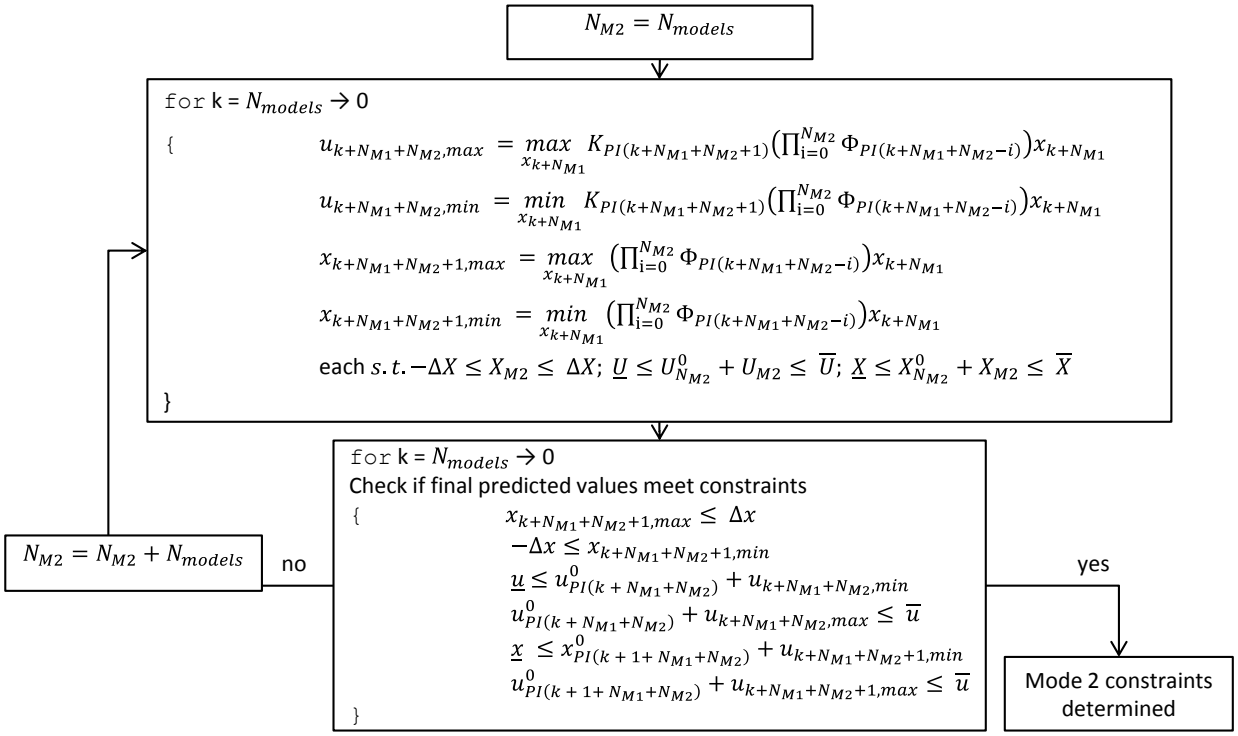

Figure 5: Mode 2 finite constraint generating algorithm.

Subsequently the last line of Equation 59 (function of $\boldsymbol{c}$) replaces the $x_{k+N_{M1}}$ term in the Mode 2 prediction equations (Equation 86 and Equation 87) forming a function of x_k for substitution into the Mode 2 constraints which are then packaged similarly to that described by Equation 80 to Equation 85.

2.3.2. Limitations and inherent assumptions

The dual mode MPC controller is both nonlinear and no longer optimal (with regard to optimization without constraints). Further, no knowledge of the disturbance has been utilised.

2.3.3. Implementation considerations

Incorporating the Mode 1 and Mode 2 constraints in Equation 70 forms Equation 88 which must be optimized online (via MATLAB's QP solver; quadprog). Note that the constraints (ΔX, $\overline{U}$ etc) are of correct block height. To increase the speed of calculation, as much computation as possible is done offline.

$\boldsymbol{c}^* = arg\min_{\boldsymbol{c}}(J)\ s.t.:$ $-\Delta X \le X_{M1} \le \Delta X; \underline{U} \le U^0_{M1} + U_{M1} \le \overline{U}; \underline{X} \le X^0_{M1} + X_{M1} \le \overline{X}$ $-\Delta X \le X_{M2} \le \Delta X; \underline{U} \le U^0_{M2} + U_{M2} \le \overline{U}; \underline{X} \le X^0_{M2} + X_{M2} \le \overline{X}$	Equation 88

2.4.Pre-stabilized time-varying stochastic model predictive control (SMPC)

As detailed in the Section 1.2 (page 12), wind is generated by a probability distribution and as the linear state space model Equation 89 shows it effects the state immediately to cause a disturbance.

$x_{k+1} = A_{PI(k)}x_k + B_{PI(k)}u_k + D_{PI(k)}w_k$	Equation 89

If the wind $\boldsymbol{w}$ was deterministic, it would be possible to include the disturbance in the optimization exactly by adding additional terms to the prediction equations. To deal with the stochastic disturbance a probabilistic approach is needed. Stochastic model predictive control (SMPC) in its various forms offers specified levels of performance of a control system given a stochastic disturbance. For this report these are soft constraints (which allow constraints to be violated up to a maximum frequency (12)) and hard constraints (even under stochastic disturbance the constraint is prevented from ever being broken). Two forms of SMPC are presented (from (12)): (i) Probabilistic stochastic model predictive control (PSMPC), where soft constraints are allowed to be broken with a given probability; (ii) feasible stochastic model predictive control (FSMPC), an extension of PSMPC where soft constraints are allowed to be broken to a given probability but the optimization remains recursively feasible at all times (requiring explicit knowledge of a finitely supported stochastic disturbance). Constraints described in Section 2.3.1.5 were modified to: hard constraints (Equation 76 and Equation 77) and a soft constraint (Equation 75).

2.4.1. Mathematical theory

By using a recursive state model with a disturbance (Equation 89) and same Mode 1 input (Equation 55) the prediction equations become Equation 90 and Equation 91 with $\boldsymbol{w}$, $E_{PI(k)}$ and $E'_{PI(k)}$ defined by Equation 92, Equation 93 and Equation 94 respectively. Note that $E_{PI(k)}$ and $E'_{PI(k)}$ are of dimensions N blocks by N blocks, to remain generic for Mode 1 ($N = N_{M1}$) or Mode 2 ($N = N_{M2}$) use.

$$X = \begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+N_{M1}-1} \\ x_{k+N_{M1}} \end{bmatrix} = M_{PI(k)}x_k + C_{PI(k)}\boldsymbol{c} + E_{PI(k)}\boldsymbol{w}$$	Equation 90
$$U = \begin{bmatrix} u_k \\ u_{k+1} \\ u_{k+2} \\ \vdots \\ u_{k+N_{M1}-2} \\ u_{k+N_{M1}-1} \end{bmatrix} = M'_{PI(k)}x_k + C'_{PI(k)}\boldsymbol{c} + E'_{PI(k)}\boldsymbol{w}$$	Equation 91
$$\boldsymbol{w} = \begin{bmatrix} w_k \\ w_{k+1} \\ w_{k+2} \\ \vdots \\ w_{k+N-2} \\ w_{k+N-1} \end{bmatrix}$$	Equation 92

$$E_{PI(k)} = \begin{bmatrix} D_{PI(k)} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \Phi_{PI(k+1)}D_{PI(k)} & D_{PI(k+1)} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \Phi_{PI(k+2)}\Phi_{PI(k+1)}D_{PI(k)} & \Phi_{PI(k+2)}D_{PI(k+1)} & D_{PI(k+2)} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \left(\prod_{i=2}^{N-1}\Phi_{PI(k+N-i)}\right)D_{PI(k)} & \left(\prod_{i=2}^{N-2}\Phi_{PI(k+N-i)}\right)D_{PI(k+1)} & \left(\prod_{i=2}^{N-3}\Phi_{PI(k+N-i)}\right)D_{PI(k+2)} & \dots & D_{PI(k+N-2)} & \mathbf{0} \\ \left(\prod_{i=1}^{N-1}\Phi_{PI(k+N-i)}\right)D_{PI(k)} & \left(\prod_{i=1}^{N-2}\Phi_{PI(k+N-i)}\right)D_{PI(k+1)} & \left(\prod_{i=1}^{N-3}\Phi_{PI(k+N-i)}\right)D_{PI(k+2)} & \dots & \left(\prod_{i=1}^{N-j}\Phi_{PI(k+N-i)}\right)D_{PI(k+N-2)} & D_{PI(k+N-1)} \end{bmatrix}$$

Equation 93

$$E'_{PI(k)} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ K_{PI(k+1)}D_{PI(k)} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ K_{PI(k+2)}\Phi_{PI(k+1)}D_{PI(k)} & K_{PI(k+2)}D_{PI(k+1)} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ K_{PI(k+N-2)}\left(\prod_{i=3}^{N-1}\Phi_{PI(k+N-i)}\right)D_{PI(k)} & K_{PI(k+N-2)}\left(\prod_{i=3}^{N-2}\Phi_{PI(k+N-i)}\right)D_{PI(k+1)} & K_{PI(k+N-2)}\left(\prod_{i=3}^{N-3}\Phi_{PI(k+N-i)}\right)D_{PI(k+2)} & \dots & \mathbf{0} & \mathbf{0} \\ K_{PI(k+N-1)}\left(\prod_{i=2}^{N-1}\Phi_{PI(k+N-i)}\right)D_{PI(k)} & K_{PI(k+N-1)}\left(\prod_{i=2}^{N-2}\Phi_{PI(k+N-i)}\right)D_{PI(k+1)} & K_{PI(k+N-1)}\left(\prod_{i=2}^{N-3}\Phi_{PI(k+N-i)}\right)D_{PI(k+2)} & \dots & K_{PI(k+N-1)}\left(\prod_{i=2}^{N-j}\Phi_{PI(k+N-i)}\right)D_{PI(k+N-2)} & \mathbf{0} \end{bmatrix}$$

Equation 94

Inserting the prediction equations containing disturbance (Equation 90 and Equation 91) into the constraint equations (Equation 75, Equation 76 and Equation 77) results in Equation 95, Equation 96 and Equation 97.

$-\Delta X \leq M_{PI(k)}x_k + C_{PI(k)}\boldsymbol{c} + E_{PI(k)}\boldsymbol{w} \leq \Delta X$	Equation 95
$\underline{U} \leq U_{M1}^0 + M'_{PI(k)}x_k + C'_{PI(k)}\boldsymbol{c} + E'_{PI(k)}\boldsymbol{w} \leq \overline{U}$	Equation 96
$\underline{X} \leq X_{M1}^0 + M_{PI(k)}x_k + C_{PI(k)}\boldsymbol{c} + E_{PI(k)}\boldsymbol{w} \leq \overline{X}$	Equation 97

SMPC then becomes a problem given the stochastic disturbance terms of ensuring the constraints are satisfied with a specified probability P. An example of this is provided for the maximum deviation from state constraint (Equation 81) of Mode 1 predictions and is given by Equation 98, where $row\ n$ indicates the block row of the corresponding matrix and g_e^T is used to select a particular element of state from that block row. Similar analysis can be performed on any of the constraints in both Mode 1 and Mode 2.

$pr\left(g_e^T\left(M_{PI(k),row\ n}x_k + C_{PI(k),row\ n}\boldsymbol{c} + E_{PI(k),row\ n}\boldsymbol{w}\right) \leq g_e^T\Delta X_{row\ n} = \Delta x\right) \geq P$	Equation 98

As the term multiplied by $\boldsymbol{w}$ is the only stochastic component, the probability P that this stochastic component is no greater than a scalar $\beta_{state,PI(k),row\ n,e}$ can be asserted via Equation 99. This enables the constraints (in both Mode 1 and Mode 2) to be modified by the simple addition or subtraction of $\beta_{state,PI(k),row\ n,e}$ to take in to account the maximum disturbance for a given probability (Equation 100). However since the constraints of Equation 95 are two-sided and symmetric, if $g_e^T\Delta X_{row\ n} \leq \beta_{state,PI(k),row\ n,e}$ there is no longer a feasible set.

$pr\left(g_e^T(E_{PI(k),row\ n}\boldsymbol{w}) \leq \beta_{state,PI(k),row\ n,e}\right) \geq P$	Equation 99
$pr\left(g_e^T\left(M_{PI(k),row\ n}x_k + C_{PI(k),row\ n}\boldsymbol{c}\right) \leq g_e^T\Delta X_{row\ n} - \beta_{state,PI(k),row\ n,e}\right) \geq P$	Equation 100

Soft constraints and hard constraints correspond to $P < 1$ and $P = 1$ respectively. Determining $\beta_{state,PI(k),row\ n,e}$ (or any of its equivalents from Equation 95, Equation 96 or Equation 97) is explored in two different ways.

2.4.1.1. Using Probabilistic constraints (PSMPC) to generate soft constraints

To determine for a probability P the value $\beta_{state,PI(k),row\ n,e}$ that the stochastic component element $g_e^T E_{PI(k),row\ n}\boldsymbol{w}$ is no greater than, the probability distribution of $E_{PI(k),row\ n}\boldsymbol{w}$ must be determined. For multivariate random variables made up of the summation of other multivariate random variables (such as

$E_{PI(k),row\ n}\boldsymbol{w}$) this involves convolving the distributions of each of these individual multivariate random variables. However this convolution is avoided due to the following assumptions (Section 1.2 (page 12)):

- Due to the truncation of the winds $\chi^2(3)$ distribution being small (1%) the probability distribution of w is assumed to be approximately a multivariate normal, which has the property that the sum of multivariate normal probability distributions is another multivariate normal probability distribution.
- Each wind vector is independent of all others (therefore any cross expectation terms are zero (Equation 24)).

Therefore covariance matrices of each disturbance component of $E_{PI(k),row\ n}\boldsymbol{w}$ can be summed for the calculation of a covariance matrix for each prediction block row (via the recursive formula; Equation 101).

$\mathbb{E}\left((E_{PI(k),row\ n}\boldsymbol{w})((E_{PI(k),row\ n}\boldsymbol{w}))^T\right) = \Omega_{state,PI(k),row\ n,Total}$ $= \Phi_{PI(k+n-1)}\Omega_{state,PI(k),row\ n-1,Total}\Phi^T_{PI(k+n-1)}$ $+ D_{PI(k+n-1)}\Omega_w D^T_{PI(k+n-1)}$	Equation 101

To explain the notation used here, consider the 2-steps-ahead prediction starting from position k on the loop. Equation 102 gives the corresponding covariance matrix (component covariance matrices $\Omega_{state,PI(k),row\ 2,0}$ and $\Omega_{state,PI(k),row\ 2,1}$ are given in Equation 103 and Equation 104 respectively). $Total$ represents all the disturbance component terms of $E_{PI(k),row\ n}\boldsymbol{w}$ if a number is present (i) it indicates the number of times this disturbance (w_i) has been projected through the state equations.

$\Omega_{state,PI(k),row\ 2,Total} = \Omega_{PI(k),row\ 2,2} + \Omega_{PI(k),row\ 2,1}$	Equation 102
$\Omega_{state,PI(k),row\ 2,0} = D_{PI(k+1)}\Omega_w D^T_{PI(k+1)}$	Equation 103
$\Omega_{state,PI(k),row\ 2,1} = \Phi_{PI(k+1)}D_{PI(k)}\Omega_w D^T_{PI(k)}\Phi^T_{PI(k+1)}$	Equation 104

Once a covariance matrix have been generated, the value of $\beta_{state,PI(k),row\ n,e}$ (which is labelled $\gamma_{SMPC,state,PI(k),row\ n,Total,e}$ to avoid confusion in the next section) from Equation 99 can be found using standard normal distribution tables as shown in Equation 105 where $\Phi^{-1}_{normal}(P)$ is taken to be the inverse cumulative normal distribution operator (with mean of zero and standard deviation of 1). Here the g_e vectors are used to select the appropriate diagonal element of $\Omega_{state,PI(k),row\ n,Total}$, corresponding to a state element e.

$\gamma_{SMPC,state,PI(k),row\ n,Total,e} = \sqrt{g_e^T \Omega_{PI(k),row\ n,Total} g_e}\ \Phi_{normal}^{-1}(P)$	Equation 105

2.4.1.2. Using Feasibility constraints (FSMPC) to generate hard constraints

The analysis presented in the previous section where probabilistic constraints were formed by taking the inverse function of a normal distribution cannot be used to generate hard constraints as $\Phi_{normal}^{-1}(1) = \infty$. If the true truncated distribution had been used at that stage it would result in a finite number for $P = 1$. However using prior knowledge of the finite support of the wind model it is possible to evaluate the worst possible disturbance and form hard constraints. For state deviation from reference constraints (Equation 95) the worst-case disturbances can be found by maximizing (for upper constraints Equation 106) and minimizing (for lower constraints Equation 107) the disturbance term of the relevant row of Equation 95.

$g_e^T (M_{PI(k),row\ n} x_k + C_{PI(k),row\ n} \boldsymbol{c} + \max_{\boldsymbol{w}}(E_{PI(k),row\ n} \boldsymbol{w})) \leq g_e^T \Delta X_{row\ n} = \Delta x$	Equation 106
$-\Delta x = -g_e^T \Delta X_{row\ n} \leq g_e^T (M_{PI(k),row\ n} x_k + C_{PI(k),row\ n} \boldsymbol{c} + \min_{\boldsymbol{w}}(E_{PI(k),row\ n} \boldsymbol{w}))$	Equation 107

Optimizations to determine the worst-case disturbances (subject to the wind being finitely supported) are defined in Equation 108 and Equation 109. New notation is introduced in a similar way to Section 2.4.1.1. If these max disturbance constraints are introduced in Equation 100 with $\beta_{state,PI(k),row\ n,e}$ as $\overline{a}_{state,PI(k),row\ n,Total,e}$ they act as hard constraints as it is ensured that the constraint is met even if the worst-case wind sequence is realised.

$\overline{a}_{state,PI(k),row\ n,Total,e} = \max_{\boldsymbol{w}}(g_e^T E_{PI(k),row\ n} \boldsymbol{w})$ $s.t.\ (w_i^T \Omega_w^{-1} w_i \leq r^2)\ \forall\ i = k, k+1, \dots, k+N$	Equation 108
$\underline{a}_{state,PI(k),row\ n,Total,e} = \min_{\boldsymbol{w}}(g_e^T E_{PI(k),row\ n} \boldsymbol{w})$ $s.t.\ (w_i^T \Omega_w^{-1} w_i \leq r^2)\ \forall\ i = k, k+1, \dots, k+N$	Equation 109

The total disturbance of a state element is the summation of a new disturbance element and the propagation of older ones. Maximizing the sum of the components for one element is the same as maximizing each of the individual components (for one element) and summing. To show this clearly for the maximum deviation from state for the 2-steps-ahead prediction starting at k on the loop for an element (e), Equation 110 is provided with the component elements shown in Equation 111 and Equation 112.

$\overline{a}_{state,PI(k),row\ 2,Total,e} = \max_{\boldsymbol{w}}(g_e^T E_{PI(k),row\ 2} \boldsymbol{w})\ s.t.\ (w_i^T \Omega_w^{-1} w_i \leq r^2)\ \forall\ i = k, k+1$ $= \overline{a}_{state,PI(k),row\ 2,1,e} + \overline{a}_{state,PI(k),row\ 2,0,e}$	Equation 110

$\overline{a}_{state,PI(k),row\ 2,1,e} = \max_{\boldsymbol{w}}(g_e^T \Phi_{PI(k+1)} D_{PI(k)} \boldsymbol{w})\ s.t.\ \left(w_k^T \Omega_w^{-1} w_k\ \leq r^2\right)$	Equation 111
$\overline{a}_{state,PI(k),row\ 2,0,e} = \max_{\boldsymbol{w}}(g_e^T D_{PI(k+1)} \boldsymbol{w})\ s.t.\ \left(w_{k+1}^T \Omega_w^{-1} w_{k+1}\ \leq r^2\right)$	Equation 112

These optimizations can be performed analytically. The maximization of an arbitrary state disturbance element (e) (element $row\ n, column\ j$ of Equation 93) is considered in Equation 113; of course the analysis could be performed on input disturbance similarly, and in both cases minimizations are simply performed by maximizing the negative of the disturbance element.

$\overline{a}_{state,PI(k),row\ n,n-j,e} = \max_{w_j}\left(g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}w_n\right) s.t.\ \left(w_j^T\Omega_w^{-1}w_j\ \leq r^2\right)$ $= g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}w_j^*$	Equation 113

At the maximum point the single constraint is active (Equation 114) and the wind is aligned (Equation 115) with the cost function gradient with λ being some scalar. Substituting Equation 115 into Equation 114 results in Equation 116 which when solving for λ results in Equation 117.

$(w_j^*)^T \Omega_w^{-1} w_j^* = r^2$	Equation 114
$w_j^* = \lambda D_{PI(k)}^T \left(\prod_{i=1}^{n-j}\Phi_{PI(k+N-i)}\right)^T g_e$	Equation 115
$\lambda^2\left(g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}\right)\Omega_w^{-1}\left(D_{PI(k)}^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)^T g_e\right) = r^2$	Equation 116
$\lambda = \frac{r}{\sqrt{\left(g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}\right)\Omega_w^{-1}\left(D_{PI(k)}^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)^T g_e\right)}}$	Equation 117

Substituting Equation 117 into Equation 115 results in Equation 118 which when substituted into Equation 113 results in Equation 119 (remembering r is a scalar).

$w_j^* = \frac{r D_{PI(k)}^T \left(\prod_{i=1}^{n-j}\Phi_{PI(k+N-i)}\right)^T g_e}{\sqrt{\left(g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}\right)\Omega_w^{-1}\left(D_{PI(k)}^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)^T g_e\right)}}$	Equation 118
$\overline{a}_{state,PI(k),row\ n,n-j,e} = \frac{r g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}D_{PI(k)}^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)^T g_e}{\sqrt{\left(g_e^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)D_{PI(k)}\right)\Omega_w^{-1}\left(D_{PI(k)}^T\left(\prod_{i=1}^{n-j}\Phi_{PI(k+n-i)}\right)^T g_e\right)}}$	Equation 119

Combining common terms in Equation 119 and remembering Ω_w is diagonal, and therefore only a single scalar term on the diagonal of Ω_w^{-1} will be selected, results in Equation 120, a worst-case state disturbance component element.

$$\overline{a}_{state,PI(k),row\ n,n-j,e} = r\sqrt{g_e^T \Omega_w g_e}\sqrt{g_e^T \left(\prod_{i=1}^{n-j} \Phi_{PI(k+n-i)}\right) D_{PI(k)} D_{PI(k)}^T \left(\prod_{i=1}^{n-j} \Phi_{PI(k+n-i)}\right)^T g_e}$$	Equation 120

2.4.1.3. Using feasibility constraints (FSMPC) to generate soft constraints

Intuitively the probabilistic constraints (PSMPC, Section 2.4.1.1) can be seen as inadequate. This is because the control method can allow constraints to be broken to such an extent that the next optimization step may not be feasible. But it is possible to combine feasibility constraints with the probabilistic constraints to generate soft constraints which are recursively feasible. The formal proof for a non-time-varying system provided in (12) is quite involved, so the method is presented intuitively here, via a series of points (the argument is made for state constraints but could equally apply to input constraints). Note Matrix 14 (maximum constraint needed for soft feasible constraints) of (12) *was* explicitly computed, confirming that the diagonal terms are the largest in every column, therefore the simplified analysis presented here is valid.

1. Via the prediction models, the disturbance is taken to be the superposition of past wind inputs fed through the recursive model and new wind input (Equation 90). At a given prediction time the disturbance is therefore a summation of disturbance components, each of which is a multivariate random variable.
2. A method has been presented to calculate the worst-case disturbance at a prediction step (Section 2.4.1.2). This method maximized/minimized the individual disturbance component elements (given that the original multivariate random variable (w) has finite support). These component elements were then summed together.
3. Section 2.4.1.1 presents a method to calculate for a given probability P, the value that the stochastic disturbance of a prediction step is no greater than. This method involves convolving the probability density functions of the multivariate random variables (the individual disturbance components) which make up the total disturbance, to form the probability density function of the total disturbance. For a

probability P the value each element of the total distrubance is to be no greater than was then evaluated using this probability density function.

4. Combining Point 2 and Point 3, constraints can be modified such that the constraints are allowed to be violated up to a given frequency and still remain feasible even if all past disturbances are worst-case. In this method Point 2 is applied to all the past disturbances at a prediction, but the disturbance at the current time for this prediction is evaluated probabilistically via Point 3. For the next prediction again all past disturbances are evaluated as worst-case (which now includes the wind from the previous prediction) the term evaluated probabilistically this time is a new disturbance component corresponding to the current prediction.
5. To explain Point 4 in more detail: At state prediction x_{k+2} the total disturbance is: $\Phi_{PI(k+1)}D_{PI(k)}w_k + D_{PI(k+1)}w_{k+1}$. For a soft constraint associated with the state prediction to remain feasible for all past wind realisations it must be assumed that the past disturbance ($\Phi_{PI(k+1)}D_{PI(k)}w_k$ term) was worst-case (and as such is evaluated using Point 2). But this is a soft constraint and as such it may be violated with given probability, therefore the disturbance component associated with the current wind ($D_{PI(k+1)}w_{k+1}$ term) is evaluated using Point 3. The contribution from Point 2 and Point 3 are then summed together elementwise (e) to create $\beta_{State,PI(k),row\ 2,e}$. For the next state prediction x_{k+3} the total disturbance is: $\Phi_{PI(k+2)}\Phi_{PI(k+1)}D_{PI(k)}w_k + \Phi_{PI(k+2)}D_{PI(k+1)}w_{k+1} + D_{PI(k+2)}w_{k+2}$. Again, to ensure that the state prediction, remains feasible for all past wind realisations for this prediction it must be assumed that the past disturbance components ($\Phi_{PI(k+2)}\Phi_{PI(k+1)}D_{PI(k)}w_k$ and $\Phi_{PI(k+2)}D_{PI(k+1)}w_{k+1}$) were worst-case (and as such are evaluated using Point 2), including the term evaluated probabilistically previously ($D_{PI(k+1)}w_{k+1}$), propagated through the recursive model ($\Phi_{PI(k+2)}D_{PI(k+1)}w_{k+1}$). A soft constraint then requires the probability distribution of $D_{PI(k+2)}w_{k+2}$ be evaluated using Point 3. Summing these results elementwise (e) generates $\beta_{State,PI(k),row\ 3,e}$.
6. This method produces recursively feasible soft constraints since a state prediction is allowed to break a constraint to a given probability as a result of the current disturbance term associated with that prediction. However when that same disturbance term is used in a prediction in the future, it is assumed that the worst-case was realised.

2.4.2. Limitations and inherent assumptions

The probabilistic constraints were evaluated based on the wind disturbance being a multivariate normal distribution so that the probability distribution of disturbance predictions could be obtained by summing covariance matrices instead of convolving probability density functions. The wind model used is really a truncated $\chi^2(3)$ distribution with modified covariance. However the approximate multivariate normal distribution will tend to exaggerate the probability of the tail, and hence provide conservative soft constraints. Probability of constraint violation and rate of infeasibility were checked numerically as described in Test 9 of Table 12 (Appendix A: Algorithm verification/Checks performed, page 46).

2.4.3. Implementation considerations

Modifying a working DMPC online controller is simple once the correct constraint terms have been generated. However the amount of offline computation to calculate these constraints does increase significantly, but perhaps more importantly the level of programming complexity for FSMPC constraint generation is significantly greater, leading to the need for thorough verification of the code implementation (Appendix A: Algorithm verification/Checks performed page 46).

The Mode 2 predictions (Equation 86 and Equation 87) were updated to include the disturbance terms (Equation 93 and Equation 94) and hence Mode 2 constraint terms were modified to deal with the disturbance and the Mode 2 length was calculated using these.

(12) gives proofs that both the probabilistic and feasibility constraints converge for a non-time-varying system. However due to the time-varying periodic nature of the linear kite system the probabilistic and feasibility constraints were found to converge to a periodic sequence (note that in this implementation constraints were calculated explicitly at each time step).

As mentioned previously, due to the non-truncated multivariate normal distribution assumption used for the probabilistic constraints, hard constraints could not be implemented via setting $P = 1$ in soft constraint generation as this would set β (for any of the SMPC constraints) to ∞. They were handled instead via FSMPC (Section 2.4.1.2, page 30).

3. Preliminary data gathering needed for performance evaluation

3.1. Determining model sampling rate (T)

By running multiple tests of the nonlinear model controlled by an LQOpt controller and evaluating performance by computing average running cost (Equation 121) a linear model sampling rate was chosen.

$$J_{runningav} = \frac{1}{testlength}\left(\sum_{k=0}^{testlength-1} \left(x_k^T Q x_k + u_k^T R u_k\right)\right) \quad \text{Equation 121}$$

For a fair comparison of results, R and Q must be the same for each test. These were therefore chosen (Table 4) such that even at low sampling rates the nonlinear model was controlled adequately by LQOpt.

Name	Symbol	Value
State deviation weighting matrix	Q	I
Input deviation weighting matrix	R	0.01

Table 4: Model sampling rate LQOpt controller design parameters.

The result of such testing is shown in Figure 6. As a model accuracy/computational speed trade off was needed, 50 samples (0.16s between each sample) proved a good compromise on which to focus design.

Average running cost of LQOpt controllers (at a range of sampling rates) controlling a nonlinear kite

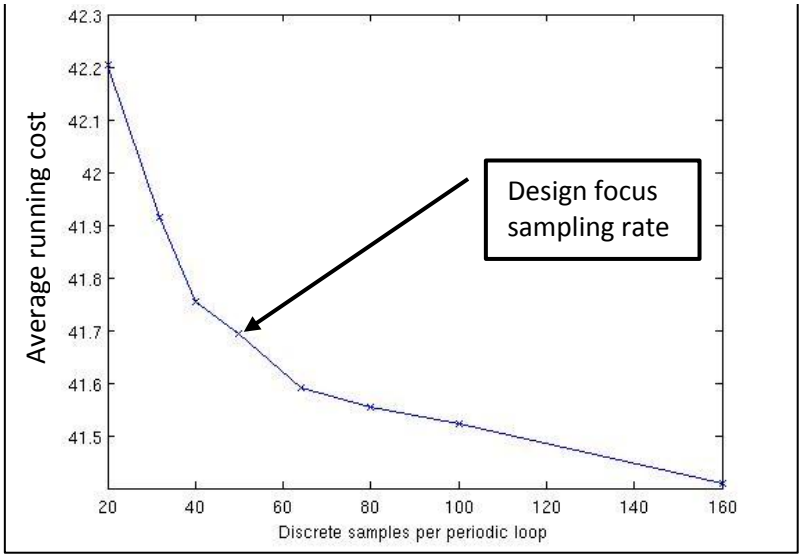

Figure 6: Average running cost of LQOpt controllers of various sampling rates controlling a nonlinear kite for 3600 loops.

3.2. Determining input constraints

The Initial kite model section describes the need to calculate α (see condition 3 of kite crashing p8). Tests were run to find the average max value of α (0.32). With an 80% factor of safety a constraint was defined; Equation 122.

$$-60° \leq u_C^{total}(t) \leq 60° \quad \text{Equation 122}$$

4. Controller comparison

4.1. Performance evaluation methods and test descriptions

4.1.1. Testing methodology and results gathered

The controllers were tested controlling both linear and nonlinear kite models over extended periods of time. Due to resource limitations only a single test was performed for each, which is recognised as being non-ideal. Due to the random nature of the disturbance, each time the test is run a different realisation of the wind occurs and therefore a different set of results are produced. Instead of a single test, the same test should be run multiple times (with different wind realisations) and an average taken. This is especially true with the tests which ended quickly due to the kite crashing.

Controller performance and model performance was evaluated by comparing various measures:

- Number of loops completed before the kite crashes (if it does at all).
- Calculation of average running cost (Equation 121).
- When the kite passed through a periodic position the state components were recorded and compared to the soft state constraints. This enabled the calculation of the percentage of time each state component of each periodic position was breaking constraints. After the test the maximum value was taken as a performance measure and named ‘max % time a soft constraint was broken’.
- For MPC controllers each time the kite passed through a periodic position the status of the QP solver was recorded. This enabled the calculation of the percentage of time that the QP solver found the optimization to be infeasible at each position in the loop. After the test the maximum value was taken as a performance measure and named ‘max % time MPC optimization is infeasible’.

Linear model accuracy at approximating the nonlinear system was evaluated by comparing how similar the performance measures were between the linear and nonlinear models when running the same controller.

After each test was run the data for the first 10 loops was discarded. If the kite crashed, the data for the last 10 loops were discarded before processing. Therefore extreme data due to crashing and initial transients are not included in the performance evaluation.

4.1.2. Systems, controllers and constraint combinations evaluated

4.1.2.1. Wind model

The wind model shown in Table 5 was used for all performance evaluations presented.

Name	Symbol	Value
Wind covariance	Ω	$\begin{bmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.1 \end{bmatrix}$
Wind mean velocity	w^0	$[6,0,0]^T$ ms^{-1}
Wind probability truncation	$\overline{p}$	0.99
Wind constant time period	T_w	T (model sampling period)

Table 5: Common wind model parameters.

4.1.2.2. Constraints of MPC optimizations

The common constraints used for controller design are shown in Table 6. The hard constraints are included to prevent the kite from crashing, as described on page 8 (Initial kite model). The other state hard constraints were chosen to be large so not to affect optimization. They are not kite crash modes.

Name	Symbol	Value
State hard constraint upper	$\overline{x}$	[1.5708rad,50rad,50rads^{-1},50 rads^{-1}]^T
State hard constraint lower	$\underline{x}$	-[1.5708rad,50rad,50rads^{-1},50 rads^{-1}]^T
Input hard constraint upper	$\overline{u}$	60°
Input hard constraint lower	$\underline{u}$	-60°

Table 6: Common constraints. Note: Hard constraints are interpreted differently for different controllers. DMPC uses the constraints with no modifications. Both PSMPC and FSMPC use the same hard constraints (modified to be feasible under any wind sequence).

Controllers were then compared with two sets of soft constraints (Test A and Test B) of max deviation from state reference; Δx. These values are displayed in Table 7.

Name	Symbol	Value
State deviation soft constraint. **Test A**	Δx	[0.09rad,0.09rad,0.09rads^{-1},0.09rads^{-1}]^T
State deviation soft constraint minimum probability constraint is met. **Test A**	$[P, P, P, P]^T$	[0.8,0.8,0.8,0.8]^T
State deviation soft constraint. **Test B**	Δx	[0.02rad,0.02rad,0.02rads^{-1},0.02rads^{-1}]^T
State deviation soft constraint minimum probability constraint is met. **Test B**	$[P, P, P, P]^T$	[0.8,0.8,0.8,0.8]^T

Table 7: Test specific constraints. Note: Soft constraints are interpreted differently for the different MPC controllers. DMPC uses the constraints with no modifications. PSMPC uses the constraints with probabilistic constraints but no feasibility constraints. FSMPC uses the constraints with combined probabilistic and feasibility constraints, but does not convert them into hard constraints.

4.1.2.3. Controllers

Parameters common to all controllers are displayed in Table 8.

Name	Symbol	Value
State deviation weighting matrix	Q	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Input deviation weighting matrix	R	1

Table 8: Common controller weighting matrices.

To evaluate the effects of different sample rates and Mode 1 horizon lengths, controllers were sub divided into 'Controller-Bases' whose parameters are shown in Table 9.

Name	Symbol	Value
MPC Mode 1 steps. **Controller-Base A**	N_{M1}	50
Discrete model sampling period **Controller-Base A**	T	0.16s
MPC Mode 1 steps. **Controller-Base B**	N_{M1}	64
Discrete model sampling period **Controller-Base B**	T	0.125s
MPC Mode 1 steps. **Controller-Base C**	N_{M1}	100
Discrete model sampling period **Controller-Base C**	T	0.16s

Table 9: Controller-Base specific design parameters.

The specific controllers which were then generated on these Controller-Bases are listed in Table 10.

Controller type	Controller code	Controller-Base		
		A	B	C
Time-varying periodic optimal linear feedback	LQOpt	✓	✓	✓
Time-varying periodic H_∞ auxiliary linear feedback	H_∞	✓	✓	✓
Deterministic MPC pre-stabilized by time-varying periodic optimal linear feedback	DMPC LQOpt	✓	×	×
Deterministic MPC pre-stabilized by time-varying periodic H_∞ auxiliary linear feedback	DMPC H_∞	✓	✓	✓
Probabilistic constrained stochastic MPC pre-stabilized by time-varying periodic H_∞ auxiliary linear feedback	PSMPC H_∞	✓	✓	✓

Table 10: The controllers created on specific Controller-Bases.

4.1.2.4. Note on FSMPC

Feasibility constraints on the soft constraints (state deviation) grow so large that they remained infeasible (they failed Test 8 of Table 12, page 46; Appendix A: Algorithm verification/Checks performed) before implementation on the controller (for all but the lowest wind covariance and largest wind truncation levels). As such no controllers are presented in this section. However hard constraints were implemented using feasibility constraints as described in Section 2.4.1.2 page 30. Hard constraints in this case are well beyond where the kite normally operates so Test 8 of Table 12 was passed. Further code verification of feasibility constraint generation was continuing while testing for the data presented here commenced. Likely impact on coding errors is discussed on page 47 (Note on Feasible Stochastic MPC (FSMPC) code verification).

4.2.Test results

A full spread sheet of the processed results is included in the attached DVD (page 49).

4.2.1. Preventing kite crashing performance

All MPC algorithms failed to control the nonlinear kite. The number of loops each kite completed before crashing is displayed in Figure 7. Figure 8 shows kite position during nonlinear Test A while being controlled using Controller-Base A PSMPC H_∞; here the kite crashes due to too large a control input. Kite crashing when under MPC based control is characterised by the QP solver being unable to find a solution to the optimization problem for an extended period of time. Figure 7 shows that PSMPC H_∞ is more robust to stochastic disturbances than DMPC H_∞ as PSMPC H_∞ was able to control the kite for a greater than or same number of loops for all test cases.

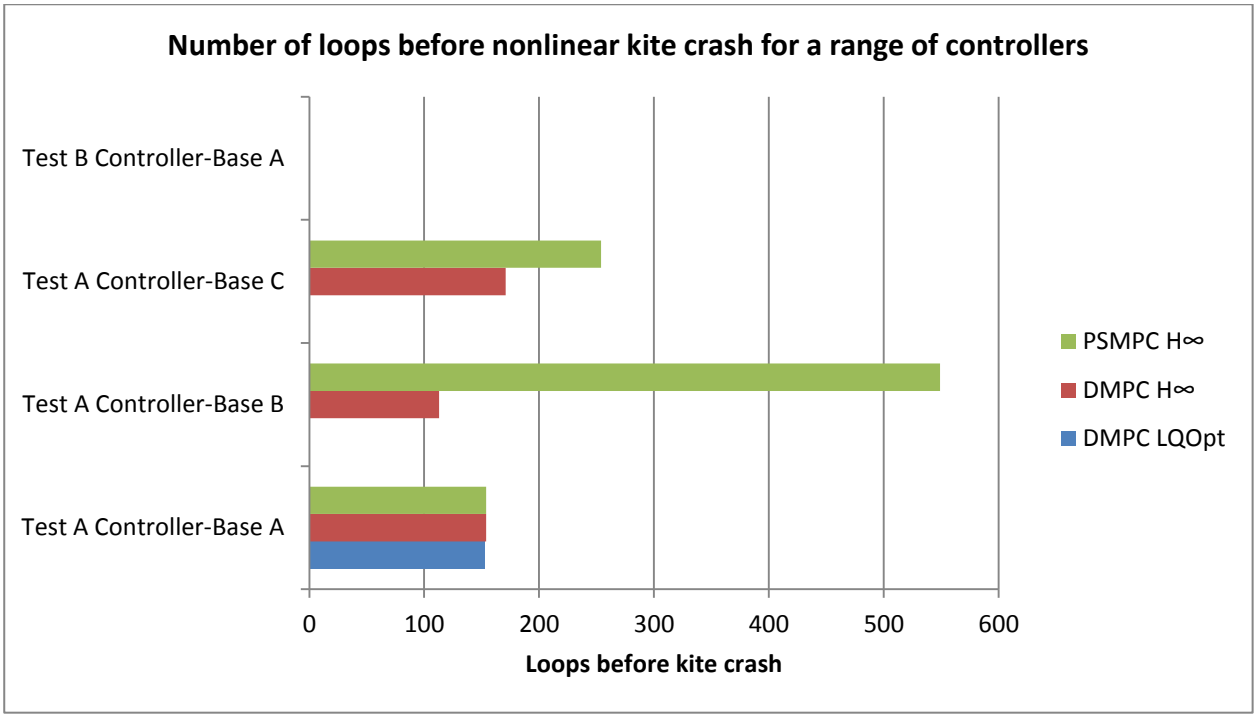

Figure 7: Number of complete loops completed by the nonlinear kite model before crashing under various MPC controllers. Note 1: DMPC LQOpt was only used with Test A Controller-Base A. Note 2: During Test B MPC controllers using Controller-Base A were unable to control the kite for a single loop.

However all the linear feedback controllers were able to control the nonlinear kite without crashing. Figure 9 displays the same test as Figure 8 but this time using the Controller-Base A H_∞ controller. In this case the kite is prevented from crashing, and remains close to the reference.

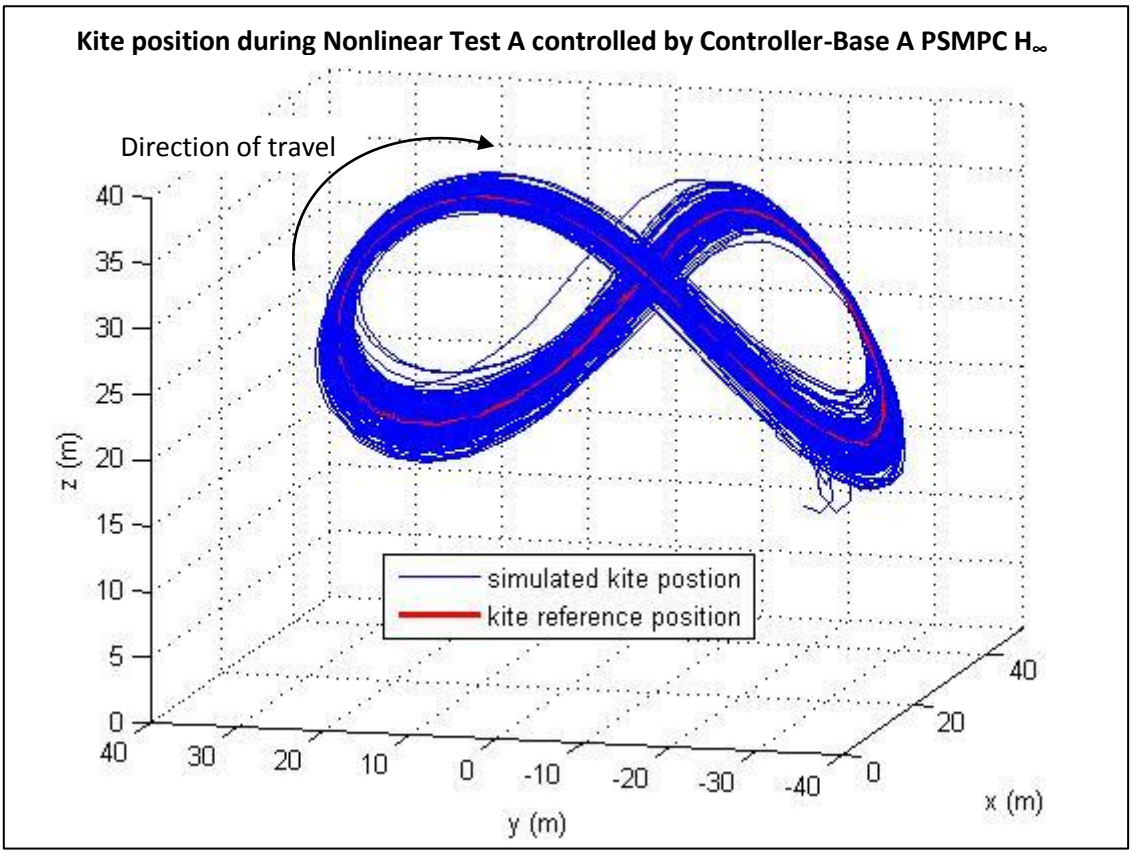

Figure 8: Kite positions during Nonlinear Test A being performed using Controller-Base A PSMPC H∞ controller (154 kite loops).

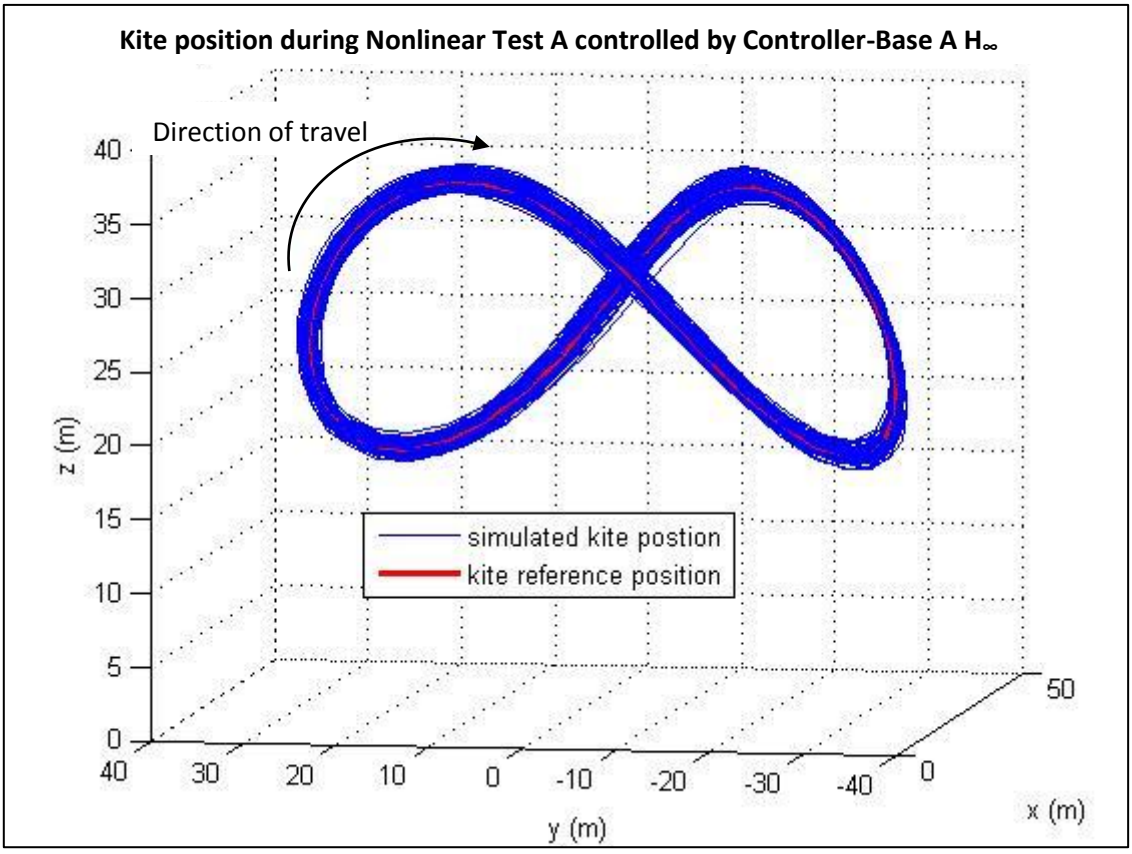

Figure 9: Kite positions during Nonlinear Test A being performed using Controller-Base A H∞ controller (21786 kite loops)

4.2.2. Accuracy of linear model approximation

Figure 10 displays the average running cost and Figure 11 'max % time a soft constraint was broken' for a range of controllers running Test A and Test B. Comparing results for the linear and nonlinear tests shows that performance is significantly worse on the nonlinear kite compared to the linear simulation. Clearly the linear model is not a good approximation to the nonlinear model.

If based purely on a linearized model, Test A would result in the selection of an LQOpt controller due to it experiencing no constraint violation and offering a lower cost (0.0017) than the other linear feedback controller (H_∞ 0.0076). However on implementation on the nonlinear kite performance is degraded (0% -> 0.91%) with regard to constraint violations, making the higher average running cost of H_∞ worthwhile.

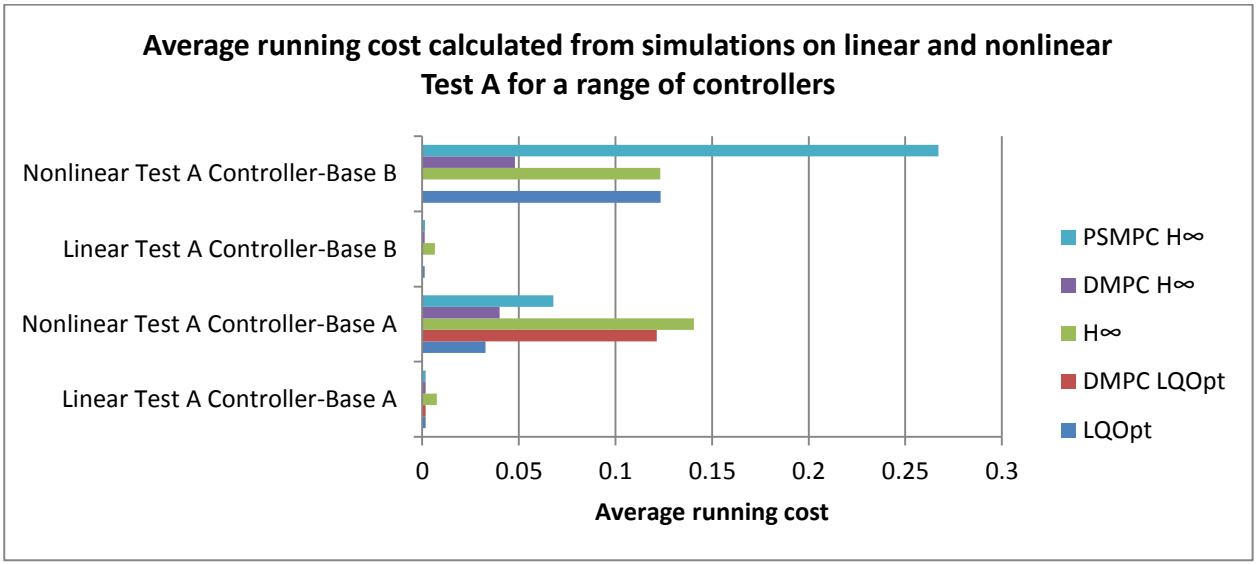

Figure 10: Average running cost calculated from simulations on linear and nonlinear Test A for a range of controllers. Note: DMPC LQOpt was only used with Test A Controller-Base A.

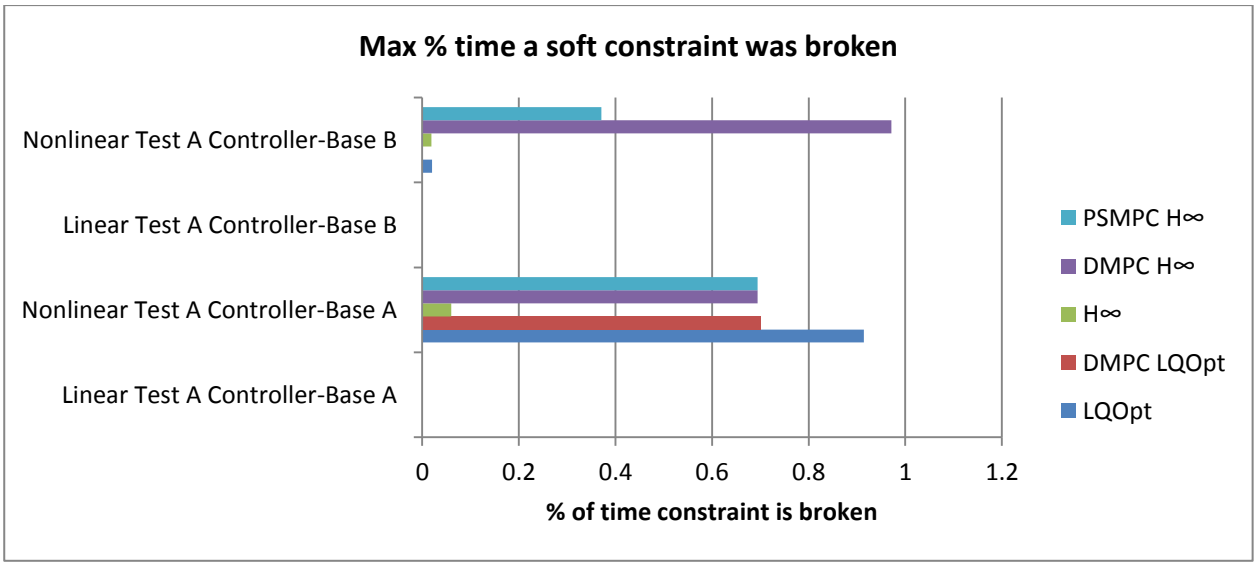

Figure 11: Max % time a soft constraint was broken calculated from simulations on linear and nonlinear Test A for a range of controllers. Note: DMPC LQOpt was only used with Test A Controller-Base A.

Figure 12 displays the results of the linear testing in Figure 10 at a more appropriate scale. It shows that the average running costs of all the MPC controllers was lower than that of the H_∞ linear feedback. The same cost of the DMPC schemes (based on any feedback controller) to LQOpt indicates that their constraints were never active on Linear Test A. As such (confirmed by inspection of recorded kite input data) all the MPC controllers with H_∞ pre-stabilization compensated for this non-optimal linear feedback and corrected it to deliver an input to the kite which was the same as the LQOpt controller. This indicates that the inadequacy of the MPC approach is mainly due to the poor approximation of the linear model to the nonlinear model (i.e. MPC algorithms presented are not robust enough to nonlinearity). However the higher cost for PSMPC indicates that constraints were active on optimization at some point under test.

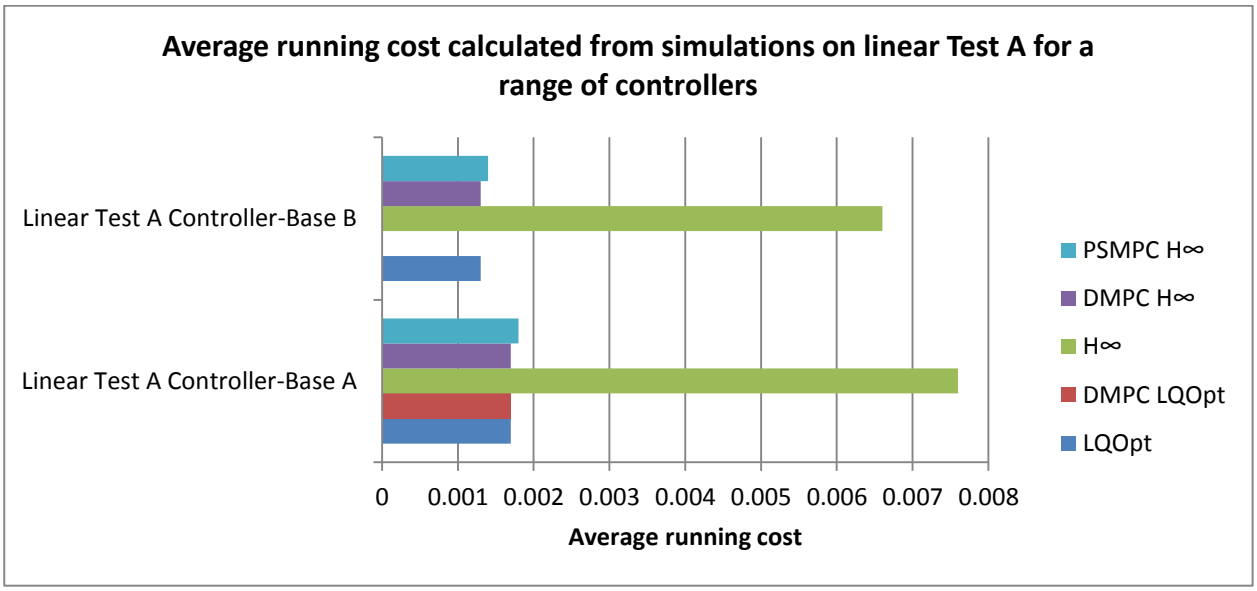

Figure 12: Average running cost calculated from simulations on linear Test A for a range of controllers (same as that presented in Figure 10). Note: DMPC LQOpt was only used with Test A Controller-Base A.

4.2.3. Level of infeasibility in MPC optimizations

Figure 13 shows for MPC controllers, DMPC H_∞ and PSMPC H_∞ when controlling the linear kite under Test A constraints no infeasibility was experienced for these 39330 and 52337 respective loop tests. The length of these tests indicates that if the linear model was a good approximation to the nonlinear kite model, then these controllers could likely be used with a low risk of infeasibility (even without feasibility terms on the soft constraints). However, changing the constraints to those of Test B does result in infeasibility, such that the PSMPC H_∞ crashes the kite after 793 loops, performing worse than the DMPC H_∞. This is as a result of the constraint tightening on the PSMPC H_∞ which makes the feasible set smaller. This indicates that feasibility constraints are required for the soft constraints.

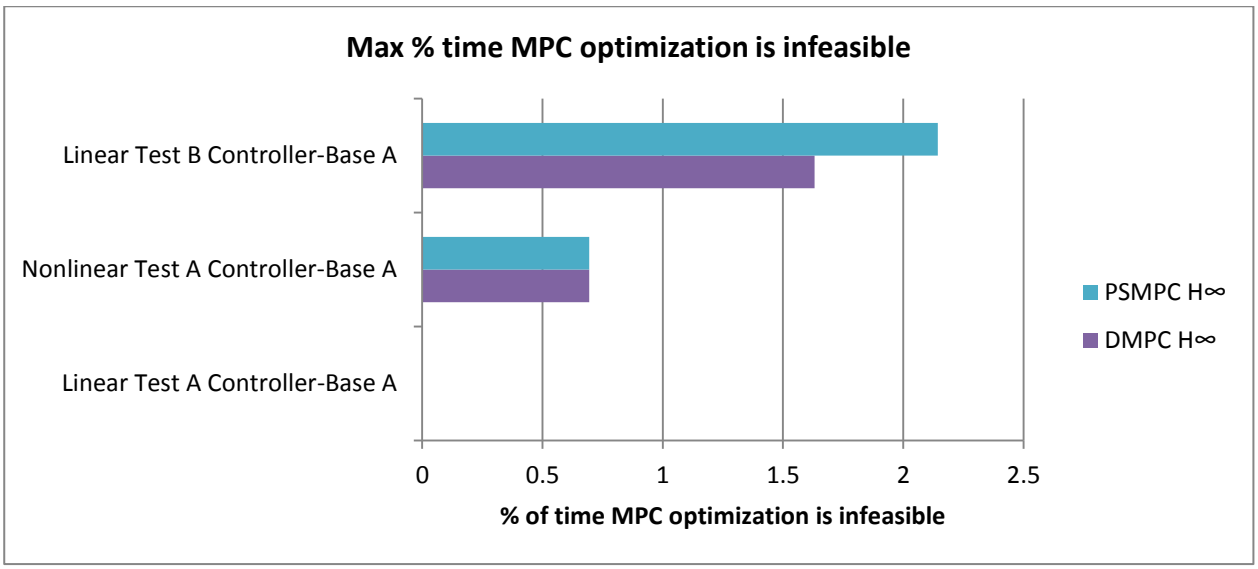

Figure 13: Max % time MPC optimization is infeasible for Controller-Base A MPC controllers running Test A and Test B.

4.3.Controller selection

As no MPC based controller was able to control the nonlinear kite subject to the constraints of Test A or Test B a MPC controller in its current form cannot be recommended. Therefore choice of the most suitable controller is a choice between the two linear feedback controllers (LQOpt and H_∞) with decision criteria being based on which breaches the soft constraints the least. For nonlinear Test B the 'max % time a soft constraint was broken' for LQOpt and H_∞ was 35.2% and 24.6% respectively. Therefore both exceed the 20% set by the bounds P in Table 7 for the SMPC controller. The greater soft constraint satisfaction of the H_∞ comes at the expense of a higher average running cost (0.1406 vs. 0.0328). However H_∞ provides performance closer to the design of 20% and so is selected as the controller of choice. Intuition is matched by computed results (Table 11) when comparing nonlinear Test A data from linear feedback controllers based on Controller-Base A (model sampling period 0.16s) and Controller-Base B (model sampling period 0.125s) which are otherwise identical. Increasing the number of samples improves performance on all measures.

Controller-Base	Average running cost	Max % time a soft constraint was broken
A	0.1406	0.0597
B	0.1232	0.0189

Table 11: Nonlinear Test A average running cost and % time a soft constraint was broken results for H_∞ linear feedback controllers.

Therefore, of the controllers developed, an H_∞ controller sampled at a very high frequency (no online optimization is required, so computational burden is lower compared to MPC) is recommended.

Conclusions and summary

Report summary: Activities carried out

To develop a control strategy for a continuous-time nonlinear kite model the following were carried out:

- A discrete-time-varying periodic linear state model was created from a provided linearized continuous-time nonlinear model.
- The nonlinear kite model simulation provided in (5) was modified to create a linear discrete-time simulation model for the purpose of testing controllers.
- The nonlinear kite model simulation provided in (5) was modified to be controlled by the controllers developed in this report, to test robustness of controllers to nonlinearity.
- From the linear model multiple control strategies were developed and implemented in MATLAB:
    - Discrete-time-varying periodic linear feedback quadratic regulator optimal control (LQOpt).
    - Discrete-time-varying periodic H_∞ auxiliary linear feedback control (H_∞).
    - Pre-stabilized time-varying periodic deterministic linear model predictive control (DMPC).
    - Pre-stabilized time-varying stochastic model predictive control (SMPC). With soft constraints in the form of:
        - Probabilistic constraints (PSMPC).
        - Feasibility constraints (FSMPC).
- Performance was evaluated with the controllers controlling both linear and nonlinear kites.

Report summary: Results

- For this dynamic system/disturbance combination worst-case disturbance constraints generated by FSMPC are infeasible; as a result, FSMPC is not a viable control strategy (Section 4.1.2.4, page 38).
- PSMPC offers promise of being a better control strategy for the cases in which a linear dynamic system can adequately approximate the actual nonlinear system. However provision must be made for the times when the online optimizer cannot find a feasible solution.
- A discrete-time-varying periodic H_∞ auxiliary linear feedback controller offered the best performance and, for the given nonlinear kite model, is recommended as the controller of choice.

Recommendations for future work: Model improvements

- The current wind model is not based on observed physical phenomena, therefore a more accurate representation should be implemented. (1) is identified as a source of such information.
- The current nonlinear model assumes the kite remains on the same line length at all times. Strategies to extract power from the kite ((9) and (3)) include the kite periodically being dragged in by the controller and then pulled out by the wind. This should be incorporated in the model.
- Add a limit on the rate of change of input (along with additional hard constraints).
- Understand the physical basis by which states will be measured to form a 'C' matrix from $y = Cx$.
- Modify the discrete linear model generation process so the initial state is at the midpoint of the continuous samples used to generate the first discrete step of the linear model.

Recommendations for future work: Controller improvements

- A nonlinear model predictive controller should be developed, as was done by others: (3), (5) and (9)
- Section 4.2.2 suggests that the linear MPC designs are not robust enough to nonlinearity, therefore to improve performance on the nonlinear kite using linear controller design, further reference trajectories could be created to be linearised about. Straying too far from one reference would result in predictions being carried out on different linear models.
- Section 4.2.1 indicates that when infeasibility is detected, the online QP solver in MATLAB returns a very large value which saturates the control input. Experimenting with a control strategy where the online solver is monitored and if infeasibility is detected an alternative control strategy is implemented (such as H_∞) may result in a more robust controller to nonlinearity.
- Separate the MPC sampling and reference/linear feedback sampling rates. This would allow the reference and linear feedback to run at a higher sampling rate and improve performance, without the need to allow enough time for the MPC optimization to be performed.
- Use a controller which attempts to cancels out disturbance explicitly (13) (i.e. use a less conservative approach, deal with actual disturbance rather than predicted worst-case).

Recommendations for future work: Testing improvements

- Re-program the test scripts so that a test continues until a predefined number of loops per test have been completed. If the kite crashes the model is restarted (but the old data is retained).

Appendix A: Algorithm verification/Checks performed

Verification tests of correct implementation of control algorithms are given in Table 12.

Controller	Test #	Test	Status and test notes
LQOpt	1	$P_{PI(k)}$ values converge.	✓ Checked on controller generation.
H_∞	2	$P'_{PI(k)}$ values converge.	✓ Checked on controller generation.
H_∞	3	Check inequality is satisfied: $\max\left(eig\left(D_{PI(k-1)}^T P_{PI(k)} D_{PI(k-1)}\right)\right) < \gamma_{H_\infty PI(k-1)}^2$	✓ Checked on controller generation.
LQOpt / H_∞ pre stabilization	4	On the linear model with no wind disturbance. For a given start state and periodic kite position, run the model controlled with rolling horizon DMPC. Check the inequality (for predicted cost) is satisfied: $J^*(k+1) \leq J^*(k)\ \forall\ k$ Run the test for all periodic kite positions for both LQOpt and H_∞ pre-stabilization	× Test not performed.
DMPC	5	On the linear model with no wind disturbance. Set the constraints to below the reference levels. Check the state and input trajectories are clipped appropriately.	✓ Test performed. Correct result. Data on DVD (page 49).
DMPC	6	On the linear model with no wind disturbance. For a given initial start state, predict the optimal cost ($J^*(0)$). Run the model with rolling horizon MPC and calculate the running cost ($J_{running}$). Check the inequality is satisfied: $J_{running} \leq J^*(0)$	× Test performed only on early non-time-varying system. $J_{cl} > J^*(0)$ by 0.5%. Suggested failed due to rounding errors.
DMPC	7	Build into state predictions a deterministic disturbance. Check on running and see if constraints are never broken while the online optimizer remains feasible.	× Test not performed
SMPC	8	Before generating Mode 2 constraints check the inequality is satisfied and all other 'equivalent' inequalities for other constraints (Equation 100, page 28): $g_e^T \Delta X_{row\ n} \geq \beta_{state,PI(k),row\ n,e}$	✓ Checked on controller generation.
PSMPC	9	On the linear model with wind disturbance. Run multiple tests with the input generated offline for an initial state condition (do not implement rolling horizon). Over Mode 1 calculate max % time a soft constraint was broken. Vary wind covariance until the maximum violation is the same as requested by the probabilistic constraints or infeasibility becomes too frequent.	✓ Test performed for PSMPC. Correct result for x1 x2. Note however the wind model used was a non-truncated normal distribution. Data on DVD (page 49).
PSMPC	10	On the linear model with wind disturbance. Run multiple tests and calculate the average running cost $J_{runningav,end\ data}$ using data available at the end of the MPC horizon. This is l_{ss} from (12). With l_{ss} create a new Lyapunov terminal cost matrix and calculate the predicted cost ($J^*(0)$). Check the inequality is satisfied: $J_{runningav} \leq J^*(0)$	× Test not performed
FSMPC	11	On the linear model with truncated wind disturbance. Run the with FSMPC and PSMPC controllers on a set of very tight constraints (+0.01 from failing Test 8 in this	✓ FSMPC never goes infeasible (PSMPC never goes

		table for state deviation and max input). Compare max % time MPC optimization is infeasible.	infeasible) Data on DVD (page 49).
FSMPC	12	On the linear model with the wind disturbance only taking values corresponding to $w^T\Omega_w^{-1}w = r^2$ or w = $[0,0,0]^T$ (i.e. an extreme wind model). Run both FSMPC and PSMPC on a set of very tight constraints (+0.01 from failing Test 8 in this table for state deviation and max input). Compare max % time MPC optimization is infeasible.	× Max % time MPC optimization is infeasible = 0.08%, not 0% as would be expected and the kite still crashes. Thought that failure is due to numerical approximations in computation. Data on DVD (page 49).

Table 12: List of tests identified to check correct implementation of control algorithms and status of implementation.

Note on Feasible Stochastic MPC (FSMPC) code verification

A bug was discovered in the generation of feasibility constraints after testing had occurred for the results presented in Section 4, Controller comparison. This mistake underestimates feasibility constraints and on correction the state hard constraint (condition 1 of kite crashing page 8) makes the controllers presented in Section 4 infeasible. However the testing is taken as still being valid when the results of linear Test A and Test B (tighter soft constraints) are compared when controlled with PSMPC H_∞ of Controller-Base A.

Linear Test A never experiences infeasibility and the kite never crashes. Linear Test B, where the only change is the tightness of the soft state constraints, does experience infeasibility and the kite crashes. Therefore it could be inferred that the reason for the crash is the change in the PSMPC constraints, not the hard constraints, which were calculated using the incorrect code implementation. Further Test 12 of Table 12 was performed on both the old and updated implementation of FSMPC. Results are presented in Table 13. It shows that the old implementation still performed well. Further no infeasibility was found when Test 11 of Table 12 was run on the controllers in Table 13, indicating that, even if the code contains bugs, it still offers good enough performance for the presented results to be valid.

	Max % time MPC optimization is infeasible			
FSMPC version (used for hard constraints in PSMPC)	**PSMPC**	**# loops tested**	**FSMPC**	**# loops tested**
Old	0.2027	1973	0.1410	4964
Updated	0.1376	15257	0.0807	4959

Table 13: Results of Test 12 of Table 12 (the extreme wind test) applied to old and updated FSMPC versions.

Appendix B: MATLAB/Simulink modelling environment

Tips / Problems to avoid with Simulink modelling

- Intermittently the current time presented to a function by Simulink is slightly early. Therefore it is recommended to add a number smaller than the sampling period (T) to the time presented by Simulink and then round down to ensure the correct time is used.
- All aspects of an open Simulink file can be controlled from the MATLAB workspace. For example to modify the variable `seed` to `wind_seed_string` (a number stored as a string) in an open Simulink model `linear_kite` (which is nested in the blocks `Wind` and `Random Number`) the following line is used:

  ```
  set_param('linear_kite/Wind/Random Number','seed',wind_seed_string);
  ```

  This is of most importance to automated testing as it enables the provision of a new random number seed to Simulink which would otherwise only have a single random number sequence.

Computing resources needed for implementation

Code was written for ease of debugging and feature addition and is not particularly optimized for performance. Despite this the controller and model simulation ran at 5 seconds per loop (simulating 8 second loops) on a 1.6 GHz Celeron. Therefore, given that the simulation was running faster than real time the algorithm is fast enough to be run on a practical system, where the computational environment has been built for the task required.

The current code is very memory intensive and needs to be reviewed (for example a 2Gb RAM 32bit computer will not run the MPC controller generator scripts). This is especially true for:

- The use of long Mode 1 lengths or a large number of discrete samples per loop. These controllers result in structures containing a large number of constraints which are passed between different functions.

Currently a lot of data is stored after testing grouped with the model (720,000 loops of test data after processing $\approx$ 10 GB). Clarification of what data is to be retained after each test needs to be made.

Appendix C: Project DVD

The project DVD below is divided into two directories, their contents is described below:

- **MATLAB** (containing kite models, control algorithms and related documentation)
    - model_documentation.docx (Explains how to form controllers, models and perform simulations). Read this document for description of all the contents of **MATLAB**.
- **verification_and_results** (Note Table 12 is referred to frequently and is within Appendix A: Algorithm verification/Checks performed)):
    - controller_ comparison_data.xlsx (data exported from the testing conducted for Section 4).
    - DMPC_verification.docx (visual record of Test 5 of Table 12).
    - FSMPC_investigate_decision_data.docx (data from PSMPC running Test 9 of Table 12 where due to infeasibility FSMPC was subsequently investigated and implemented).
    - FSMPC_verification.xlsx (data from Test 11 and 12 of Table 12).
    - PSMPC_verification.xlsx (data from Test 9 of Table 12).
    - reference_trajectories_sampling_choice.docx (record of decision made in Section 1.1.2.2).

Appendix D: Risk assessment

Generic Display Screen Equipment Risk Assessment	Pages: 2
In Building: Department of Engineering Science	
Assessment undertaken by: R. Dodsworth (DSO) Signed: Richard Dodsworth	Date: 11/10/2010

Please Note: This is a generic Risk Assessment and highlights some common hazards identified with the use of Display Screen Equipment. Please note that:

Students should read the University Policy: http://www.admin.ox.ac.uk/safety/ups0809.shtml. This risk assessment is suitable for those 4th Year project students whose project only involves work on a computer.

Hazard	**Persons at Risk**	**Risk Controls In Place**	**Further Action Necessary To Control Risk**
Eyestrain/ Headaches	User	**Take regular breaks every hour**. - undertake a different task. - adjust screen location to prevent glare or bright reflections. - Angle screen downwards to prevent reflection. - ensure no screen flicker. - ensure screen surface is clean. - ensure lighting is adequate for the task. - have an eye test if problems persist. - close blinds to prevent glare (as appropriate)	Consult Supervisor and advise Departmental Safety Officer (DSO) if problems persist. Please refer to the following link for a picture of good posture: http://www.hse.gov.uk/pubns/indg36.pdf
Back pain	User	**Ensure Workplace is correctly set up** - e.g. height of chair needs to be set so that forearms are parallel to desk. - ensure good posture at all times, sitting upright or slightly reclining. - Lower back supported to maintain natural curves.	Refer any medical issues to Supervisor or Departmental Safety Officer (DSO)
Aching	User	**Check seat height is correct**	Refer any medical issues to Supervisor or Departmental

shoulders, wrists		- forearms horizontal, level with top of desk. - keep wrists straight, use wrist rest. - No overreaching, exercise muscles. - Arms relaxed by side.	Safety Officer (DSO)
Aching neck	User	**Check screen height is correct** - eyes level with top of screen. - use document holder. - exercise muscles. - Check chair height e.g. forearms horizontal, level with top of desk	
Aching legs	User	**Check space under desk** to stretch legs, feet rest comfortably on floor otherwise get footrest. - exercise muscles. - Knees level with pelvis or slightly below. - Feet flat on the floor or use a footrest.	Remove items under desk which are preventing correct use e.g. boxes.
Water/Liquids	User	Ensure no water containers, coffee machines, kettles etc are located on or in close proximity to your workstation.	Building Inspections.
240 VAC Electrical shock	User	User to check that all electrical leads to their PC are in good working order. Contact Electronics (Thom 5th floor) if Portable Appliance Label 'out of date' or not visible.	Supervisor/Student to check validity of PAT test label.

References

1. Burton T. 2001. *Wind energy handbook,* ed. T Burton, Chichester: Chichester : Wiley

2. Campi MC, Garatti S, Prandini M. 2008. *The scenario approach for systems and control design. Presented at Proceedings of the 17th World Congress, the International Federation of Automatic Control, Seoul, Korea, 2008,* 381. IFAC

3. Canale M, Fagiano L, Milanese M. 2010. High altitude wind energy generation using controlled power kites. *Control Systems Technology, IEEE Transactions on.* 18 : 279-93

4. Diehl M. 2001. *Real-time optimization for large scale nonlinear processes.* PHD. Heidelberg University, Heidelberg University.

5. Diehl M, Magni L, De Nicolao G. 2004. Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control.* 28 : 37-45

6. GE Power and Water. 2010. *4.0-110 offshore wind turbine brochure.*

7. Gilbert EG, Tan KT. 1991. Linear systems with state and control constraints: The theory and application of maximal output admissible sets. *Automatic Control, IEEE Transactions on.* 36 : 1008-20

8. Houska B. 2008. *Presentation: Robust nonlinear model predictive control for power generating kites. 2008,*

9. Houska B, Bock HG. 2007. *Robustness and stability optimization of open-loop controlled power generating kites.* PHD. Heidelberg University,

10. How JP. 2008. *16.323 principles of optimal control.* http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-323-principles-of-optimal-control-spring-2008/

11. Howatson AM. 1991. *Engineering tables and data,* ed. JD(D Todd, London: London : Chapman & Hall

12. Kouvaritakis B, Cannon M, Rakovic SV, Cheng Q. 2010. Explicit use of probabilistic distributions in linear predictive control. *Automatica.* 46 : 1719-24

13. Kouvaritakis B, Cannon M, Yadlin R. 2010. *On the centres and scalings of robust and stochastic tubes. Presented at UKACC Control 2010, Coventry University, 2010,*

14. Kreyszig E. 2006. *Advanced engineering mathematics,* ed. H Kreyszig, New York: New York : Wiley

15. Loyd ML. 1980. Crosswind kite power. *Journal of Energy.* 4 : 106-11

16. Makani Power. 2010. *Makani power - flights.* http://www.makanipower.com/flights/

17. MathsWorks T. *Documentation - simulink control design.* http://www.mathworks.com/help/toolbox/slcontrol/ug/bso7hby-1.html#bspks1b

18. Mayne DQ, Seron MM, Raković SV. 2005. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica.* 41 : 219-24

19. Yuan L, Achenie LEK, Jiang W. 1996. Robust H∞ control for linear discrete-time systems with norm-bounded time-varying uncertainty. *Systems & Control Letters.* 27 : 199-208