Swarming Fixed-Wing Trajectory Planning Using Distributed Model Predictive Control

Final Year Project

Oliver Wykes

Nomenclature

$\cdot^c$	Hard obstacle centre
$\cdot^r$	Reference state
$\cdot^v$	Virtual agent state
$\cdot_i$	Relating to agent i
$\hat{\mathbf{x}}$	Aircraft state vector
$\mathbf{c}_{eq}$	Equality constraint
$\mathbf{c}_{in}$	Inequality constraint
$\mathbf{f}(\cdot)$	Agent dynamics function
$\mathbf{g}(\cdot)$	State generator function
$\mathbf{u}$	Control vector
$\mathbf{u}^*$	Optimal control vector
$\mathbf{x}$	Trajectory state vector
$\mathbf{x}^*$	Optimal trajectory state vector
$\mathcal{N}_i$	Set of neighbouring agents of agent i
$\mathcal{N}_i^v$	Set of neighbouring virtual agents of agent i; used for soft obstacles
$\mathcal{U}_i$	Set of admissable control inputs for agent i
$\mathcal{X}_i$	Set of feasible states for agent i
k	Discrete time; $t = k\Delta t$
m	Number of control variables
n	Number of state variables
N_a	Number of agents
N_a^v	Maximum number of active virtual agents
N_c	Control horizon
N_o	Number of hard obstacles
N_p	Prediction horizon
O	Matrix weighting function for the hard inequality constraints
Q	Matrix weighting function for the agent tracking error
R	Matrix weighting function for the agent control input cost
S	Matrix weighting function for inter-agent repulsion
S_v	Matrix weighting function for repulsion between an agent and its detected virtual agents

Contents

1 Introduction

1.1 Motivation

Swarming behaviour has been observed in nature to confer significant benefits to animals due to its multiplicative effect: each animal is safer and capable of achieving more in a group than it is alone. Schooling fish and murmurations of starlings make each animal safer from predators, while colonies of ants are capable of moving objects significantly larger than themselves. The innumerable examples of swarming behaviour in nature highlight the fundamental advantages of this behaviour for modern robots.

Figure 1: A murmuration of starlings - an example of swarming behaviour in nature [1]

As research into multi-agent systems develops, numerous applications of the technology have been highlighted. Fundamental advantages of swarming imitate those found in nature. The large number of agents decreases the risk to each individual agent, thus making the swarm robust to failure; the swarm can occupy a wide area simultaneously, accelerating search and coverage problems; and information gathered at different points in the swarm may be synthesised to great effect. The latter two points form the basis of applications such as search and rescue [2], forest fire monitoring [3], aerial surveillance [4], nuclear radiation mapping [5], oil spill tracking [6], formation of wireless communication networks [7], and deep-space formation flying [8].

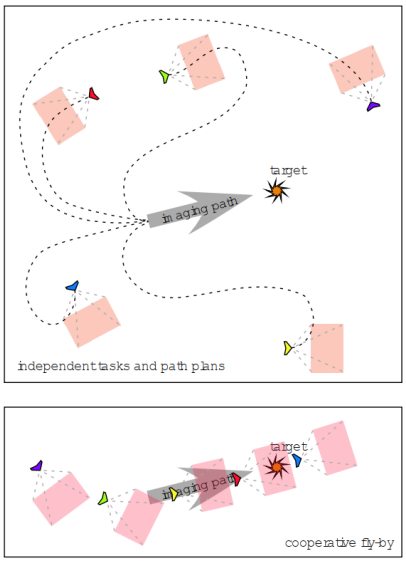


Figure 2: An example of how a swarm of fixed-wing aircraft could be used to continually surveil a target, as described in [4]

This project focused on swarming fixed-wing agents, due to their desirable properties such as long range and high cruise speeds. While these properties make them well-suited to many of the aforementioned applications, they also introduce additional complications. Fixed-wing aircraft have limited angular rates and stall speeds, effectively imposing constraints on the state vector $\mathbf{x}$. As such, any trajectory planner for such an aircraft must be able to handle state constraints as part of its formulation. This motivates the use of Model Predictive Control (MPC), which permits constraints on both the control and state vectors: $\mathbf{u} \in \mathcal{U}, \mathbf{x} \in \mathcal{X}$. Collision avoidance remains a significant concern, however, as the inherently nonlinear dynamics and the inability to hover make fixed-wing aircraft difficult to safely navigate in a constrained environment.

1.2 System Hierarchy

This report intends to demonstrate the feasibility of distributed MPC for fixed-wing trajectory planning. Figure 3 depicts the system hierarchy proposed in this report, of which three blocks have been explored in detail. A distributed Model Predictive Control (MPC) scheme (see Section 3.1) is used to generate realistic fixed-wing trajectories $\mathbf{x}^*$, which are then supplied to a simulated aircraft. To verify that the MPC trajectories are able to be flown by a real fixed-wing aircraft, a nonlinear model of an F-16 has been implemented. Associated PID autopilots have been developed to allow the aircraft's state $\hat{\mathbf{x}}$ to track $\mathbf{x}^*$. As will be discussed in Section 5, these three systems alone are insufficient for real-world implementation. A top-level decision-making system is proposed that is capable of modifying parameters of the distributed MPC scheme to overcome its identified limitations.

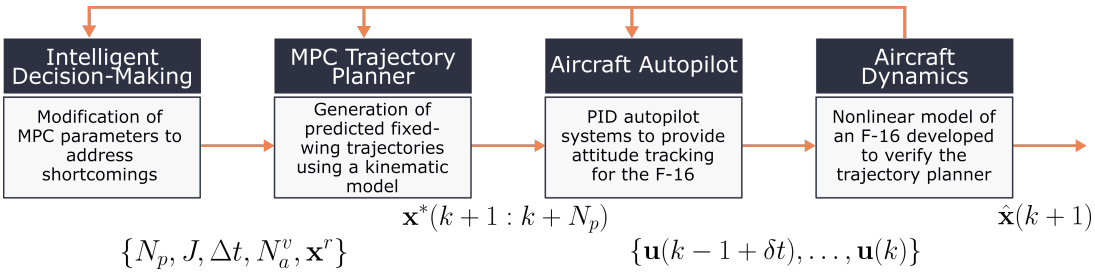


Figure 3: Proposed system hierarchy for real-world implementation

2 Background

2.1 Optimal Control

Optimal control aims to find the sequence of open-loop control inputs $\{\mathbf{u}^*(t)\}$ such that an objective function J is minimised subject to a set of constraints. A general form of such an objective function in continuous time is

$$J = S(\mathbf{x}(T), t) + \int_{t_0}^{T} L\left(\mathbf{x}(t), \mathbf{u}(t), t\right) dt, \tag{1}$$

where $L\left(\mathbf{x}(t), \mathbf{u}(t), t\right)$ is the so-called stage (or running) cost, and $S(\mathbf{x}(T), t)$ is the terminal cost. While the stage cost is the most common method by which desired system behaviour is obtained, the terminal cost is frequently important when enforcing system stability for MPC schemes [9][10].

A common example of optimal control is the infinite-horizon, continuous-time Linear Quadratic Regulator (LQR), which aims to solve the following optimisation problem.

$$\begin{aligned} \{\mathbf{u}^*(t)\} &= \underset{\{\mathbf{u}^*(t)\}}{\arg\min} \frac{1}{2} \int_0^{\infty} \|\mathbf{x}(t)\|_Q^2 + \|\mathbf{u}(t)\|_R^2 dt \\ \text{s.t.} &\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{x}(t_0) = \mathbf{x}^0 \end{cases} \end{aligned} \tag{2}$$

The LQR is one of the few optimal control problems with a known, closed-form solution [11]. A significant reason for this is the linear dynamics, and hence linear equality constraints. For systems with nonlinearities, this problem

rapidly becomes intractable with analytical methods, and numerical schemes become necessary. While some important system characteristics - namely stability - may still be proved analytically using the Lypanuov function method [12], numerical solvers such as IPOPT [13] must be used to produce solutions.

2.2 Model Predictive Control

Model Predictive Control (also known as receding horizon control) is a scheme that iteratively solves a discrete-time optimisation problem over fixed length of time. MPC uses a known model of the system's dynamics, $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$, to predict its future state trajectory over a finite time horizon, dependent on the control sequence $\mathbf{u}^*$. An MPC objective function in the same form as Equation 1, but expressed in discrete time, is

$$J = S(\mathbf{x}(N_p|k)) + \sum_{l=0}^{N_p} L\left(\mathbf{x}(k+l|k) + \mathbf{u}(k+l|k)\right), \tag{3}$$

where the notation $\mathbf{x}(k+l|k)$ represents the predicted state $\mathbf{x}$ at a future discrete time $k+l$ given initial conditions at time k. To limit the computational burden of minimising this objective function, the prediction horizon N_p is limited in size. However, for desirable performance and stability characteristics N_p should be made as large as possible. Computational speed may be improved by increasing the control horizon N_c, which represents which represents the number of time steps before a new optimal control problem must be solved. As the system remains open-loop during this time, N_c should be small relative to N_p. Note that this definition of N_c differs from that used elsewhere [14]. As the problem is iteratively solved over time using the system's state as an input, MPC is a closed-loop scheme. An illustration of the control and prediction horizons can be seen in Figure 4 below.

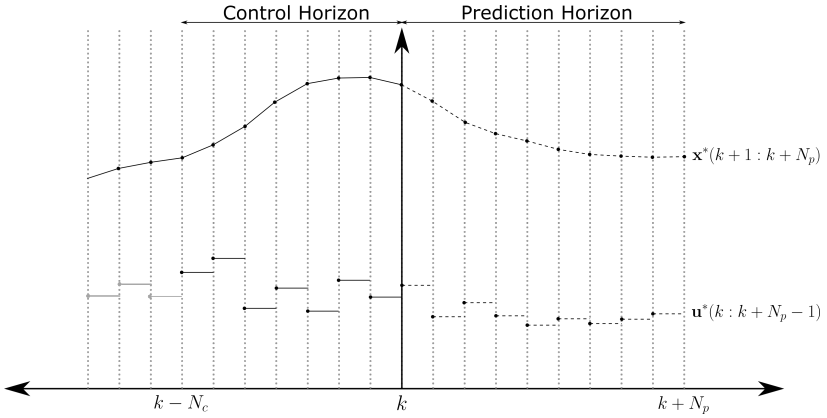


Figure 4: Operation of model predictive control

2.3 Distributed Model Predictive Control

The use of MPC for multi-agent systems, such as the swarming fixed-wing aircraft examined in this report, necessitates the inclusion of neighbour-dependence in the objective function J of each agent. Extending the objective function form used in Equation 3, a distributed MPC objective function J_i for each agent i is (adopting the notation of [15])

$$\begin{aligned} S(\mathbf{x}(N_p|k)) &= S_{ii}(\mathbf{x}_i(N_p|k)) + \sum_{j\in\mathcal{N}_i} S_{ij}(\mathbf{x}_j(N_p|k)) \\ L\left(\mathbf{x}(k+l|k) + \mathbf{u}(k+l|k)\right) &= L_{ii}\left(\mathbf{x}_i(k+l|k) + \mathbf{u}_i(k+l|k)\right) + \sum_{j\in\mathcal{N}_i} L_{ij}\left(\mathbf{x}_j(k+l|k) + \mathbf{u}_j(k+l|k)\right) \end{aligned} \tag{4}$$

where $\cdot_{ii}$ refers to self-dependence of an agent, and $\cdot_{ij}$ is coupling between an agent and its set of neighbours $\mathcal{N}_i$. How these neighbouring agents are defined is implementation-dependent, however a set of simplified assumptions are used for this project. Aircraft within communication range R_{comm} of each other are considered to be neighbours, represented by the set $\mathcal{N}_i$. Virtual agents, which are used to represent detected points on obstacles, are represented by the set $\mathcal{N}_i^v$. To limit the computational burden of this approach, the size of this set is limited such that only the nearest N_a^v virtual agents are considered.

3 Distributed Model Predictive Control Scheme

3.1 MPC Objective Function

In the distributed MPC formulation explored in this report, each agent is assumed to have identical dynamics and control laws; thus forming a homogenous multi-agent system. As such, the optimisation problem $\mathcal{P}_i$ takes the same form for each agent i. By using both the sequence of control inputs $\mathbf{u}_i(k:k+N_p-1)$ and state variables $\mathbf{x}_i(k+1:k+N_p)$ as optimisation variables, this problem uses the collocation method [16]. A notable omission in Equation 5 is that of the terminal cost function, which has been widely found to be an important factor for MPC stability [10][17][9]. This is an intentional omission, as this term was observed to produce oscillatory solutions due to the stage and terminal costs being in conflict for a fixed-aircraft orbiting a target point.

$$\{\mathbf{x}_i(\cdot),\mathbf{u}_i(\cdot)\} = \underset{\{\mathbf{x}_i(\cdot),\mathbf{u}_i(\cdot)\}}{\arg\min} \sum_{l=1}^{N_p} \Bigg[\|\mathbf{x}^r(k+l) - \mathbf{x}_i(k+l|k)\|_Q^2 + \|\mathbf{u}_i(k+l-1|k)\|_R^2 + \sum_{j\in\mathcal{N}_i} \frac{1}{\|\mathbf{x}_i(k+l|k) - \mathbf{x}_j(k+l|k)\|_S^2 + \epsilon} + \sum_{j\in\mathcal{N}_i^v} \frac{1}{\|\mathbf{x}_i(k+l|k) - \mathbf{x}_j^v(k+l|k)\|_{S_v}^2 + \epsilon} \Bigg] \tag{5}$$

$$\forall l = \{0,1,\ldots,N_p-1\},\ \text{s.t.}$$

Optimisation problem $\mathcal{P}_i$ above is solved subject to the following constraints

$$\begin{cases} \mathbf{u}_i(k+l|k) \in \mathcal{U}_i \\ \mathbf{x}_i(k+l+1|k) \in \mathcal{X}_i \\ \mathbf{x}_i(k+l+1|k) = \mathbf{g}\left[\mathbf{x}_i(k+l|k), \mathbf{u}_i(k+l|k)\right] \end{cases} \tag{6}$$

where the admissable control set $\mathcal{U}_i$ is compact, with box constraints

$$\mathcal{U}_i \triangleq \{\mathbf{u}_i \in \mathbb{R}^m | \mathbf{u}_{i,min} \leq \mathbf{u}_i \leq \mathbf{u}_{i,max}\}, \tag{7}$$

and the feasible state set is

$$\mathcal{X}_i \triangleq \{\mathbf{x}_i \in \mathbb{R}^n \mid 1 - \|\mathbf{x}_i(k+l|k) - \mathbf{x}_c\|_{O_j}^2 > 0,\ \forall j \in \mathcal{O},\ l = \{1,2,\ldots,N_p\}\}. \tag{8}$$

The feasible state set $\mathcal{X}_i$ is not necessarily compact nor connected due to the use of inequality constraints $\mathbf{c}_{in} = 1 - \|\mathbf{x}_i(k+l|k) - \mathbf{x}_c\|_{O_j}^2 \leq 0$ which are used to represent ellipsoidal obstacles. Hard equality constraints are

$$\mathbf{c}_{eq}(\mathbf{x}(k),\mathbf{u}(k),\mathbf{x}(k+1)) = \frac{\mathbf{x}(k+1) - \mathbf{x}(k)}{\Delta t} - \mathbf{f}(\mathbf{x}(k),\mathbf{u}(k)), \tag{9}$$

where $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$ are the nonlinear agent dynamics. As is evident from the equation above, an Euler integration scheme was used for its low computational burden and simplicity. Although this scheme has poor numerical precision, this is of little relevance for MPC trajectory planning - particularly considering the choice to reinitialise at the simulated aircraft's location (see Section 5.1). This equation is rearranged to form the state generator function in, as used in Equation 6.

$$\mathbf{g}(\cdot) = \mathbf{x}_i(k+l|k) + \Delta t \cdot \mathbf{f}(\mathbf{x}(k+l|k), \mathbf{u}(k+l|k)) \tag{10}$$

Note that in $\mathcal{P}_i$ it is assumed that all weighting matrices $Q \in \mathbb{R}^{n\times n}, R \in \mathbb{R}^{m\times m}, S \in \mathbb{R}^{n\times n}, S_v \in \mathbb{R}^{n\times n}, O \in \mathbb{R}^{n\times n}$ are symmetric. This property simplifies the calculation of derivatives of the objective function $J(\cdot)$ for use with the solver IPOPT (see Appendix A). Notation used in [15] is also adopted here, with $\mathbf{x}(k+1|k)$ representing the state at instant $k+1$ as predicted by the state at the initial time k. The colon operator, as used in $\mathbf{x}(k:k+N_p)$, represents the the sequence of states $\{\mathbf{x}(k|k), \mathbf{x}(k+1|k), \ldots, \mathbf{x}(k+N_p|k)\}$.

3.2 Fixed-Wing Kinematics

To reduce the computational cost of simulating fixed-wing aircraft trajectories, a kinematic model was used. This model is based on the Dubins path model presented by [18], with constrained control inputs providing the maximum roll rate $\dot{\phi}$ and pitch rate $\dot{\theta}$. It is similar to the bicycle model used for similar application [19] [20]. Its use of 6 state variables and 2 control variables, rather than the 13 state and 4 control required for the F-16 (see Section 4), permit prediction horizons to be doubled for approximately the same computational cost.

$$\begin{aligned}
\dot{x} &= V_T \cos\theta \cos\psi \\
\dot{y} &= V_T \cos\theta \sin\psi \\
\dot{z} &= -V_T \sin\theta \\
\dot{\phi} &= u_{\dot{\phi}} \\
\dot{\theta} &= u_{\dot{\theta}} \\
\dot{\psi} &= \frac{g}{V_T} \tan\phi
\end{aligned} \tag{11}$$

This model assumes that given the constraints on the control inputs $u_{\dot{\theta}}, u_{\dot{\phi}}$ are reasonable for the aircraft, that this kinematic model will generate realistic trajectories for a low-level autopilot. The aircraft's yaw rate $\dot{\psi}$ is constrained to perform coordinated turns, which may be easily tracked by an aircraft autopilot. Variable airspeed has not been taken into account to reduce the computational burden. It is expected that should the airspeed be V_T variable, the solver will reduce the aircraft's speed to its minimum value to permit more time for obstacle avoidance. Although this may be overcome through the inclusion of an aircraft energy term in the objective function $J(\cdot)$, it is outside the scope of this project.

The state vector for the fixed-wing kinematic model is hence

$$\mathbf{x}(k) = \begin{bmatrix} x(k) \\ y(k) \\ z(k) \\ \phi(k) \\ \theta(k) \\ \psi(k) \end{bmatrix}, \tag{12}$$

where x, y, z are expressed in north-east-down coordinates, and the control vector is

$$\mathbf{u}(k) = \begin{bmatrix} u_{\dot{\theta}}(k) \\ u_{\dot{\phi}}(k) \end{bmatrix}. \tag{13}$$

For effective tracking by the F-16 model, these control inputs were constrained to $u_{\dot{\theta}} \in [-5, +5]$ deg/s and $u_{\dot{\phi}} \in [-90, +90]$ deg/s.

3.3 High-Level Algorithm

The figure below describes the general procedure of the distributed MPC algorithm described in this report. It is implicitly assumed that agents are able to either synchronise their communications, although the method by which this is achieved is not considered here. The required communication bandwidth can be dramatically reduced in the following algorithm by instead transmitting control sequences, provided each agent is aware of the appropriate

kinematic models of its neighbours.

Algorithm 1: Distributed MPC Algorithm

initialisation;
$k \leftarrow 1$;
while $k < k_{max}$ **do**
  for *agent* $i = \{1, 2, \ldots, N_a\}$ **do**
    `// MPC trajectory`
    find the set of neighbouring agents $\mathcal{N}_i$;
    find the set of neighbouring virtual agents $\mathcal{N}_i^v$;
    reset the active constraint indices: $Q_i \leftarrow \{\}$;
    activeSetIndex $\leftarrow 0$;
    while *activeSetIndex==0* or collision detected **do**
      activeSetIndex $\leftarrow$ activeSetIndex $+ 1$;
      initialise the control sequence: $\mathbf{u}_i^0 \leftarrow \{\mathbf{u}_i(k : k + N_p - 2), \mathbf{0}\}$;
      initialise the state sequence $\mathbf{x}_i^0(k + 1 : k + N_p)$ using $\mathbf{u}_i^0$;
      if *activeSetIndex==1* **then**
        `// Cold-start solver`
        Solve for $\{\mathbf{x}_i^*(k + 1 : k + N_p), \mathbf{u}_i^*(k : k + N_p - 1)\}$;
      else
        `// Warm-start solver`
        record previous Lagrange multipliers: $\lambda \leftarrow \lambda_{prev}$;
        Solve for $\{\mathbf{x}_i^*(k + 1 : k + N_p), \mathbf{u}_i^*(k : k + N_p - 1)\}$ using λ;
      end
      update Q_i;
      if *activeSetIndex*> 10 **then**
        increase maximum solver iterations: $N_s \leftarrow N_s + 50$;
      end
    end
    reset maximum solver iterations: $N_s \leftarrow 50$;
    `// Aircraft autopilot`
    track the optimal state trajectory $\mathbf{x}_i^*(k + 1 : k + N_c)$;
    record the aircraft's trajectory $\hat{\mathbf{x}}_i(k + 1 : k + N_c)$;
    reset the trajectory planner to the aircraft's current position: $\mathbf{x}_i(k + N_c) \leftarrow \hat{\mathbf{x}}_i(k + N_c)$;
  end
  communicate state sequence to neighbours, $\mathcal{N}_i$;
  $k \leftarrow k + N_c$;
end
post-processing;

3.4 Solver

The constrained optimisation described in Equation 5 was solved using a MATLAB interface to COIN-OR's Interior Point Optimiser (IPOPT) [13]. Table 1 summarises the settings used, however the results obtained are largely insensitive to variations in these values. Gradients, Jacobians and Hessians were supplied to IPOPT, and a detailed description of their structures can be found in Appendix A. The number of iterations specified was intentionally chosen to be low to increase the effectiveness of the active set strategy implemented [21].

Table 1: IPOPT settings

Variable	Value
Linear Solver	MUMPS
μ Strategy	Adaptive
Maximum Solver Iterations	50
Objective Tolerance	10^{-8}
Constraint Violation Tolerance	10^{-4}

4 Nonlinear F-16 Model

To verify that the trajectories produced by the fixed-wing kinematic model described in Section 3.2 can be tracked by a real aircraft, a simulated F-16 model was developed in MATLAB. Using parametric models of aerodynamic data presented in [22], a nonlinear model of the dynamics of an F-16 was developed. Aerodynamic force and moment coefficients were calculated as follows.

$$\begin{aligned}
C_X &= C_X(\alpha, \delta_e) + C_{X_Q}(\alpha)\hat{Q} \\
C_Y &= C_Y(\beta, \delta_a, \delta_r) + C_{Y_P}(\alpha)\hat{P} + C_{Y_R}(\alpha)\hat{R} \\
C_Z &= C_Z(\alpha, \beta, \delta_e) + C_{Z_Q}(\alpha)\hat{Q} \\
C_L &= C_L(\alpha, \beta) + C_{L_P}(\alpha)\hat{P} + C_{L_R}(\alpha)\hat{R} + C_{L_{\delta_a}}(\alpha, \beta)\delta_a + C_{L_{\delta_r}}(\alpha, \beta)\delta_r \\
C_M &= C_M(\alpha, \delta_e) + C_{M_Q}(\alpha)\hat{Q} + C_Z(x_{cg,ref} - x_{cg}) \\
C_N &= C_N(\alpha, \beta) + C_{N_P}(\alpha)\hat{P} + C_{N_R}(\alpha)\hat{R} + C_{N_{\delta_a}}(\alpha, \beta)\delta_a + C_{N_{\delta_r}}(\alpha, \beta)\delta_r - C_Y(x_{cg,ref} - x_{cg})\left(\frac{\bar{c}}{b}\right) \\
\hat{P} &= \frac{Pb}{2V_T} \\
\hat{Q} &= \frac{Q\bar{c}}{2V_T} \\
\hat{R} &= \frac{Rb}{2V_T}
\end{aligned} \tag{14}$$

Note that in the above equations all variables represent standard aerospace quantities, and are independent of the variables defined for the MPC formulation described earlier. To provide a positive stability margin, $x_{cg} = 0.25\bar{c}$ was used. Although this is not necessary for this aircraft model to operate, it was used to increase the region of stability for the relatively simple pitch controller used (see Appendix B). These parameters were dimensionalised using the following equations.

$$\begin{aligned}
X_a &= \bar{q}SC_X \\
Y_a &= \bar{q}SC_Y \\
Z_a &= \bar{q}SC_Z \\
L &= \bar{q}SbC_L \\
M &= \bar{q}S\bar{c}C_M \\
N &= \bar{q}SbC_N
\end{aligned} \tag{15}$$

Table 2 summarises the key physical constants for the F-16 model used. Both gravitational acceleration and aircraft mass have been assumed constant as variations in these quantites will not affect the aircraft's ability to track trajectories produced by the distributed MPC scheme.

Table 2: F-16 physical constants

Parameter	Value
Mass, m	9298.60 kg
Wing span, b	9.14 m
Mean aerodynamic chord, $\bar{c}$	3.45 m
Wing area, S	27.87 m^2
Principal moment of inertia, J_{xx}	12820.61 kg·m^2
Principal moment of inertia, J_{yy}	75673.62 kg·m^2
Principal moment of inertia, J_{zz}	85552.11 kg·m^2
Product of inertia, J_{xz}	1331.41 kg·m^2
Reference CG position, $x_{cg,ref}$	$0.35\bar{c}$
Maximum aileron deflection, δ_a	$\pm$25.0 deg
Maximum elevator deflection, δ_e	$\pm$21.5 deg
Maximum rudder deflection, δ_r	$\pm$30.0 deg
Maximum thrust lever, δ_t	$\pm$1

To model the F-16's engine and the associated spool-up time, the approximate model given in [23] was used. This introduces an additional state variable to track the engine's power level P_e, and models the spool-up time using a first-order lag. As this model is algorithmic, it is represented by $f(V_T, h, \delta_t)$ in Equation 16 below.

The equations of motion of the 13 state variables can be seen below. It can be readily seen that the 6, relatively simple, equations of motion proposed in Section 3.2 are preferable for use in a nonlinear constrained optimiser.

$$
\begin{aligned}
\dot{U} &= RV - QW - g_D \sin\theta + (X_A + X_T)/m \\
\dot{V} &= -RU + PW + g_D \sin\phi\cos\theta + Y_A/m \\
\dot{W} &= QU - PV + g_D \cos\phi\cos\theta + Z_A/m \\
\dot{P} &= \frac{1}{J_{xx}}\left((J_{yy} - J_{zz})QR + J_{xz}PQ + L\right) \\
\dot{Q} &= \frac{1}{J_{yy}}\left((J_{zz} - J_{xx})RP + J_{xz}(R^2 - P^2) + M\right) \\
\dot{R} &= \frac{1}{J_{zz}}\left((J_{xx} - J_{yy})PQ - J_{xz}QR + N\right) \\
\dot{x} &= U\cos\theta\cos\psi + V(-\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi) + W(\sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi) \\
\dot{y} &= U\cos\theta\sin\psi + V(\cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi) + W(-\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi) \\
\dot{h} &= U\sin\theta - V\sin\phi\cos\theta - W\cos\phi\cos\theta \\
\dot{\phi} &= P + \tan\theta(Q\sin\phi + R\cos\phi) \\
\dot{\theta} &= Q\cos\phi - R\sin\phi \\
\dot{\psi} &= \frac{1}{\cos\theta}(Q\sin\phi + R\cos\phi) \\
\dot{P}_e &= f(V_T, h, \delta_t)
\end{aligned}
\tag{16}
$$

5 Results and Discussion

5.1 Realistic Fixed-Wing Trajectories

To verify the claim that trajectories generated by the distributed MPC scheme described in Section 3 are realistic for fixed-wing aircraft, the aforementioned F-16 model was directed to track these trajectories. Commanded attitude angles ϕ, θ, ψ and altitude h were provided to PID attitude autopilots (see Appendix B) which produced trajectories such as that depicted in Figure 5. These autopilots were simulated to operate at 100 Hz using a Runge-Kutta 4th-order integration scheme, however results have been plotted at the same frequency as the trajectory planner (1 Hz) for visualisation purposes.

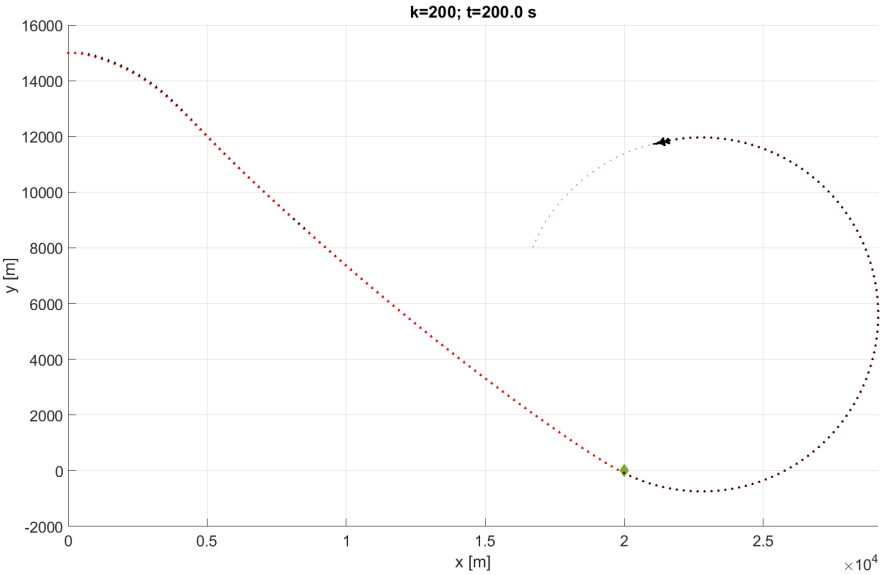


Figure 5: Top-down view of error between MPC-generated trajectories and F-16 flight path

Attitude and altitude tracking performance can be seen in Figure 6, where F-16 data sampled at 1 Hz is illustrated in red. Error in yaw can be seen to be the most significant, however this is likely due to the minimisation of sideslip angle β in the yaw autopilot; an inclusion intended to reduce the presence of nonlinear aerodynamic effects on the aircraft. As discussed in Section 3.2, control constraints limiting the commanded bank and pitch rates, $\dot{\phi}, \dot{\theta}$, provided an effective means of ensuring that trajectories were flyable for the aircraft. It was found during testing that pitch rates in excess of ± 5 deg/s could result in instability during aggressive roll and yaw manoeuvres. While this is not surprising considering the simplicity of the autopilots used for such an unstable aircraft, it highlights the flexibility of this scheme to adapt to both limitations in the aircraft and its low-level control systems. Roll rate limits were found to be able to be set significantly higher, at a value of ± 90 deg/s, leading to the aggressive turn rates illustrated in Figure 6.

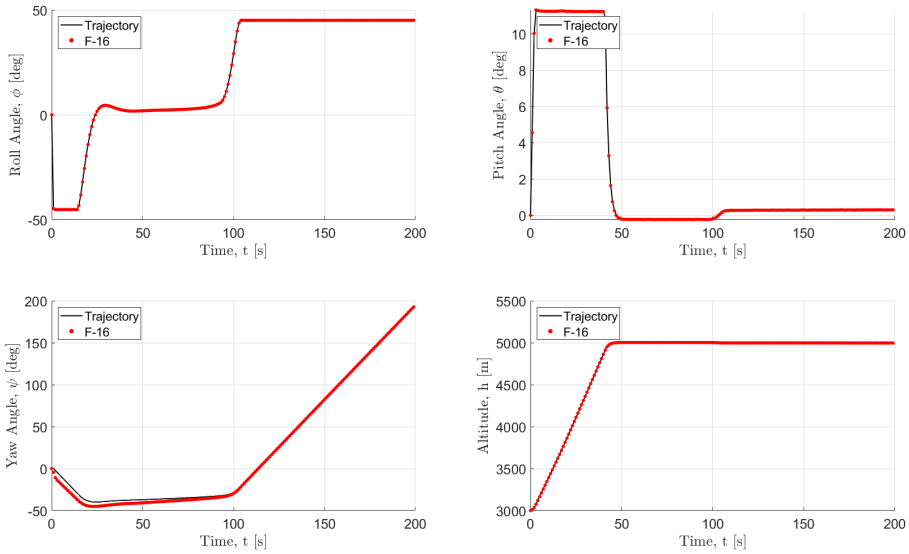
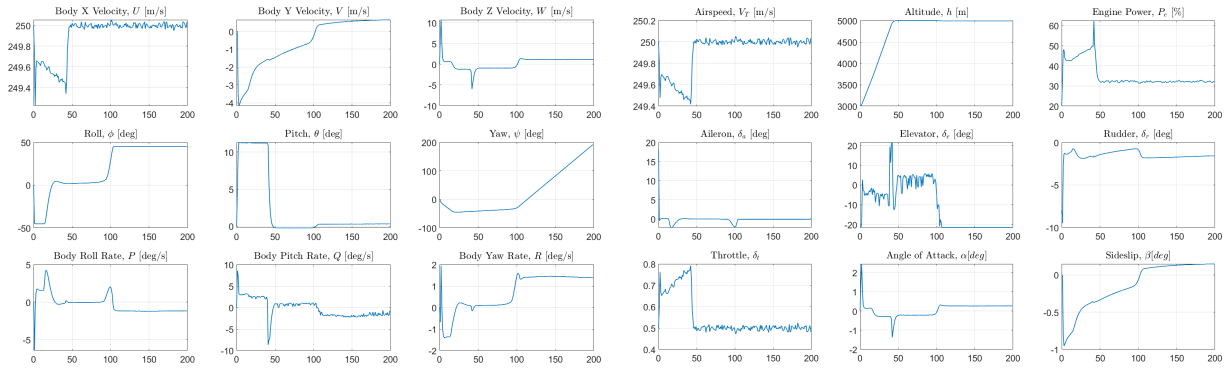


Figure 6: F-16 tracking error time histories for Figure 5

To prevent the development of steady-state tracking errors, the MPC trajectory planner was reinitialised every N_c steps at the aircraft's current position. In addition to reducing the error between the planned and aircraft trajectories, this also allowed predicted trajectories to more accurately predicted the aircraft's future motion. A limitation of this approach is discussed in Section 5.2, where small tracking errors could result in the MPC algorithm being initialised at an infeasible point - resulting in an unexpected failure of the solver.

Figure 7 has been included for completeness, and demonstrates that the F-16's 13 state variables, 4 control variables, and 2 key derived quantities (α, β) were both simulated and realistic. Control inputs can be seen to be aggressive, however they remain limited to physical constraints on the F-16's control surfaces.



(a) F-16 state variables

(b) F-16 state, control and derived variables

Figure 7: F-16 state time histories for Figure 5

5.2 Collision Avoidance Using State Constraints

One of the two methods of collision avoidance used was the inclusion of state constraints, $\mathbf{x}_i \in \mathcal{X}_i$. As this method requires the solver to produce collision-free (feasible) trajectories, obstacles of this kind are referred to as 'hard'. Hard obstacles were modelled as ellipsoids, such that the inequality condition imposed was

$$c_{in,i}(\mathbf{x}_i(k)) = 1 - \|\mathbf{x}_i(k) - \mathbf{x}^c\|^2_{O_j} \leq 0. \tag{17}$$

Although the matrix $O_j \in \mathbb{R}^{n \times n}$ could in general by non-symmetric, for ease of differentiation in forming constraint Jacobians and Hessians (see Appendix A) it has been assumed that $O_j^T = O_j \; \forall \; j \in \mathcal{O}$, where $\mathcal{O}$ is the set of all hard obstacles.

Due to the significant computational burden imposed by the inclusion of state constraints, the active set method proposed in [21] has been adopted. This method iteratively increases the number of active inequality constraints considered by only considering predicted collisions at the boundaries of obstacles, thereby reducing the total computation time. For the simulation depicted in Figure 8, for example, computation time without the active set method is 173.6 s, whereas with it activated this falls to 113.6 s: a reduction of 34.6%. Considering the relatively simple implementation of this active set strategy (see Algorithm 1), it appears worthwhile implementing for any MPC scheme using state constraints for obstacle avoidance.

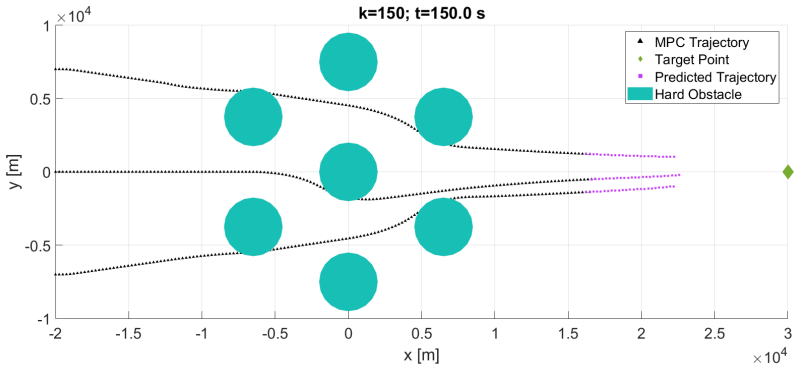


Figure 8: Collision-free trajectories generated using state constraints

To illustrate the capabilities of the MPC scheme, Figure 8 was created. It demonstrates how state contraints allow aircraft to begin adjusting well in advance of the obstacle, in contrast to the finite detection and communication ranges discussed in Section 5.3. An important feature of these trajectories is their tendency to lie tangent to the surface of the obstacle - as this is the path that permits the aircraft minimum control inputs. While this is the optimal method of minimising the objective function, it is problematic when a simulated aircraft is included. As the aircraft autopilots implemented for the F-16s simulated in this project often contain small steady-state tracking errors, it is possible for the aircraft to enter the region occupied by the hard obstacle. If the MPC scheme is then re-initialised from the aircraft's position within such an obstacle, the solver will fail due to an infeasible initial condition. While this problem may be ameliorated by placing virtual agents at the centre of each hard obstacle, whose reciprocal objective functions precipitate trajectories with greater distances from the boundaries of obstacles, this solution was not found to be reliable. This highlights a significant issue with this type of obstacle which, when combined with its high computational cost, suggests that soft obstacles are preferable for fast and reliable operation of this system.

5.3 Collision Avoidance Using the Objective Function

Through the inclusion of reciprocal functions of the form

$$\frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|_A^2 + \epsilon}, \tag{18}$$

for each agent i and neighbouring virtual agent $j \in \mathcal{N}_i^v \cup \mathcal{N}_i$, collision avoidance was incorporated in the MPC objective function. The weighting matrix A was selected such that a sharp rise in the objective function would occur if the aircraft approach within 2 km of any obstacle or 1 km of any aircraft. Using $\epsilon = 10^{-16}$, the maximum value of this function reached working precision of the simulation: an effectively infinite value. While this approach does not necessarily guarantee collision-free trajectories, in practice it has been found to work well both in this project (see Figure 9) and in similar work [24][25].

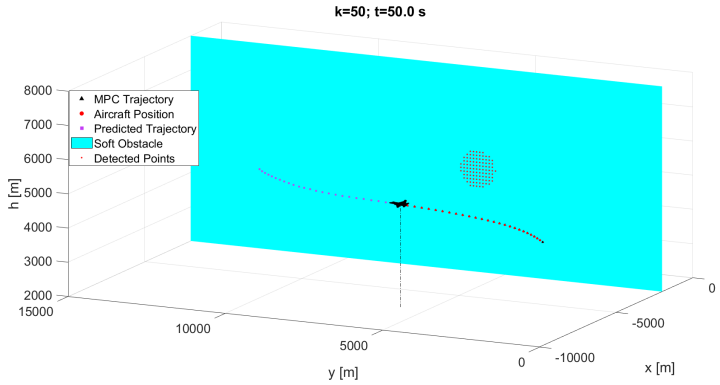


Figure 9: Closest $|\mathcal{N}_i^v|$ points used for soft obstacle avoidance, using the algorithm described in Appendix C

To simulate the detection of obstacles in an aircraft's environment, a ray-tracing algorithm was implemented (see Appendix C). This algorithm ensures that only the nearest $|\mathcal{N}_i^v|$ virtual agents that are visible from the aircraft's position are included in the objective function. Limiting the number of virtual agents served to increase the relative effect of each individual point. As $|\mathcal{N}_i^v|$ is increased, the weighting matrix S_v must be commensurately decreased to ensure a balance between the terms in the objective function. In simulations where this balance was not enforced the generated trajectories no longer approached the target point; effectively leading to an unstable solution.

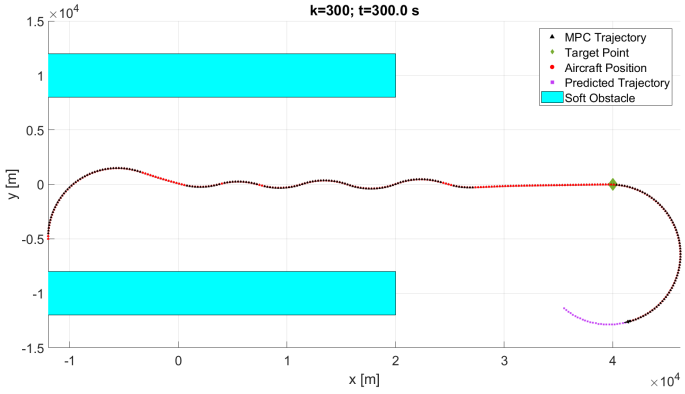


Figure 10: Oscillations induced by the limited number of points $|\mathcal{N}^v|$ used for soft obstacle avoidance

A notable advantage of this approach to collision avoidance is the decreased computational cost, as an unconstrained state space allows for faster solutions and guaranteed feasibility; a desirable property for any distributed MPC scheme [15]. A lack of state constraints also preserves the state space's properties of being convex and connected, which are intuitive requirements for stabilisability [12][17].

A significant limitation of this approach is its inability to handle local minima in the objective function. As can be seen in Figure 11, a barrier in the objective function formed by the obstacles results in the aircraft becoming trapped. While this could be overcome by extending the aircraft's prediction horizon N_p such that it could predict a path around the obstacle, this would require a dramatic increase in computational work. Furthermore, selecting N_p to be applicable for any set of obstacles would remain impossible, as increasingly large local minima could feasibly be formed.

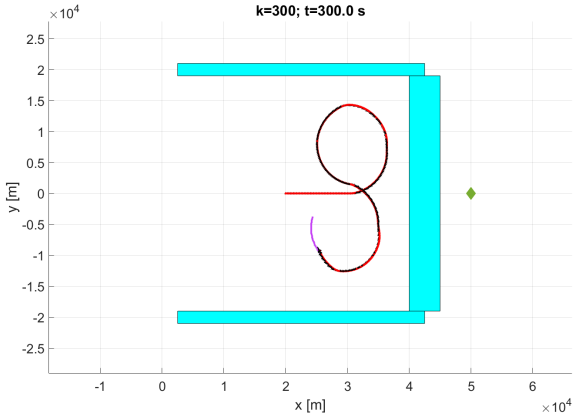


Figure 11: A local minimum in the objective function formed by the use of an objective-function method of collision avoidance

This shortcoming motivates the system hierarchy proposed in Section 1.2, in which a decision-making layer capable of adjusting parameters such as N_p and $\mathbf{x}^r$ is proposed. Should a local minimum such as the one depicted above be detected, the prediction horizon N_p could be adaptively increased, or the target point $\mathbf{x}^r$ translated, to direct the aircraft away from the local minimum. Continual adjustment of $\mathbf{x}^r$ could potentially be used in this way to allow aircraft to avoid dead-ends and navigate complex environments.

As can be seen in Equation 3, collision avoidance between aircraft is implemented in the same manner as the soft obstacle avoidance discussed above. Aircraft use predicted trajectories communicated by their neighbours (within R_{comm}) to plan their own, collision-free trajectories over the prediction horizon. This is illustrated in Figure 12, in which horizontal and vertical separation is increased between the aircraft, whilst both continue moving toward the target point. A limitation of this approach is the significant quantity of information that must be rapidly communicated (and synchronised in time). While the information quantity may be reduced by communicating control sequences $\mathbf{u}^*(k:k{+}N_p{-}1)$ instead, and relying on neighbours' internal dynamic models of each other to estimate $\mathbf{x}^*(k{+}1:k{+}N_p)$ [15], this is still a significant quantity of data. Each aircraft must receive mN_p control values for each of its neighbours before it can calculate its own predicted trajectory. It is thus desirable for any real-world implementation of this algorithm to use relatively large Δt and N_c to provide adequate time for this information to be communicated.

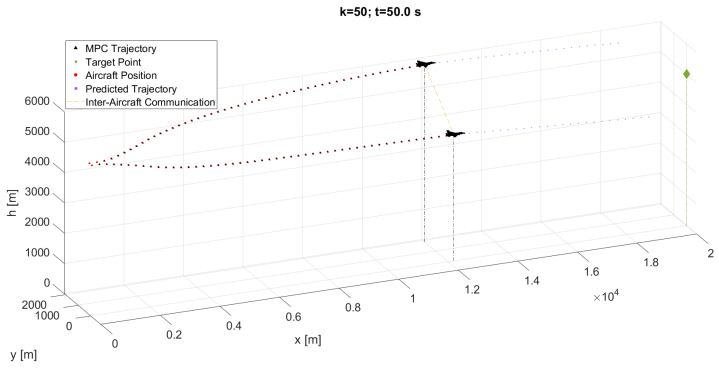


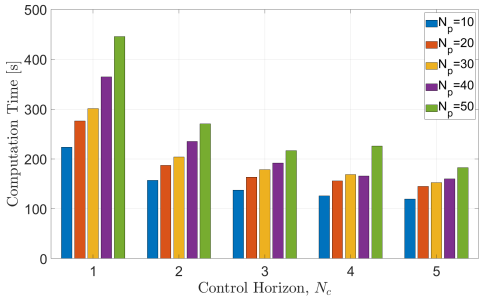
Figure 12: Aircraft increasing their standoff distance using objective function repulsion

5.4 Parameter Selection

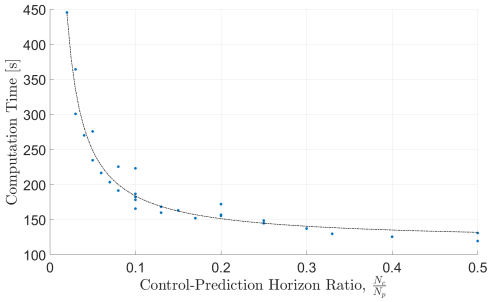
Selecting the value of parameters used in this formulation involves a series of trade-offs, which are illustrated in the figures that follow. The key parameters used are; the prediction horizon N_p, which controls the number of predicted future states; the control horizon N_c, which determines how frequently the predicted trajectory is updated; the time

step Δt, determining the time between state predictions; the communication and prediction ranges R_{comm}, R_{pred}, used to simulate finite communication and sensor distances; and the number of virtual agents N_a^v, which sets the maximum number of detected points on an obstacle to be used in the objective function.

Under ideal conditions the prediction horizon N_p would be set to an effectively infinite value, however this is not possible given computational constraints. Although a predicted trajectory generated using a large value of N_p is preferable, if it is calculated long after the instant at which it was to be applied, then the solution is no longer applicable. As depicted in Figure 13a, which was generated from a 1000 s simulation of a single aircraft flying a collision-free environment, computational time increases rapidly with rising N_p. This can be somewhat improved by increasing the control horizon N_c, however its effect rapidly diminishes. These diminishing returns are highlighted in Figure 13b, in which the increasing fraction N_c/N_p confers limited computational advantage. Furthermore, as sufficiently large control horizons have the potential to destabilise the system it is preferable to use ratios of $N_c/N_p \approx 0.1$. For much of the results presented in this report, values of $N_p = 25, N_c = 3$ were used to produce satisfactory results.



(a) Diminishing returns for increasing N_c, N_p



(b) Inverse correlation between computation time and the ratio N_c/N_p

Figure 13: Trade Studies for Control and Prediction Horizons N_c, N_p

While increasing N_p resulted in increased computational burden, this increase was observed to be small relative to the effects of inappropriate time step Δt. As shown in the log-log plot, Figure 14, changes in time step could result in dramatic changes, particularly for small Δt. For instance, decreasing the time step from $\Delta t = 0.05$ s to $\Delta t = 0.01$ s increased the computation time by a factor of 20. Although small Δt produced lower roundoff errors from the finite difference approximation in Equation 9, by resetting the trajectory planner to the aircraft's state every N_c steps lthe growth of these numerical errors was limited. Based on the data shown in Figure 14, a time step of $\Delta t = 1.0$ s was used to generate the results in this report. The increased computation time for $\Delta t > 1$ s is predominantly due to the increased difficulty in satisfying equality constraints for the solver. This effect is exacerbated by the aircraft autopilots, which must provide aggressive control inputs to track the often oscillatory solutions for large time steps.

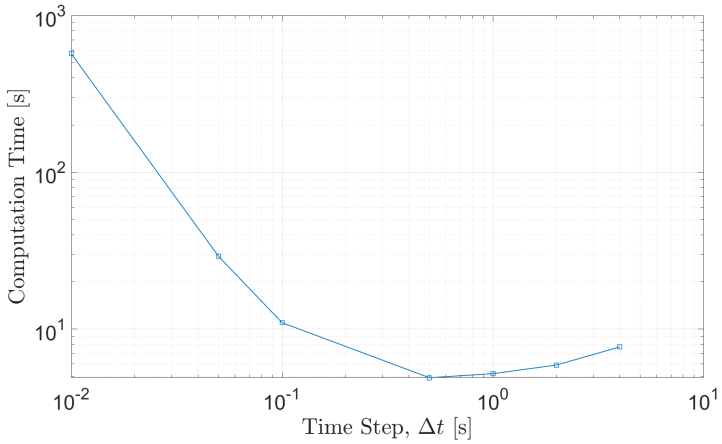
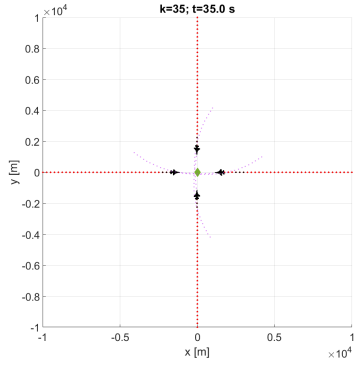


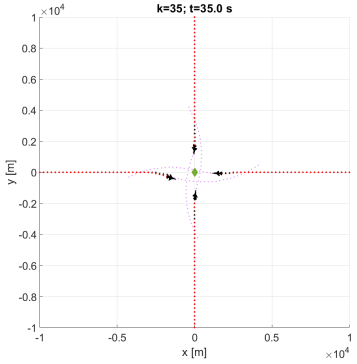
Figure 14: Time step selection for obstacle-free reference tracking

The data in Figure 14 above were formed from the flight path of a single aircraft in a obstacle-free environment. For each Δt, a fixed prediction horizon of $T_p = N_p \Delta t = 20$ s and control horizon of $T_c = N_c \Delta t = 4$ s were used to focus on the influence of Δt alone.

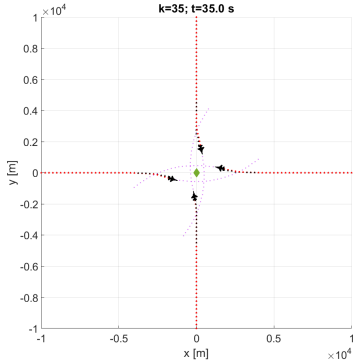
The communication range R_{comm} and detection range R_{det} have a similar, intuitive effect: they increase the aircraft's ability to maintain a safe distance from obstacles. In Figure 15 a sequence of different communication ranges is shown. When aircraft are capable of communicating their predicted trajectories well in advance of entering a region of potential collision, inter-aircraft distances remain safe.



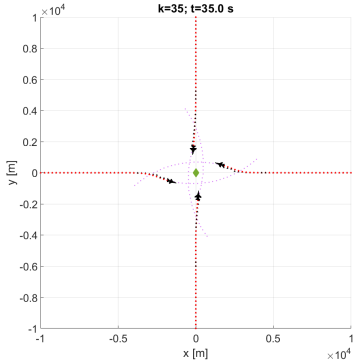
(a) $R_{comm} = 2.5$ km



(b) $R_{comm} = 5.0$ km



(c) $R_{comm} = 7.5$ km



(d) $R_{comm} = 10.0$ km

Figure 15: Effect of maximum communication range on aircraft avoidance

This result can also be seen in Figure 16, in which $R_{comm} > 6$ km produces trajectories with distances of almost 900 m between aircraft. Considering the wingspan of the F-16 is 9.1 m, this distance provides a significant margin for error. While the specific values obtained are only relevant to the scenario depicted in Figure 15, these data illustrate the broad benefits of an increased communication (or, equivalently) or detection horizon.

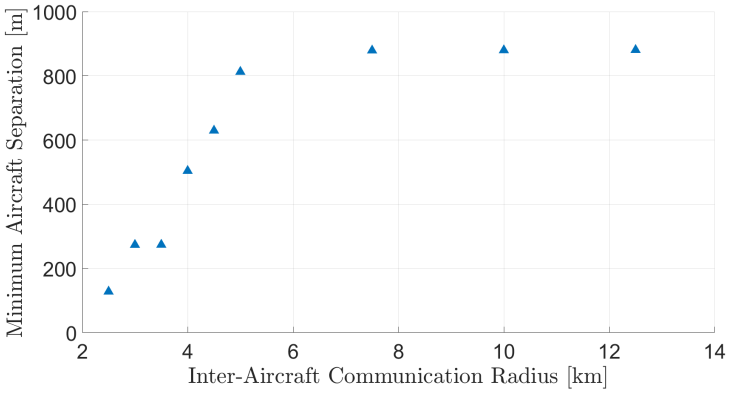


Figure 16: Minimum distance between aircraft due to changing communication ranges

5.5 Large-Scale Swarming

Combining the features described in the preceding sections, and omitting hard constraints due to the issues described in Section 5.2, the simulation depicted in Figure 17 was produced. During this simulation, the point of closest approach between any two aircraft was 676 m; a safe distance considering the F-16's 9.1 m wingspan. In this simulation, two groups of 16 aircraft move along perpendicular paths toward separate targets. By avoiding the first set of obstacles they are forced to move closer to neighbouring aircraft, demonstrating this schemes ability to balance obstacle and aircratf avoidance. The two groups then cross after approximately 200 s, adjusting their altitudes and headings to ensure no collisions occur despite all aircraft flying at $M = 0.7$ ($V_T = 250$ m/s). Both groups then avoid a final obstacle, before orbiting their respective target points until $t = 850$ s. The success of this simulation demonstrates the capacity of this scheme to be applied to large-scale swarms of fixed-wing aircraft.

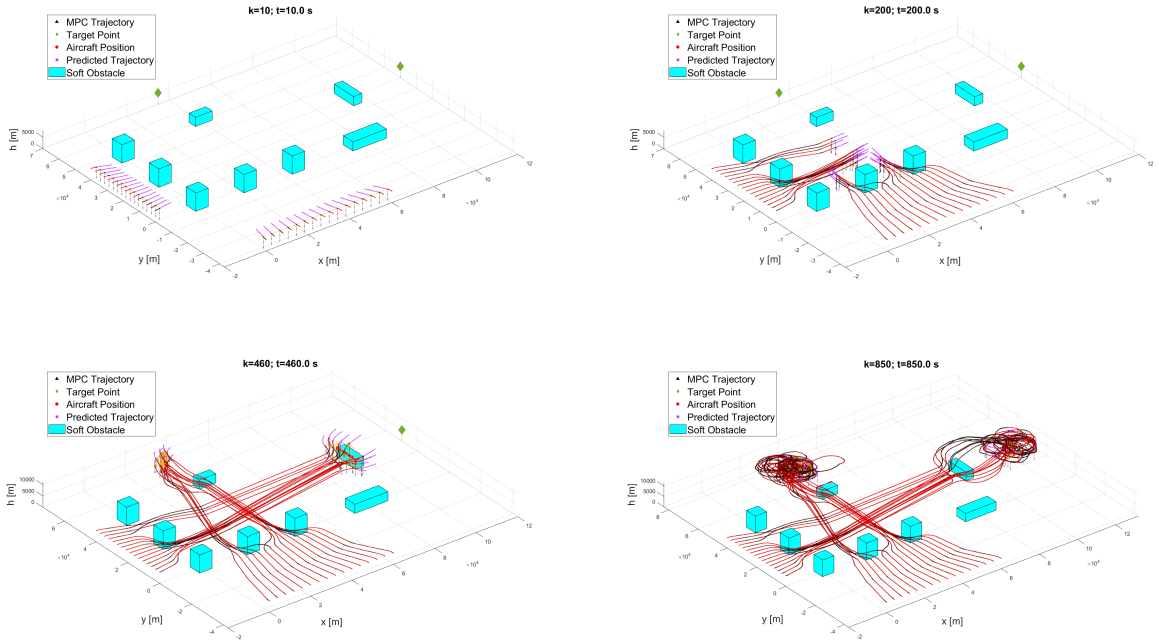


Figure 17: Collision-free swarming of 32 F-16 aircraft

6 Conclusion

This project has demonstrated the feasibility of a distributed MPC scheme for fixed-wing trajectory planning of swarming aircraft. The capability of MPC to handle control and state constraints makes it well suited to use with fixed-wing aircraft due to their flight envelopes and limited manoeuvrability. These constraints were incorporated into a kinematic model approximating fixed-wing dynamics by limiting the maximum pitch and roll rates, $\dot{\theta}, \dot{\phi}$. To verify that this kinematic model produced realistic trajectories, a nonlinear F-16 model was developed with associated PID autopilots. As shown in Section 5.1, the MPC trajectories generated were able to be accurately tracked by the F-16; justifying the use of the kinematic approximation.

Two types of collision avoidance were examined; the use of state constraints for 'hard' obstacles, and objective function terms for 'soft' obstacles and inter-agent avoidance. In addition to the known issue of high computational cost, the possibility of the MPC scheme being initialised within an infeasible region made the use of hard obstacle problematic. As such, objective function based obstacles were concluded to be preferable, and were shown to work well even with large numbers of aircraft in Section 5.5. The significant volume of data required to be communicated between aircraft with this approach was identified as an issue for real-world applications. Transmission of control sequences $\mathbf{u}^*(k : k + N_p - 1)$, in addition to increased control horizons N_c and time steps Δt, were suggested as methods to mitigate this limitation. Trade studies were performed to identify $\Delta t \approx 1$ s and $N_c/N_p \approx 0.1$ as appropriate parameter choices for the F-16 with this MPC scheme. Large scale swarming was simulated with these parameters to demonstrate how the proposed scheme can be dramatically increased in scale while remaining collision-free.

7 Acknowledgements

The author would like to thank his supervisor, Dr. Hoam Chung, for his invaluable guidance and support throughout this project.

References

[1] M. Schaefer. *The Murmurations of Starlings.*

[2] Daniel J Pack, Pedro DeLima, Gregory J Toussaint, et al. "Cooperative control of UAVs for localization of intermittently emitting mobile targets". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39.4 (2009), pp. 959–970.

[3] David W Casbeer, Derek B Kingston, Randal W Beard, et al. "Cooperative forest fire surveillance using a team of small unmanned air vehicles". In: *International Journal of Systems Science* 37.6 (2006), pp. 351–360.

[4] Randal W Beard, Timothy W McLain, Derek B Nelson, et al. "Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs". In: *Proceedings of the IEEE* 94.7 (2006), pp. 1306–1324.

[5] Jinlu Han, Yaojin Xu, Long Di, et al. "Low-cost multi-UAV technologies for contour mapping of nuclear radiation field". In: *Journal of Intelligent & Robotic Systems* 70.1-4 (2013), pp. 401–410.

[6] Stephanie Petillo, Henrik Schmidt, and Arjuna Balasuriya. "Constructing a distributed AUV network for underwater plume-tracking operations". In: *International Journal of Distributed Sensor Networks* 8.1 (2011), p. 191235.

[7] Achudhan Sivakumar and Colin Keng-Yan Tan. "UAV swarm coordination using cooperative control for establishing a wireless communications backbone". In: *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 3-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems. 2010, pp. 1157–1164.

[8] Roy S Smith and Fred Y Hadaegh. "Control of deep-space formation-flying spacecraft; relative sensing and switched information". In: *Journal of Guidance, Control, and Dynamics* 28.1 (2005), pp. 106–114.

[9] David Mayne. "An apologia for stabilising terminal conditions in model predictive control". In: *International Journal of Control* 86.11 (2013), pp. 2090–2095.

[10] Fernando ACC Fontes. "A general framework to design stabilizing nonlinear model predictive controllers". In: *Systems & Control Letters* 42.2 (2001), pp. 127–143.

[11] William L. Brogan. *Modern control theory*. 3rd ed. Englewood Cliffs, N.J.: Englewood Cliffs, N.J. : Prentice Hall, 1991.

[12] Frank Allgöwer and Alex Zheng. *Nonlinear model predictive control*. Vol. 26. Birkhäuser, 2012.

[13] Andreas Wächter and Lorenz T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57. ISSN: 0025-5610. DOI: 10.1007/s10107-004-0559-y.

[14] Carlos E Garcia, David M Prett, and Manfred Morari. "Model predictive control: theory and practice—a survey". In: *Automatica* 25.3 (1989), pp. 335–348.

[15] José M Maestre, Rudy R Negenborn, et al. *Distributed model predictive control made easy*. Vol. 69. Springer, 2014.

[16] Oskar Von Stryk and Roland Bulirsch. "Direct and indirect methods for trajectory optimization". In: *Annals of operations research* 37.1 (1992), pp. 357–373.

[17] David Q Mayne, James B Rawlings, Christopher V Rao, et al. "Constrained model predictive control: Stability and optimality". In: *Automatica* 36.6 (2000), pp. 789–814.

[18] Mark Owen, Randal W Beard, and Timothy W McLain. "Implementing dubins airplane paths on fixed-wing uavs". In: *Handbook of Unmanned Aerial Vehicles* (2015), pp. 1677–1701.

[19] Jessica Pannequin, Alexandre Bayen, Ian Mitchell, et al. "Multiple aircraft deconflicted path planning with weather avoidance constraints". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2007, p. 6588.

[20] Yeonsik Kang and J Karl Hedrick. "Linear tracking for a fixed-wing UAV using nonlinear model predictive control". In: *IEEE Transactions on Control Systems Technology* 17.5 (2009), pp. 1202–1210.

[21] Hoam Chung, Elijah Polak, and Shankar Sastry. "On the use of outer approximations as an external active set strategy". In: *Journal of optimization theory and applications* 146.1 (2010), pp. 51–75.

[22] Eugene A Morelli. "Global nonlinear parametric modelling with application to F-16 aerodynamics". In: *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*. Vol. 2. IEEE. 1998, pp. 997–1001.

[23] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[24] David Shim, Hoam Chung, Hyoun Jin Kim, et al. "Autonomous exploration in unknown urban environments for unmanned aerial vehicles". In: *AIAA Guidance, Navigation, and Control Conference and Exhibit*. 2005, p. 6478.

[25] DS US, H Kim, and Shankar Sastry. "Decentralized reflective model predictive control of multiple flying robots in dynamic environment". In: *Proc. 42nd IEEE Conf. Decis. Control*. 2003.

A Derivatives Provided to IPOPT

To permit the efficient solution of contrained nonlinear problems as described in Section 3.1, IPOPT required a number of derivatives to be supplied. These include the gradient and Hessian of the objective function, Jacobian and Hessian of the constraints, and the sparse matrix structure of these matrices. This appendix provides the gradients, Jacobians, and Hessians used in this project.

A.1 Optimisation Variable Structure

To form the derivative vectors and matrices described in this appendix, the structure of the optimisation vector must be known and remain consistent. The structure chosen was

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}_i^*(k+1|k) \\ \mathbf{x}_i^*(k+2|k) \\ \vdots \\ \mathbf{x}_i^*(k+N_p|k) \\ \mathbf{u}_i^*(k|k) \\ \mathbf{u}_i^*(k+1|k) \\ \vdots \\ \mathbf{u}_i^*(k+N_p-1|k) \end{bmatrix}, \tag{19}$$

which vertically concatenates the optimal state and control sequences for each agent i. In the equations that follow, the superscript $\cdot^*$ and the syntax $(\cdot|k)$ are omitted to improve readability.

A.2 Objective Function

The gradient of the obective function, $\nabla J \in \mathbb{R}^{(n+m)N_p \times 1}$, (see Equation 5) is formed in the following structure

$$\nabla J = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}(k+1)} \\ \frac{\partial}{\partial \mathbf{u}(k+1)} \\ \frac{\partial}{\partial \mathbf{x}(k+2)} \\ \frac{\partial}{\partial \mathbf{u}(k+2)} \\ \vdots \\ \frac{\partial}{\partial \mathbf{x}(k+N_p)} \\ \frac{\partial}{\partial \mathbf{u}(k+N_p)} \end{bmatrix} J_i. \tag{20}$$

These elements have the following general pattern for $l \in [0, N_p - 1]$.

$$\begin{aligned} \frac{\partial}{\partial \mathbf{x}} J_i(k+l) &= -2Q\left(\mathbf{x}^r(k+l) - \mathbf{x}_i(k+l|k)\right) + \sum_{j \in \mathcal{N}_i \cup \mathcal{N}_i^v} \frac{-2S\left(\mathbf{x}_i(k+l|k) - \mathbf{x}_j(k+l|k)\right)}{\left(\|\mathbf{x}_i(k+l|k) - \mathbf{x}_j(k+l|k)\|_S^2 + \epsilon\right)^2} \\ \frac{\partial}{\partial \mathbf{u}} J_i(k+l) &= 2R\mathbf{u}_i(k+l|k) \end{aligned} \tag{21}$$

Due to the lack of a terminal cost in the objective function, the final $(n+m)$ terms are zero. Note that since the weighting matrices, such as Q, only have nonzero weights for the position elements $\{x, y, z\}$, this vector is sparsely populated.

The Hessian matrix of the objective function, $\mathbf{H}(J_i) \in \mathbb{R}^{(n+m)N_p \times (n+m)N_p}$, can be obtained similarly to the gradient

above, with the general form as follows:

$$\mathbf{H}(J_i) = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}(k+1)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+1)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{x}(k+1)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+1)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \\ \frac{\partial}{\partial \mathbf{x}(k+2)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+2)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{x}(k+2)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+2)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{x}(k+N_p)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \\ \frac{\partial}{\partial \mathbf{u}(k+1)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+1)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{u}(k+1)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+1)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \\ \frac{\partial}{\partial \mathbf{u}(k+2)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+2)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{u}(k+2)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+2)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{u}(k+N_p)}\frac{\partial}{\partial \mathbf{x}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\frac{\partial}{\partial \mathbf{x}(k+N_p)} & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\frac{\partial}{\partial \mathbf{u}(k+1)} & \cdots & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\frac{\partial}{\partial \mathbf{u}(k+N_p)} \end{bmatrix} J_i \tag{22}$$

Representing this in block form

$$\mathbf{H}(J_i) = \begin{bmatrix} \mathbf{H}_{xx}(J_i) & \mathbf{H}_{xu}(J_i) \\ \mathbf{H}_{ux}(J_i) & \mathbf{H}_{uu}(J_i) \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{xx}(J_i) & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_{uu}(J_i) \end{bmatrix}, \tag{23}$$

where $\mathbf{H}_{xu}(J_i) = \mathbf{H}_{ux}(J_i)^T = \mathbf{0}$ due to the lack of coupling between control inputs and states in the objective function.

The specific forms of these submatrices are

$$\begin{aligned} \mathbf{H}_{xx}(J_i) &= \begin{bmatrix} \alpha(k+1) & 0 & 0 & \dots & 0 & 0 \\ 0 & \alpha(k+2) & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \alpha(k+N_p-1) & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} \\ \mathbf{H}_{uu}(J_i) &= 2R \otimes I_{N_p} \end{aligned} \tag{24}$$

where $\alpha(k) = 2Q + \sum_{j \in \mathcal{N}_i \cup \mathcal{N}_i^v} \frac{2S\left(4\|\mathbf{x}_i(k)-\mathbf{x}_j(k)\|_2^2 S - 2\left(\|\mathbf{x}_i(k)-\mathbf{x}_j(k)\|_S^2+\epsilon\right)\right)}{\left(\|\mathbf{x}_i(k)-\mathbf{x}_j(k)\|_S^2+\epsilon\right)^3}$.

A.3 Constraint Equations

As described earlier in this report, two types of constraints were used: equality constraints to impose aircraft dynamics, and inequality constraints to impose collision avoidance for 'hard' obstacles. The equality constraints function can be expressed as

$$\mathbf{c}_{eq,i}(\mathbf{x}_i(k+l), \mathbf{x}_i(k+l-1), \mathbf{u}_i(k+l-1)) = \frac{\mathbf{x}_i(k+l) - \mathbf{x}_i(k+l-1)}{\Delta t} - \mathbf{f}_i\left[\mathbf{x}_i(k+l-1), \mathbf{u}_i(k+l-1)\right], \tag{25}$$

where $l = \{0, 1, \dots, N_p\}$.

For each 'hard' obstacle $j \in \{1, 2, \dots, N_o\}$, the inequality constraint function is

$$c_{in,i}(\mathbf{x}_i(k)) = 1 - \|\mathbf{x}_i(k) - \mathbf{x}^c\|_{O_j}^2 \leq 0. \tag{26}$$

Using the two equations above, the constraints Jacobian can be formed. Using IPOPT's format of constraint equations running along the rows, and optimisation variables running along the columns, a block matrix representation can be formed.

$$\mathbf{J}(\mathbf{C}_i) = \begin{bmatrix} M_1 & M_2 \\ M_3 & M_4 \end{bmatrix} \tag{27}$$

In the block matrix above, $M_4 = \mathbf{0}$ can be readily identified due to the lack of inequality constraint dependence on control inputs $\mathbf{u}$. The remaining, nonzero, block matrices have the following general forms.

$$M_1 = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{eq}(k) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{eq}(k) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{eq}(k) \\ \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{eq}(k+1) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{eq}(k+1) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{eq}(k+1) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{eq}(k+N_p) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{eq}(k+N_p) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{eq}(k+N_p) \end{bmatrix} \tag{28}$$

$$M_2 = \begin{bmatrix} \frac{\partial}{\partial \mathbf{u}(k+1)}\mathbf{c}_{eq}(k) & \frac{\partial}{\partial \mathbf{u}(k+2)}\mathbf{c}_{eq}(k) & \dots & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\mathbf{c}_{eq}(k) \\ \frac{\partial}{\partial \mathbf{u}(k+1)}\mathbf{c}_{eq}(k+1) & \frac{\partial}{\partial \mathbf{u}(k+2)}\mathbf{c}_{eq}(k+1) & \dots & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\mathbf{c}_{eq}(k+1) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{u}(k+1)}\mathbf{c}_{eq}(k+N_p) & \frac{\partial}{\partial \mathbf{u}(k+2)}\mathbf{c}_{eq}(k+N_p) & \dots & \frac{\partial}{\partial \mathbf{u}(k+N_p)}\mathbf{c}_{eq}(k+N_p) \end{bmatrix} \tag{29}$$

$$M_3 = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{in}(k) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{in}(k) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{in}(k) \\ \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{in}(k+1) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{in}(k+1) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{in}(k+1) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial}{\partial \mathbf{x}(k+1)}\mathbf{c}_{in}(k+N_p) & \frac{\partial}{\partial \mathbf{x}(k+2)}\mathbf{c}_{in}(k+N_p) & \dots & \frac{\partial}{\partial \mathbf{x}(k+N_p)}\mathbf{c}_{in}(k+N_p) \end{bmatrix} \tag{30}$$

Given that the equality constraint equation used a first-order integration scheme, these matrices can be made increasingly specific to this problem.

$$M_1 = \begin{bmatrix} I/\Delta t & 0 & 0 & \dots & 0 \\ -I/\Delta t - \frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}(k+1),\mathbf{u}(k+1)) & I/\Delta t & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & -I/\Delta t - \frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\mathbf{x}(k+N_p-1)),\mathbf{u}(k+N_p-1) & I/\Delta t \end{bmatrix} \tag{31}$$

$$M_2 = \begin{bmatrix} -\frac{\partial}{\partial \mathbf{u}}\mathbf{f}(\mathbf{x}(k),\mathbf{u}(k)) & 0 & \dots & 0 \\ 0 & -\frac{\partial}{\partial \mathbf{u}}\mathbf{f}(\mathbf{x}(k+1),\mathbf{u}(k+1)) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -\frac{\partial}{\partial \mathbf{u}}\mathbf{f}(\mathbf{x}(k+N_p-1),\mathbf{u}(k+N_p-1)) \end{bmatrix} \tag{32}$$

$$M_3 = \begin{bmatrix} -2(\mathbf{x}(k+1|k)-\mathbf{x}_1^c)^T O_1 & 0 & 0 & \dots & 0 \\ 0 & -2(\mathbf{x}(k+2|k)-\mathbf{x}_1^c)^T O_1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -2(\mathbf{x}(k+N_p|k)-\mathbf{x}_1^c)^T O_1 \\ -2(\mathbf{x}(k+1|k)-\mathbf{x}_2^c)^T O_2 & 0 & 0 & \dots & 0 \\ 0 & -2(\mathbf{x}(k+2|k)-\mathbf{x}_2^c)^T O_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -2(\mathbf{x}(k+N_p|k)-\mathbf{x}_2^c)^T O_2 \\ \vdots & \vdots & \vdots & & \vdots \\ -2(\mathbf{x}(k+1|k)-\mathbf{x}_{N_o}^c)^T O_{N_o} & 0 & 0 & \dots & 0 \\ 0 & -2(\mathbf{x}(k+2|k)-\mathbf{x}_{N_o}^c)^T O_{N_o} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -2(\mathbf{x}(k+N_p|k)-\mathbf{x}_{N_o}^c)^T O_{N_o} \end{bmatrix} \tag{33}$$

Note that in the equation for M_1 above, since $x_i(k+1|k)$ must be consistent with the initial condition $x_i(k)$, which is not an optimisation variable, the first element is $I/\Delta t$. The Jacobian submatrices $\frac{\partial}{\partial \mathbf{x}}\mathbf{f}(\cdot)$, $\frac{\partial}{\partial \mathbf{u}}\mathbf{f}(\cdot)$ that appear in M_1

and M_2 above can be readily evaluated using Equation 11.

$$\frac{\partial}{\partial \mathbf{x}}\mathbf{f}\left[\mathbf{x}(k),\mathbf{u}(k)\right] = \begin{bmatrix} 0 & 0 & 0 & 0 & -V_T\sin\theta(k)\cos\psi(k) & -V_T\cos\theta(k)\sin\psi(k) \\ 0 & 0 & 0 & 0 & -V_T\sin\theta(k)\sin\psi(k) & V_T\cos\theta(k)\cos\psi(k) \\ 0 & 0 & 0 & 0 & -V_T\cos\theta(k) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{g}{V_T}\sec^2\phi(k) & 0 & 0 \end{bmatrix} \tag{34}$$

$$\frac{\partial}{\partial \mathbf{u}}\mathbf{f}\left[\mathbf{x}(k),\mathbf{u}(k)\right] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{35}$$

IPOPT requires constraint Hessian matrices be provided for each constraint equation at each point in the prediction horizon, thus requiring N_pN_o such matrices. As these matrices are exceedingly sparse, only the nonzero elements have been included below alongside their corresponding equation.

$$\frac{x(k+1)-x(k)}{\Delta t} - V_T\cos\theta(k)\cos\psi(k) \implies$$

$$\begin{bmatrix} \ddots & & & & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & V_T\cos\theta(k)\cos\psi(k) & -V_T\sin\theta(k)\sin\psi(k) & 0 \\ & 0 & 0 & 0 & -V_T\sin\theta(k)\sin\psi(k) & V_T\cos\theta(k)\cos\psi(k) & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \ddots \end{bmatrix} \tag{36}$$

$$\frac{y(k+1)-y(k)}{\Delta t} - V_T\cos\theta(k)\sin\psi(k) \implies$$

$$\begin{bmatrix} \ddots & & & & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & V_T\cos\theta(k)\sin\psi(k) & V_T\sin\theta(k)\cos\psi(k) & 0 \\ & 0 & 0 & 0 & V_T\sin\theta(k)\cos\psi(k) & V_T\cos\theta(k)\sin\psi(k) & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \ddots \end{bmatrix} \tag{37}$$

$$\frac{z(k+1)-z(k)}{\Delta t} + V_T\sin\theta(k) \implies$$

$$\begin{bmatrix} \ddots & & & & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & -V_T\sin\theta(k) & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \ddots \end{bmatrix} \tag{38}$$

$$\frac{\psi(k+1)-\psi(k)}{\Delta t} - \frac{g}{V_T}\tan\phi(k) \implies$$

$$\begin{bmatrix} \ddots & & & & & & \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & -\frac{2g}{V_T}\tan\phi(k)\sec^2\phi(k) \\ & & & & & & \ddots \end{bmatrix} \tag{39}$$

$$1 - \|\mathbf{x}_i(k+l) - \mathbf{x}^c\|^2_{O_j} \implies$$

$$\begin{bmatrix} \ddots & & & & & & \\ & O_{1,1} & 0 & 0 & 0 & 0 & 0 \\ & O_{2,1} & O_{2,2} & 0 & 0 & 0 & 0 \\ & O_{3,1} & O_{3,2} & O_{3,3} & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 & 0 \\ & & & & & & \ddots \end{bmatrix} \tag{40}$$

B F-16 PID Autopilots

To allow the F-16 model described above to track trajectories produced by the MPC scheme, four rudimentary autopilots were developed. Each autopilot uses commanded values, $\cdot_c$, as their inputs, and output control surface deflections. The first of these autopilots is a PID roll controller (see Figure 18).

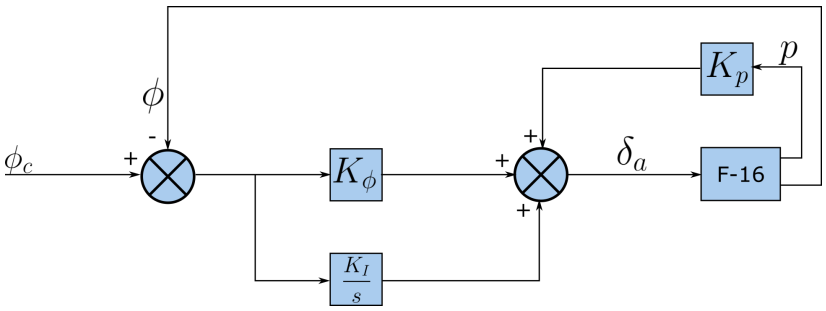


Figure 18: F-16 roll autopilot

The pitch autopilot, illustrated in Figure 19, used both the commanded pitch θ_c and altitude h_c as inputs. With the use of the $max()$ function, this allowed the autopilot to protect the aircraft from descending far below an altitude of 2 km.

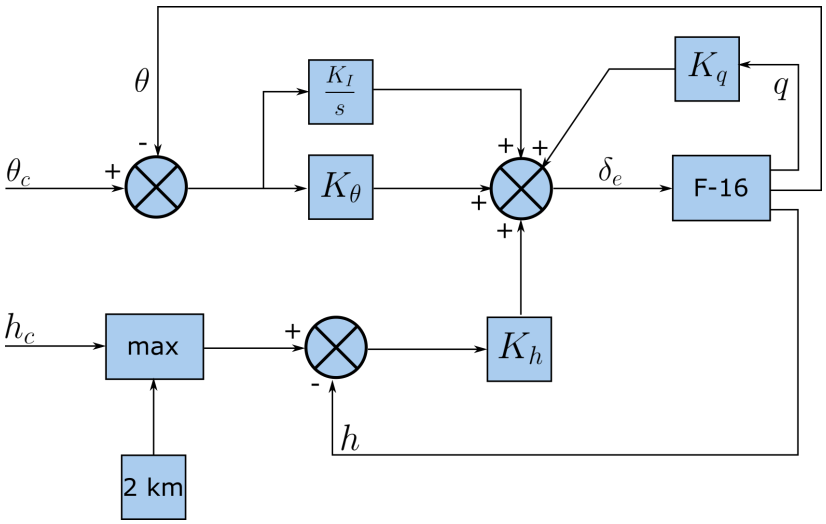


Figure 19: F-16 pitch autopilot

The yaw autopilot (see Figure 20) is largely identical to the roll autopilot, with the only notable exception being the use of sideslip β as feedback. This was used to to minimise the sideslip angle, and the nonlinearities associated with it.

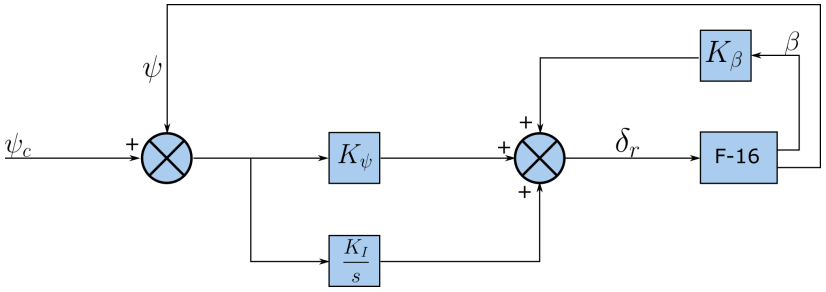


Figure 20: F-16 yaw autopilot

The airspeed autopilot (see Figure 21) is a proportial controller which, while simple, provided adequate tracking for the purposes of this project.

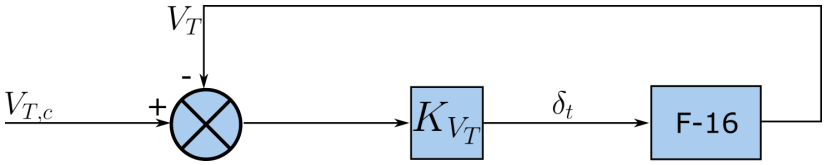


Figure 21: F-16 airspeed autopilot

C Ray-Tracing Algorithm

To simulate the detection of obstacles by each aircraft, a modified ray-tracing algorithm was used. This approach used the known width w between detected points to estimate an angular width γ, with an associated safety factor of 0.5. The algorithm below summarises the approach used. Although it is far from the most efficient implementation

possible, it was sufficient for the purposes of this project.

Algorithm 2: Ray-Tracing Algorithm

set mesh spacing, w;
sort by distance, r;
remove all $r > R_{det}$;
for *each remaining point from current position,* $\{x, y, z\}$ **do**
    calculate azimuth to each point, $\psi = \text{atan2}\left(\frac{y-y_p}{x-x_p}\right)$;
    calculate inclination to each point, $\theta = \text{atan2}\left(\frac{z-z_p}{r}\right)$;
end
$chosenPts \leftarrow \{\}$;
for *each point* g **do**
    calculate angular width for each point in $chosenPts$, $\gamma = 0.5 \max\left(|\frac{w\cos\psi}{r}|, |\frac{w\cos\theta}{r}|\right)$;
    $shouldAddPt \leftarrow 1$;
    for *each other point* h **do**
        if $|\psi_g - \psi_h| < \gamma$ and $|\theta_g - \theta_h| < \gamma$ **then**
            $shouldAddPt \leftarrow 0$;
            break;
        end
    end
    if $shouldAddPt$ **then**
        $chosenPts \leftarrow \{chosenPts, g\}$;
    end
    if $|chosenPts| \geq |\mathcal{N}^v|$ **then**
        break;
    end
end

Figure 22 illustrates how the angular width γ was used to determine which points were visible (black), detected (red), or not visible (grey). As can be seen in Section 5.3, this method worked well, albeit at significant computational expense.

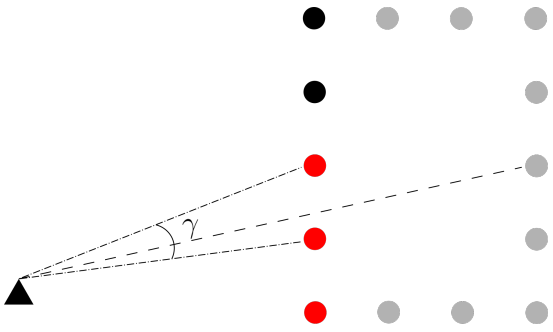


Figure 22: Visual representation of angular width used to determine which points are visible to an aircraft