NONLINEAR MODEL PREDICTIVE CONTROL FOR OMNIDIRECTIONAL ROBOT MOTION PLANNING AND TRACKING

Timothy A. V. Teatro, J. Mikael Eklund SMIEEE

Department of Electrical, Computer and Software Engineering,
University of Ontario Institute of Technology, Oshawa, ON L1H 7K4

ABSTRACT

We present a nonlinear model predictive control algorithm for online motion planning and tracking of an omnidirectional autonomous robot. The formalism is based on the minimization of a control Hamiltonian related to the cost function. This minimization is constrained by a nonlinear plant model. The algorithm considers point obstacles and uses a potential field to penalize proximity to those obstacles. A simple and demonstrative example simulation is presented.

Index Terms— NMPC, Control, Motion, Path, Tracking, Planning, Autonomous.

1. INTRODUCTION

In the past four decades, many methods have been developed for planning and controlling robot motion. In the beginning, many were suited toward very simple robots with crude sensors and limited computing power. As technology has progressed, so has the methodology. Now there are numerous planning routines which incorporate rich sensor data with elegant and computationally demanding procedures.

Model predictive control (MPC) was originally developed for process control [1, 2]. The method involves using a mathematical model to predict the consequences of a specified set of control input over a time called the preview horizon. That model is used in conjunction with optimal control methods to find an optimal control plan for the preview horizon. The optimization is performed online, over short time segments yielding short time control plans. The global state trajectory is formed by executing subsets of the local plans. If a non-linear system model is used, then the method is referred to as Non-linear MPC (NMPC).

This article outlines the development of a simple NMPC path planner and tracker as it is applied to an omnidirectional vehicle with (only) steering rate control.

The underlying mathematical framework is based on the optimization problem phrased as a Hamiltonian minimization. Algorithms for both the minimization, and a governing NMPC controller will be shown to perform as it was designed to.

2. VEHICLE DYNAMICS

For this research a simple robot is modelled wherein only position, speed, steering angle and steering rate are considered. The steering angle always represents the direction of travel (precluding the possibility of slipping), and the speed is taken to be constant at v. The direction of travel is controlled in time by varying the steering rate. The discrete-time state ($\boldsymbol{q}_k$) and control ($\boldsymbol{u}_k$) vectors are

$$\boldsymbol{q}_k = \begin{bmatrix} x_k \; \dot{x}_k \; y_k \; \dot{y}_k \; \theta_k \end{bmatrix}^{\mathrm{T}}, \quad and \quad \boldsymbol{u}_k = \begin{bmatrix}\dot{\theta}\end{bmatrix}. \tag{1}$$

which express the rectangular x and y coordinates with corresponding speeds $\dot{x}$ and $\dot{y}$, along with steering angle θ and steering rate $\dot{\theta}$ at time-step k. The time-step index k is related to real time t as $t = kT$ with interval T. Since the control input vector contains only a single element, it will henceforth be written as a scalar $u_k = \dot{\theta}$.

Integrating the equations of motion with a first-order Euler scheme yields the discrete equations approximating the dynamics of the vehicle as

$$\boldsymbol{a}(\boldsymbol{q}_k, \boldsymbol{u}_k) = \boldsymbol{q}_{k+1} = \begin{bmatrix} x_k + \dot{x}_k T \\ v\cos\theta_{k+1} \\ y_k + \dot{y}_k T \\ v \sin\theta_{k+1} \\ \theta_k + u_k T \end{bmatrix}. \tag{2}$$

3. NMPC CONTROLLER DESIGN

The NMPC controller is founded by the optimization problem

$$u^* = \underset{\{u_1, u_2, \ldots u_N\}}{\arg\min} \; J(\boldsymbol{q}, u),$$

wherein, the designed cost functional $J(\boldsymbol{q}, u)$ is minimized with respect to the control input set $u = \{u_1, u_2, \ldots u_N\}$, over the NMPC horizon N, giving the optimal control set u^*. The control algorithm with use the model in (2) to obtain an approximation of $\boldsymbol{q}$ from u.

For this research, the cost functional is chosen as

$$\begin{aligned} J(\boldsymbol{q}, \boldsymbol{u}) =\; & \boldsymbol{\delta}_N^{\mathrm{T}} \boldsymbol{Q}_0 \boldsymbol{\delta}_N + \sum_{k=1}^{N-1} \Bigg(R u_k^2 + \boldsymbol{\delta}_k^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{\delta}_k \\ & + \sum_{(X_i, Y_i) \in \mathcal{O}} \frac{1}{(x_k - X_i)^2 + (y_k - Y_i)^2 + \epsilon} \Bigg) \\ =\; & \phi(\boldsymbol{\delta}_N) + \sum_{k=1}^{N-1} L\big(\boldsymbol{q}_k, u_k\big). \end{aligned} \tag{3}$$

where the tracking error $\boldsymbol{\delta}_k$, large input value and proximity to point obstacles $\mathcal{O}$ are penalized. That is, these phenomena increase the cost functional which we are minimizing. The terminal cost $\phi(\boldsymbol{\delta}_N) = \boldsymbol{\delta}_N^{\mathrm{T}} \boldsymbol{Q}_0 \boldsymbol{\delta}_N$ encourages the end of the path ($\boldsymbol{q}_N$) to be as close to the target as possible, and the cost of running operation is encapsulated in the running cost function $L\big(\boldsymbol{q}_k, u_k\big)$. The weighting parameter matrices $\boldsymbol{Q}_0$, $\boldsymbol{Q}$ and scalar R, and the scaling parameter ϵ are chosen to reflect the relative balance of their respective cost terms. Simple quadratic forms are chosen for terminal, input and tracking costs. The obstacle avoidance term scales as the inverse square of the robot's proximity to the obstacles. These relationships are chosen so that the cost functional is readily differentiable.

The tracking error $\boldsymbol{\delta}_k = \boldsymbol{q}_k^{\text{ref}} - \boldsymbol{q}_k$ is the difference between the state vector and the reference path $\boldsymbol{q}_k^{\text{ref}}$. For this research, the reference path is computed as the straight line from $\boldsymbol{q}_1$ to the target, sampled in intervals of vT. This path is regenerated for each progression of the NMPC horizon.

The weighting scalar R deserves special consideration, since it controls level of holonomicity we observe. If this parameter is set to zero, the robot can make instantaneous changes in direction since nothing else limits the turn radius of the vehicle. However, sudden changes in direction may pose practical problems, so modulating R will indirectly add the turn radius into the cost balance. The value of R can even be a function of velocity so that narrower turns can be executed at lower speeds.

Following the path of similar research [3, 4, 5], the cost measure in Equation (3) is augmented with a set of Lagrange multipliers $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_N\}$ enforcing Equation (2) as a constraint:

$$J_\Lambda = \phi(\boldsymbol{\delta}_N) + \sum_{k=1}^{N-1} \left(L(\boldsymbol{q}_k, \boldsymbol{u}_k) + \boldsymbol{p}_{k+1}^{\text{T}} \left(\boldsymbol{a}(\boldsymbol{q}_k, \boldsymbol{u}_k) - \boldsymbol{q}_{k+1} \right) \right). \tag{4}$$

As long as the control history corresponds to the state trajectory in accordance with the model, it will always be true that $\boldsymbol{q}_{k+1} = \boldsymbol{a}(\boldsymbol{q}_k, \boldsymbol{u}_k)$ and the constraint terms will be zero. Therefore, we may choose the Lagrange multipliers arbitrarily without affecting the balance of Equation (4).

In following with Pontryagin's formalism [6, Ch. 5], we define the control Hamiltonian as

$$\mathcal{H}_k = L(\boldsymbol{q}_k, \boldsymbol{u}_k) + \boldsymbol{p}_{k+1}^{\text{T}} \boldsymbol{a}(\boldsymbol{q}_k, \boldsymbol{u}_k). \tag{5}$$

Combining Equations (4) and (5), the augmented cost measure becomes

$$J_\Lambda = \phi(\boldsymbol{\delta}_N) + \boldsymbol{p}_N^{\text{T}} + \sum_{k=1}^{N-1} \left(\mathcal{H}_k - \boldsymbol{p}_k^{\text{T}} \boldsymbol{q}_k \right) + \mathcal{H}_0 \tag{6}$$

In order to minimize J_Λ, we compute the differential change in J_Λ with respect to its dependants

$$\mathrm{d}J_\Lambda = \left(\frac{\partial \phi}{\partial \boldsymbol{q}_N} - \boldsymbol{p}_N^{\text{T}} \right) \boldsymbol{q}_N + \sum_{k=1}^{N-1} \left[\left(\frac{\partial \mathcal{H}_k}{\partial \boldsymbol{q}_k} - \boldsymbol{p}_k^{\text{T}} \right) \mathrm{d}\boldsymbol{q}_k + \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{u}_k} \mathrm{d}\boldsymbol{u}_k \right] + \frac{\partial \mathcal{H}_0}{\partial \boldsymbol{q}_0} \mathrm{d}\boldsymbol{q}_0 + \frac{\partial \mathcal{H}_0}{\partial \boldsymbol{u}_0} \mathrm{d}\boldsymbol{u}_0. \tag{7}$$

Now in a position to make choices regarding our Lagrange multipliers so as to simplify Equation (7) as much as possible, we choose

$$\boldsymbol{p}_N^{\text{T}} = \frac{\partial \phi}{\partial \boldsymbol{q}_N} \quad \text{and} \quad \boldsymbol{p}_k^{\text{T}} = \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{q}_k} + \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{\delta}_k} \frac{\partial \boldsymbol{\delta}_k}{\partial \boldsymbol{q}_k}$$

or more explicitly

$$\boldsymbol{p}_N^{\text{T}} = \boldsymbol{\delta}_N^{\text{T}} \boldsymbol{Q}_0 \frac{\partial \boldsymbol{\delta}_N}{\partial \boldsymbol{q}_N} \tag{8}$$

$$\boldsymbol{p}_k^{\text{T}} = \boldsymbol{\delta}_k^{\text{T}} \boldsymbol{Q} \frac{\mathrm{d}\boldsymbol{\delta}_k}{\mathrm{d}\boldsymbol{q}_k} + \boldsymbol{p}_{k+1}^{\text{T}} \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{q}_k} + \frac{\partial \Phi_k}{\partial \boldsymbol{q}_k}, \tag{9}$$

with Φ_k as the obstacle potential at step k. Consequently, Equation (7) is reduced to

$$\mathrm{d}J_\Lambda = \sum_{k=1}^{N-1} \frac{\partial \mathcal{H}_k}{\partial \boldsymbol{u}_k} \mathrm{d}\boldsymbol{u}_k + \boldsymbol{p}_0^{\text{T}} \mathrm{d}\boldsymbol{q}_0. \tag{10}$$

The right-most term of the above equation is zero, since $\boldsymbol{q}_0$ is constant and $\mathrm{d}\boldsymbol{q}_0$ is therefore also zero. The final effort is in evaluating $\partial \mathcal{H}_k / \partial \boldsymbol{u}_k$ by expanding and differentiating Equation (5), which reveals that

$$\frac{\partial \mathcal{H}_k}{\partial \boldsymbol{u}_k} = \boldsymbol{u}_k \boldsymbol{R} + \boldsymbol{p}_{k+1}^{\text{T}} \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{u}_k}. \tag{11}$$

Equation (10) shows that the curvature of J_Λ and the curvature of $\mathcal{H}_k$ with respect to $\boldsymbol{u}$ are proportional. The piece-wise (at each time step) minimization of $\mathcal{H}_k$ is then equivalent to the global minimization of J_Λ. In this context, Equation (7) can be interpreted as relating the gradient of J_Λ to the values of $\partial \mathcal{H}_k / \partial \boldsymbol{u}_k$.

A basic gradient decent algorithm implementing this formalism is listen in Algorithm 1. Starting from some initial condition $\boldsymbol{q}_1$,

Algorithm 1 A general algorithm for iteratively optimizing a control history against a cost function.

```
$\boldsymbol{u} \leftarrow$ initial guess
$\boldsymbol{q}_1 \leftarrow$ starting position
**while** norm$([\partial\mathcal{H}_1/\partial\boldsymbol{u}_1\ \partial\mathcal{H}_2/\partial\boldsymbol{u}_2 \cdots \partial\mathcal{H}_N/\partial\boldsymbol{u}_N])$ > *tolerance* **do**
    **for** $k = 1, \ldots, (N-1)$ **do**
        $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{a}(\boldsymbol{q}_k, \boldsymbol{u}_k)$
    **end for**
    **for** $k = (N-1), \ldots, 1$ **do**
        Compute Lagrange Multiplier $\boldsymbol{p}_k$ from Eq. (8) or (9).
    **end for**
    **for** $k = 1, \ldots, (N-1)$ **do**
        Compute gradient element $\partial\mathcal{H}_k/\partial\boldsymbol{u}_k$ from Eq. (11).
    **end for**
    **for** k=1,..., N **do**
        $\boldsymbol{u} \leftarrow \boldsymbol{u} - \Delta\left(\partial\mathcal{H}_k/\partial\boldsymbol{u}_k\right)$
    **end for**
**end while**
```

and some initial guess for the optimal control history $\boldsymbol{u}$, one can use the derived formalism to iteratively improve $\boldsymbol{u}$ to approach $\boldsymbol{u}^*$. The relationships developed in this section are used to find the gradient elements $\partial \mathcal{H}_k / \partial \boldsymbol{u}_k$, which are then used to step $\boldsymbol{u}$ towards $\boldsymbol{u}^*$. Convergence rate and stability are balanced (in part) by the step factor Δ. The iteration loop is broken when the Euclidean norm of the vector formed by the gradient elements is sufficiently small. This algorithm is by no means the best way to implement the minimization, but it may be the most straightforward. Sacrificing elegance (and performance) for intuition, it illustrates the basic relationships that might lead us to an optimal control history.

A NMPC algorithm based on Algorithm 1 is listed in Algorithm 2. Speaking generally, NMPC limits the scope of the optimization problem to N time-steps, the *prediction horizon*. The optimization is carried out for a segment of the path to the target that is N steps long. A subset of that optimal path, the first C steps are executed, and the optimization is carried out again for the next set of N steps. This process is repeated until the target is reached. The integer C is the *control horizon*. The general configuration of an NMPC controller is illustrated in the block diagram in Figure 1. In that figure, the global path planner takes the target and plans a route

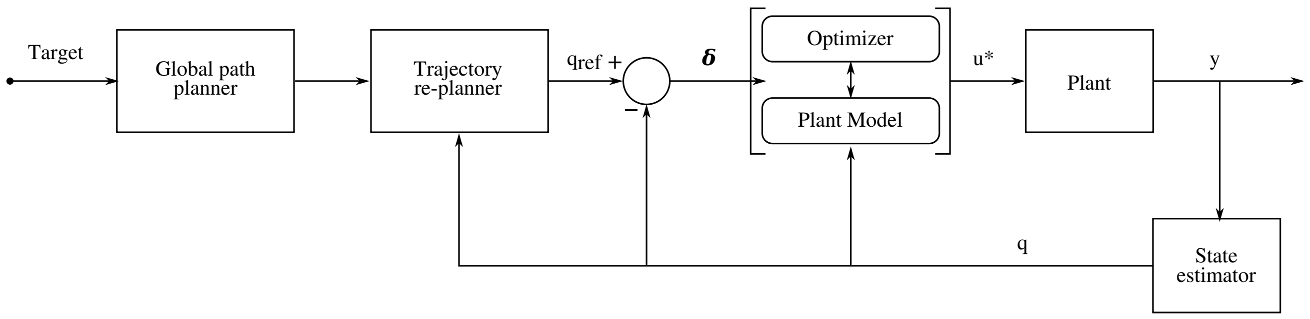

Fig. 1. A block diagram for a system with MPC controller.

Algorithm 2 A general statement of the Model Predictive Control algorithm

```
$\boldsymbol{u} \leftarrow$ initial guess
$\boldsymbol{q}_1 \leftarrow$ starting position
**while** not sufficiently close to target **do**
    Get $\boldsymbol{u}^*$ and $\boldsymbol{q}^*$ over $N$ using Algorithm 1
    Execute the first $C$ steps of $\boldsymbol{u}^*$
    **for** $k = C, \dots, N$ **do**
        $\boldsymbol{q}_{k-C} \leftarrow \boldsymbol{q}_k^*$
        $\boldsymbol{u}_{k-C} \leftarrow \boldsymbol{u}_k^*$
    **end for**
**end while**
```

to it. That part of the diagram is typically offline. The *re-planner* however runs online and adapts the path dynamically to new information. The desired path is passed into the MPC block which consists of an optimizer and system model which pass information back and fourth until it has computed the optimal path $\boldsymbol{u}^*$. A subset of $\boldsymbol{u}^*$ is executed by the plant (the mobile robot), and the resulting pose of the system and other output information is available in y. Whatever routines and devices are needed to convert y into the state vector $\boldsymbol{q}$ is encompassed in the *state estimator*. The state estimator passes the state vector to the blocks that require it.

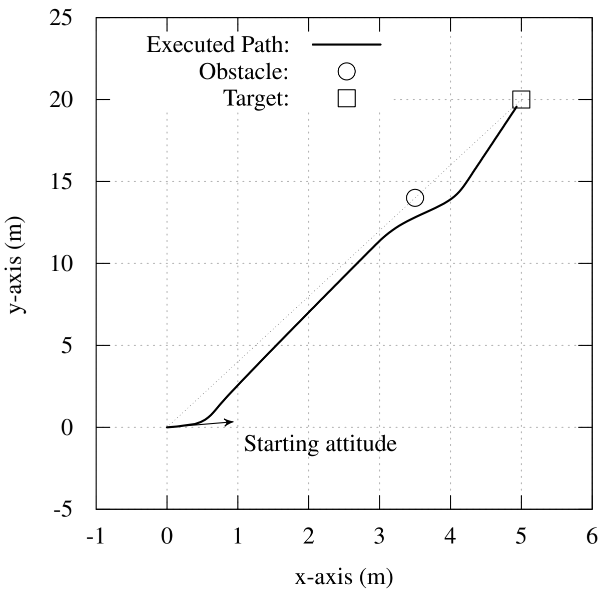

Fig. 2. An illustrative summary of the simulation described in this chapter.

4. SIMULATION RESULTS

As an illustrative example, a simulation was run for a target set at the point $[5, 20]$ in the xy-plane, which forms a steep angle to the x-axis. The robot was set to start at the origin, with a steering angle $\theta_1 = \pi/9$, which forms a narrow angle to the x-axis. This contrast serves to emphasize the anholonomic nature of the path. A point-obstacle was placed at $[3.5, 14]$, which is 0.7 times the vector to the target. The prediction horizon $N = 20$ and control horizon $C = 3$ was set.

The results are summarized in Figure 2.

The path begins in the direction of $\theta = \pi/9$, and gradually steers towards the target. While executing the turn, the steering rate is tempered by the cost measure term penalizing large input values. The tendency to move towards the target is motivated by the terminal cost and tracking terms in the cost measure. The tracking term always favours the linear path from the robot's current location to the target (henceforth referred to as the direct route). The turn radius is decided by the balance of these terms, as it is not prescribed by the model. Figure 3 (left) illustrates how, for this steering event, the individual prediction horizons favour the direct routes. But as the robot moves, the direction of direct routes change too, until the robot's velocity is in alignment with the direct route.

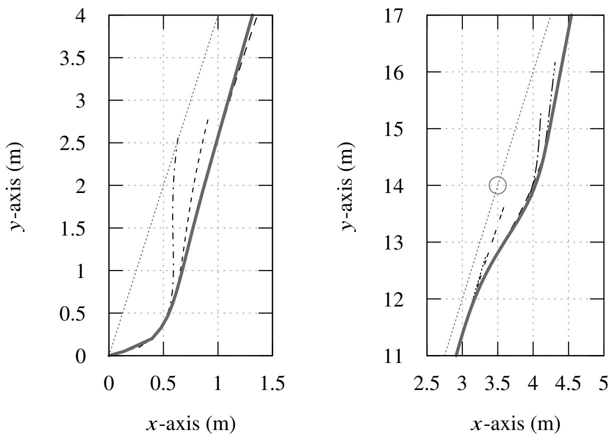

Fig. 3. Illustrations of the predicted optimal paths over each prediction horizon during the two major steering events. Dotted lines are the paths for the individual horizons and the bold solid line is the executed path. (left) At the beginning as the robot steers from the initial attitude to direct its motion to the target. (right) As the robot steers to avoid the obstacle at $[3.5, 7]$.

Once the robot has turned to direct its motion toward the target, it moves in a straight line until the edge of the prediction trajectory comes close enough to the obstacle. At that point, the path begins to curve around the obstacle to avoid it. This steering event differs from the first one in that the cost function is now a stage for three competing terms: tracking, input value *and* obstacle avoidance. The turn rate will therefore depend on how strong the tendency to avoid obstacles is, and the prediction horizon length. A longer prediction horizon causes the robot to react sooner to the obstacle allowing favour of the input penalty. The prediction paths for this steering event are illustrated in 3 (right). Notice how, once again, the prediction paths have greater curvature than the executed one. This is due again to the tendency to follow the direct route, even if it moves through the obstacle. The obstacle avoidance term will become dominant at some point, which does force the robot away from the direct route. The curvature of the prediction paths show to robot's tendency to return to the direct route after the object is cleared. But, as the robot navigates, the direct route changes so that the robot always tends to the shortest distance between its *current* location and the target.

After the turn to avoid the obstacle, the robot continues on its new straight line path, until reaching the target.

Figure 4 shows the steering angle θ_k and corresponding input value $u_k = \dot{\theta}_k$ during the simulation. The features of both major steering events are apparent from the plots. Made particularly clear is the high steering rate at the beginning of the simulation due to the narrow starting attitude. It is nearly an order of magnitude higher than the steering rates associated with the turn to avoid the obstacle.

5. CONCLUSIONS

An NMPC based algorithm is developed with an underlying optimization framework for a minimialistic omnidirectional robotic platform. Based on the prescribed design characteristics, the NMPC control algorithm was shown to behave as expected. Obstacles are avoided and the holonomicity of the executed path is controlled so that high acceleration maneuvers can be discouraged in the cost balance.

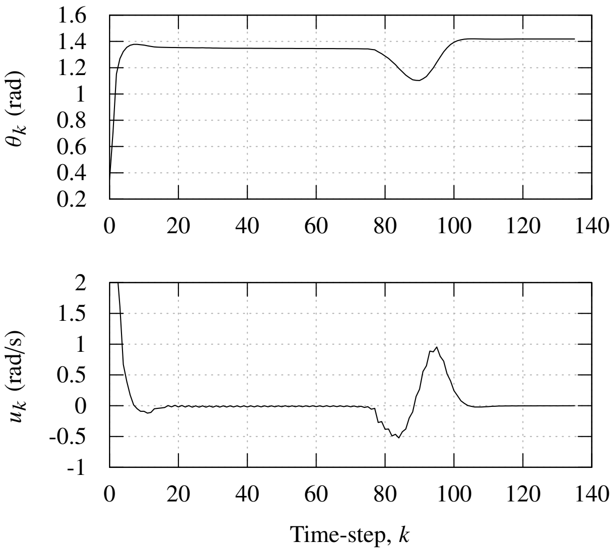

Fig. 4. The steering angle (top) and steering rate (bottom) during the simulation. The vertical axis range was truncated for clarity. The steering rate at the beginning of the simulation was 7.28 rad/s, but the axis only shows as high 2 rad/s.

6. REFERENCES

[1] S.Joe Qin and Thomas a. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, July 2003.

[2] J Richalet, A Rault, J.L. Testud, and J Papon, "Model predictive heuristic control," *Automatica*, vol. 14, no. 5, pp. 413–428, Sept. 1978.

[3] F Fahimi, "Non-linear model predictive formation control for groups of autonomous surface vessels," *International Journal of Control*, vol. 80, no. 8, pp. 1248–1259, Aug. 2007.

[4] Gordon J Sutton and Robert R Bitmead, "Performance and Computational Implementation of Nonlinear Model Predictive Control on a Submarine," in *Nonlinear Model Predictive Control*, Frank Allgöwer, Alex Zheng, and Christopher I Byrnes, Eds., vol. 26 of *Progress in Systems and Control Theory*, pp. 461–472. Birkhäuser Basel, 2000.

[5] J Mikael Eklund, Jonathan Sprinkle, and S Shankar Sastry, "Switched and Symmetric Pursuit/Evasion Games Using Online Model Predictive Control With Application to Autonomous Aircraft," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 3, pp. 604–620, May 2012.

[6] Donald E. Kirk, *Optimal Control Theory: An Introduction*, Dover Publications, 2004.