MOVING HORIZON CONTROL IN THE
GENERATION OF INITIAL SOLUTION TO DYNAMIC
OPTIMIZATION PROBLEM

November 21, 2005

Helsinki University of Technology
Department of Engineering Physics and Mathematics
Systems Analysis Laboratory

Henri Hytönen
54394U
Contents

1 Introduction ................................................. 2

2 Moving Horizon Control Method ......................... 4
   2.1 Description of the Algorithm ....................... 4
   2.2 Generating Initial Trajectory for the Algorithm .... 7
   2.3 Sequential Quadratic Programming (SQP) .......... 8
   2.4 Determination of End State ....................... 10

3 Example Problem ............................................. 13
   3.1 Model Dynamics ........................................ 14
   3.2 Problem Formulation .................................. 15

4 Numerical Results ......................................... 17
   4.1 Results of the MHC method .......................... 17
   4.2 Comparison of Methods ............................. 19

5 Summary and Conclusions ................................ 22

References ..................................................... 25
1 Introduction

This research project was inspired by the work [4], where an optimal control problem was formulated to analyze whether missile avoidance in air-to-air combat would be possible by utilizing the constraints of the air-to-air missile’s seeker head. The gimbal angle of the seeker head or the angle between the missile centerline and the direction of the seeker head is limited. Therefore the missile will lose its lock on the target if the target maneuvers outside the seeker head movement area. Similarly, the lock is also lost if the seeker head can not turn rapidly enough in order to follow the target. In the study [4] it was found that some trajectories for the evading aircraft exceed the gimbal limit of the missile’s seeker head and thus breaking the lock-on of the missile. However, the turn rate constraint was not thoroughly examined due to the failure of the nonlinear solver to reliably converge to a solution. The numerical computation of the optimal trajectories in different initial states turned out to be difficult as the nonlinear solver used in the dynamic optimization problem required a somewhat accurate initial guess, which was generated with a genetic algorithm (for a description of the genetic algorithm used, see [10]), which in turn turned out to be somewhat time-consuming. Also the stochastic nature of the initial guess made it hard to accurately measure the difference between the initial guess and the final optimal solution.

This study proposes an alternative technique for the generation of the initial guess that can be used in dynamic optimization applications. We apply a moving horizon control (MHC) approach to the generation of the initial guess by stepwise calculation of the (sub)optimal state and control trajectories. For each step, one smaller optimization problem is solved, that is for a limited time interval forward and the whole initial solution is obtained by combining these stepwise solutions. If the optimization horizon is long enough and if the exact optimal trajectory is not important, the result given by the moving horizon optimization could be accurate enough so that actual full-horizon open-loop optimization of the problem, based e.g. on single or multiple direct or indirect shooting methods and sequential quadratic programming (SQP), is not necessarily needed in all applications. In this study, the initial solution algorithm is applied to the missile evasion prob-
1 INTRODUCTION

lem described above, but similar technique should be easily applicable to other
tproblems of similar nature as well.

The remainder of the study is organized as follows. In the Chapter 2, we present
the basic principles of the moving horizon control approach and an outline of the
optimization algorithm utilizing the method. In Chapter 3, we give a brief overview
of the moving horizon control method applied to the missile evasion problem in
air combat. Numerical results of the method applied to this problem are given in
Chapter 4. Finally, Chapter 5 concludes the study and discusses the applicability
of the proposed formulation as well as presents proposals for further study.
2 Moving Horizon Control Method

This chapter presents our version of moving horizon control method. Section 2.1 presents an outline of the algorithm used and illustrates the moving horizon especially in the context of the example problem introduced later in the paper. In Section 2.2 we discuss how an initial trajectory is generated for each interval. This initial trajectory is required by the optimization algorithm. 2.3 briefly reviews the sequential quadratic programming algorithm used in the optimization of the single steps in the moving horizon. Section 2.4 discusses the determination of the endpoint of the algorithm, illustrated again in the scope of our example problem.

Moving horizon control (MHC), also called model predictive control (MPC), has been proposed as a method to optimize control trajectories as dynamic optimization problems are generally very hard to solve, even in the absence of constraints (see [2]). Although the method yields suboptimal solutions, it is particularly practical when the optimization is performed on-line, i.e. the solution obtained during a single time interval is used as a control during the next one. In addition to traditional control problems of dynamic physical systems, MHC has also been applied to numerous other applications, e.g. dynamic games in economic context [11].

2.1 Description of the Algorithm

The moving horizon control algorithm proceeds by optimizing the control variable in discrete steps in time. At each time step, an optimization subproblem up to \( n \) time steps ahead is solved. We then use the found optimal control to reach the next step from current step. During the next step, we then continue optimization and extend the time horizon one step further. This is continued until the endpoint is reached. In our applications the performance measure to be optimized is a function of end state only. With the moving horizon control, we optimize each step with the performance measure calculated at the end of the step, rather than at the endpoint of the whole optimization interval. In some applications, where the value of the performance criterion depends on the whole path in a highly nonlinear way, this might not be a suitable approach.
In the remainder, we will consider dynamic optimization problems with the dynamics of the following type

$$\begin{align*}
\max & \quad f(\mathbf{x}(t_f), t_f) \\
\text{s.t.} & \quad \dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t) \\
& \quad 0 = h(\mathbf{x}(t), \mathbf{u}(t), t) \\
& \quad t \in [t_0, t_f],
\end{align*}$$

(2.1)

where $\mathbf{x}(t)$ is the state vector, $f(\mathbf{x}(t_f), t_f)$, is the performance criterion, depending only on the final state and time. In our applications, the end time $t_f$ can also be an unknown variable. $\mathbf{u}(t)$ is the control vector and $g(\mathbf{x}(t), \mathbf{u}(t), t)$ gives the derivative of $\mathbf{x}$ as a function of $\mathbf{x}$, $\mathbf{u}$, and $t$. $h(\mathbf{x}(t), \mathbf{u}(t), t)$ are the constraints associated with $\mathbf{x}$ and $\mathbf{u}$. We divide the problem (2.1) to time steps of size $\Delta t$ and denote these steps by $t_0, t_1, \ldots, t_f$ and the corresponding states by $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_f$. Let our horizon length be $n\Delta t$, i.e. it consists of $n$ steps of size $\Delta t$. In other words, at each $t_p$, a dynamic optimization problem is solved in the interval $[t_p, t_p + n\Delta t]$. This time interval is denoted by $T_p$. At each interval $T_p$, we thus try to find the control trajectory $\mathbf{u}(t)$ that maximizes $f(\mathbf{x}(t_{p+n}), t_{p+n})$. When the optimal control trajectory $\mathbf{u}^*$ is found, we integrate the system forward with

$$\mathbf{x}_{p+1} = \mathbf{x}_p + \int_{t_p}^{t_p + \Delta t} g(\mathbf{x}(\tau), \mathbf{u}^*(\tau), \tau) \, d\tau. \quad (2.2)$$

Note that for the optimization and the integration, the discretization can be internally more dense than $\Delta t$, so that the results for optimization and integration, if performed numerically, yield results that are accurate enough.

For each optimization subproblem, an initial solution must be constructed so that the numerical solver could proceed. This must be done at each step before the actual optimization. Similarly, at each step we need to check whether the next subproblem will be the last one, i.e. whether or not $t_f \in T_p$. The moving horizon control method examined can then be roughly outlined as follows:

1. Generate initial trajectory for the new interval part $[t_p + (n-1)\Delta t, t_p + n\Delta t]$.

2. Estimate $\hat{t}_f$ so that $d(\hat{t}_f) = d_f$. If $\hat{t}_f \in T_p = [t_p, t_p + n\Delta t]$, set additional equality
2 MOVING HORIZON CONTROL METHOD

Figure 2.1: Optimization horizons \( T_p = [t_p, t_p + n\Delta t] \) and discretization in the example problem.

constraint \( d(t_f) = d_f \).

3. Solve the subproblem with SQP method in the interval \( T_p \).

4. If the endpoint condition was set in step 1, goto 4b. Otherwise, goto step 4a.

   (a) Add part \( [t_p, t_p + \Delta t] \) to the set of solved trajectory intervals and set \( p \rightarrow p+1 \). Goto step 1.

   (b) Add part \( [t_p, t_p + n\Delta t] \) to the set of solved trajectory intervals. Goto step 5.

5. Interpolate the resulting trajectory to the desired accuracy.

Step 1 is discussed in Section 2.2, step 2 is considered in Section 2.4 and step 3 in Section 2.3. An illustration of the moving horizon control optimization is given in Fig. 2.1. Here, we optimize the trajectory of a fighter aircraft that is evading a missile launched towards it. The objective is to evade the missile by maximizing the final line of sight angle between the missile’s seeker head and the missile centerline (see [4]). The optimization ends when the distance between the aircraft and the missile is equal to \( d_f \). At each point in time \( t_p \), we first check
whether the estimated endpoint \( \hat{t}_f \) is in the interval \( \mathcal{T}_p \). We already have the initial trajectory for the interval \([t_p, t_p + (n - 1)\Delta t]\) from the previous optimization step and therefore only need to generate the initial trajectory for the interval \([t_p + (n - 1)\Delta t, t_p + n\Delta t]\). In Fig. 2.1, we have \( n = 3 \). When we have the initial trajectory for \( \mathcal{T}_p \), we optimize the part of the aircraft trajectory with the SQP algorithm and integrate the system \( \Delta t \) forward with (2.2). These steps are then continued until the endpoint is met. We will return to the problem illustrated in the figure in more detail in Chapters 3 and 4.

## 2.2 Generating Initial Trajectory for the Algorithm

Even for the limited horizon optimization, we need an initial solution for the optimization algorithm for every interval \( \mathcal{T}_k \). In order to keep the algorithm quick and efficient, we do not want the algorithm to use too much time to generating an initial solution to an initial solution. Therefore, the calculated initial solution needs to be simple and easily computable. The resulting trajectory needs however to be feasible so that the nonlinear optimization algorithm could find the optimal solution. In our application, we will use the final controls in the current optimized trajectory to calculate the initial solution for the next step.

Due to the overlap of the successive optimization horizons, the initial solution need to be formulated only for the new addition to the optimization interval, since the results from the previous integration can be used as an initial solution for the other \( n - 1 \) steps. To obtain the initial solution for the final interval \([t_p + (n - 1)\Delta t, t_p + n\Delta t]\), we use the final controls that were given in the last optimization for the whole remaining interval. In other words, we set \( u(t) \equiv u(t_p + (n - 1)\Delta t) \) for \( t \in [t_p + (n - 1)\Delta t, t_p + n\Delta t] \). This way, we will have continuous control trajectories. The initial solution for the last part of the optimization horizon is therefore

\[
x_{p+1} = x(t_p + (n - 1)\Delta t) + \int_{t_p + (n-1)\Delta t}^{t_p + n\Delta t} g(x(\tau), u(t_p + (n - 1)\Delta t), \tau) \, d\tau. \tag{2.3}
\]

Additionally, we need to establish an initial solution to the first step of the algo-
rithm, where there are no previous intervals that have already been solved. In this case, we can use the controls in the initial state of the problem, and integrate the interval $T_0$ with those controls

$$x(t_0 + n\Delta t) = \int_{t_i}^{t_0 + n\Delta t} g(x(\tau), u(t_0), \tau) \, d\tau. \quad (2.4)$$

The methods used above in the calculation of the initial trajectories are not the only possible options to arriving at suitable initial solutions. We can use also other heuristic methods to obtain good starting values for the optimizer. It must be noted, however, that the controls resulting in initial trajectory can not be such that the possible constraints $h(x(t), u(t), t)$ would be violated during the interval, which would lead to infeasible initial solution.

### 2.3 Sequential Quadratic Programming (SQP)

In the study, each MHC subinterval is optimized with SQP method. The SQP or sequential quadratic programming algorithm is a method to solve constrained nonlinear programming problems. SQP employs Newton’s method to directly solve Karush-Kuhn-Tucker or KKT conditions (see e.g. [1]) of the original problem. The subproblem that needs to be solved is the quadratic approximation of the Lagrangian of the original problem with linear approximations of the constraints. Consider a nonlinear problem in the regular form

$$\min \quad f(y) \quad \quad y \in \mathbb{R}^n$$

$$\text{s.t.} \quad \quad g_i(y) \leq 0 \quad \quad i = 1, \ldots, m$$

$$h_i(y) = 0 \quad \quad i = 1, \ldots, l$$

(2.5)

where inequality and equality constraints $g_i$ and $h_i$ are assumed to be continuously twice-differentiable. Given some iterate $y_k$ with its Lagrange multipliers $u_k \geq 0$ and $v_k$ corresponding to inequality and equality constraints, the following
2 MOVING HORIZON CONTROL METHOD

A quadratic programming problem is formulated

\[
\min f(y_k) + \nabla f(y_k)^T d + \frac{1}{2} d^T \nabla^2 L(y_k) d \quad d \in \mathbb{R}^n
\]

s.t. \[ g_i(y_k) + \nabla g_i(y_k)^T d \leq 0 \quad i = 1, \ldots, m \quad (2.6) \]

\[ h_i(y_k) + \nabla h_i(y_k)^T d = 0 \quad i = 1, \ldots, l \]

The objective function in (2.6) is the second-order approximation of the Lagrangian of the original problem at point \( y_k \). Constraints in the quadratic problem are the linearized constraints at point \( y_k \). \( \nabla^2 L(y_k) \) in (2.6) is the usual Hessian of the Lagrangian of the problem (2.5) at \( y_k \), defined as

\[
\nabla^2 L(y_k) = \nabla^2 f(y_k) + \sum_{i=1}^{m} u_{ki} \nabla^2 g_i(y_k) + \sum_{i=1}^{l} v_{ki} \nabla^2 h_i(y_k). \quad (2.7)
\]

After solving the problem (2.6), with some \( d_k \) and Lagrange multipliers \( u_{k+1} \) and \( v_{k+1} \), we set \( y_{k+1} = y_k + d \) and solve the new QP with \( (y_{k+1}, u_{k+1}, v_{k+1}) \). The algorithm is continued this way until at some point QP is solved with \( d = 0 \). This point, \( y^* \), is then a point satisfying KKT conditions for problem (2.5) with Lagrange multipliers \( \bar{u} \) and \( \bar{v} \). It can be shown that if \( (y, u, v) \) is a regular KKT point satisfying sufficient conditions and if the iterative process is started at \( (y_k, u_k, v_k) \), a point close enough to \( (y^*, \bar{u}, \bar{v}) \), the algorithm converges quadratically to \( (y^*, \bar{u}, \bar{v}) \) (for convergence rate analysis, see [1]).

In many cases, as with the NPSOL software used in this study, the exact value of \( \nabla^2 L(y_k) \) is not used as it would require second-order derivatives to be calculated for both the objective function and the constraints. In addition, the Hessian is not necessarily positive definite and thus \( d \) might not be a descent direction. To alleviate these issues, a quasi-Newton positive-definite approximation \( B_k \) of \( \nabla^2 L(y_k) \) is used instead. \( B_k \) approximation is updated with BFGS update (see for example [1], [3]), defined as

\[
B_{k+1} = B_k + \frac{q_k q_k^T}{q_k^T p_k} - \frac{B_k p_k p_k^T B_k}{p_k^T B_k p_k}, \quad (2.8)
\]

where \( p_k \) and \( q_k \) are the changes in the vector of variables and in the gradient of
the Lagrangian

\[ p_k = y_{k+1} - y_k, \quad q_k = \nabla L(y_{k+1}) - \nabla L(y_k). \]

With the approximation (2.8), the convergence of the algorithm is superlinear under the necessary conditions and in the vicinity of the solution. To achieve global convergence, the unit step size in the direction \( d \) is abandoned and the new iterate \( y_{k+1} \) is determined with \( y_{k+1} = y_k + \sigma d \), where the nonnegative multiplier \( \sigma \) is chosen so that it produces sufficient decrease in the augmented Lagrangian merit function [3].

The different variables have different magnitudes for their numerical values, which might lead to significant numerical errors in the optimization process. To remedy this, each of the variables and the objective function are scaled so that their values would normally be on the \([0, 1]\) range. Before the optimization, the scaling of each variable is performed by dividing it by a variable-dependent scalar \( \beta_i \). The scaled variables \( \bar{y}_i = y_i / \beta_i \) are then used in the optimization, and the resulting values of \( y \) are descaled back to actual values with \( y_i = \beta_i \bar{y}_i \) after the optimization. This scaling naturally also needs to be taken into account in the constraints and in the derivatives of the objective function.

### 2.4 Determination of End State

The final time \( t_f \) is free in our applications and therefore needs to be estimated for each step. In the missile evasion problem, the optimization ends when the distance between the missile and the evading fighter aircraft is equal to some predetermined distance, \( d_f \). The final time must then satisfy the condition \( d(t_f) = d_f \). If it can reliably estimated that \( t_f \in T_k \) for some \( T_k \), it is known that \( T_k \) is the final optimization interval.

We can calculate the distance \( d \) for two successive discretization points \( t_p \) and \( t_{p-1} \) and approximate the time derivative of the distance as

\[ \delta \dot{d} = \frac{\Delta d}{\Delta t} = \frac{d_p - d_{p-1}}{t_p - t_{p-1}}. \]
and use $\delta d$ to make a linear approximation of $d(t)$ at $t_p$. The final time $\hat{t}_f$ such that $d(\hat{t}_f) = d_f$ can then be estimated as

$$\hat{t}_f = t_p + \frac{d_f - d_p}{\delta d}.$$  \hfill (2.9)

Another alternative is to extrapolate the trajectories of the aircraft and missile linearly to determine the future distance. The distance between the aircraft and the missile is

$$d(t) = \sqrt{(x_a(t) - x_m(t))^2 + (y_a(t) - y_m(t))^2 + (h_a(t) - h_m(t))^2},$$

where $x_a$, $x_m$, $y_a$, $y_m$, $h_a$, and $h_m$, are the $x$ and $y$-coordinates and the altitude of the aircraft and the missile, respectively. By approximating that the derivatives of the coordinates are constant, i.e. the future trajectories are linear, we can set $d(t)^2 = d_f^2$ and solve the resulting quadratic equation for $\hat{t}_f$

$$\hat{t}_f = t_p + \frac{-b - \sqrt{b^2 - 4ac}}{2a},$$ \hfill (2.10)

\[
\begin{align*}
    a &= \left(\Delta v_x\right)^2 + \left(\Delta v_y\right)^2 + \left(\Delta v_h\right)^2 \\
    b &= 2\left(\Delta x v_x + \Delta y v_y + \Delta h v_h\right) \\
    c &= \left(\Delta x\right)^2 + \left(\Delta y\right)^2 + \left(\Delta h\right)^2.
\end{align*}
\]

$\Delta x$ is the difference in $x$-coordinate at time $t_p$ and similarly for $\Delta y$ and $\Delta h$. $v_x$, $v_y$, and $v_h$ are the constant derivatives of the coordinates, i.e. velocity components. It must be noted that if $b^2 - 4ac < 0$, the equation (2.10) has no solution. In this case, the flight paths can be assumed not to reach the end distance in the examination interval.

Fig. 2.2 represents an example of the two methods compared to actual development of the distance. Here, $d_f = 300$ m. The approximation made with the time derivative of the distance is a straight line with a slope equal to the magnitude of the derivative. The linear trajectories estimation shows a convex function with a minimum point in time where the distance between the aircraft and the missile
Figure 2.2: Accuracy of the distance approximations with linear trajectories and distance gradient.

is the shortest if both continue on rectilinear paths. In the numerical examples, we will use the distance gradient method to estimate the final distance in each step, since the development of the distance according to it seems to more closely follow the actual distance, which decreases almost linearly in time. In addition, the distance gradient is more easily calculated.
3 Example Problem

Here we will very briefly review the dynamics of the problem that will be our application of the moving horizon control method in this study. The model represents a generic fighter aircraft trying to evade a medium-range air-to-air missile that is guided by proportional navigation (PN). The missile evasion is successful if the missile’s seeker can not follow the aircraft at some point in the duration of the flight. However, during early phases of the missile’s flight, the aircraft and the missile are far away from each other and therefore the aircraft is nearly straight ahead of the missile. The shorter the distance between the aircraft and the missile, the greater effect the maneuvers of the aircraft have on the missiles ability to follow the aircraft. Therefore, the objective function to be maximized is the final angle between the missile’s seeker head and the missile’s centerline, i.e. the line of sight (LOS) angle.

For a more rigorous treatment of the dynamics of the flight paths and aircraft trajectory optimization, see [6], [8], or [9]. The similar problem to the one presented here, was solved with open-loop optimization in [4].
3 EXAMPLE PROBLEM

3.1 Model Dynamics

The aircraft is a three-dimensional point-mass object with state equations

\[ \dot{x}_a = v_a \cos \gamma_a \cos \chi_a \quad (3.1) \]
\[ \dot{y}_a = v_a \cos \gamma_a \sin \chi_a \quad (3.2) \]
\[ \dot{h}_a = v_a \sin \gamma_a \quad (3.3) \]
\[ \dot{\gamma}_a = \frac{1}{m_a v_a} \{ \{ L(\alpha, h_a, v_a, M(h_a, v_a)) + \eta T_{\text{max}}(h_a, M(h_a, v_a)) \sin \alpha \} \cos \mu - m_a g \cos \gamma_a \} \quad (3.4) \]
\[ \dot{\chi}_a = \frac{1}{m_a v_a \cos \gamma_a} \{ L(\alpha, h_a, v_a, M(h_a, v_a)) + \eta T_{\text{max}}(h_a, M(h_a, v_a)) \sin \alpha \} \sin \mu \quad (3.5) \]
\[ \dot{v}_a = \frac{1}{m_a} \{ \eta T_{\text{max}}(h_a, M(h_a, v_a)) \cos \alpha - D(\alpha, h_a, v_a, M(h_a, v_a)) \} - g \sin \gamma_a. \quad (3.6) \]

The corresponding state equations for the point-mass missile are

\[ \dot{x}_m = v_m \cos \gamma_m \cos \chi_m \quad (3.7) \]
\[ \dot{y}_m = v_m \cos \gamma_m \sin \chi_m \quad (3.8) \]
\[ \dot{h}_m = v_m \sin \gamma_m \quad (3.9) \]
\[ \dot{\gamma}_m = \frac{a_1 - g \cos \gamma_m}{v_m} \quad (3.10) \]
\[ \dot{\chi}_m = \frac{a_2}{v_m \cos \gamma_m} \quad (3.11) \]
\[ \dot{v}_m = \frac{1}{m_m(t)} \{ T(t) - D(a, h_m, v_m) \} - g \sin \gamma_m \quad (3.12) \]
\[ \dot{a}_1 = \frac{a_{1c} - a_1}{\tau} \quad (3.13) \]
\[ \dot{a}_2 = \frac{a_{2c} - a_2}{\tau}. \quad (3.14) \]

\(x, y, h\) represent the \(x\) and \(y\)-coordinates and the altitude, as already mentioned earlier. \(\gamma\) and \(\chi\) are the flight path and heading angle of the missile and the aircraft. The flight path angle is the angle between the velocity vector and the \((x, y)\)-plane. The heading angle is the angle between the \(x\)-axis and the projection.
of the velocity vector on the \((x, y)\)-plane. \(v\) represents the velocity. \(a_1\) and \(a_2\) are the vertical and horizontal accelerations of the missile, which are determined by proportional navigation guidance. The control vector \(u = [\alpha, \mu, \eta]^T\) consists of the angle of attack, the bank angle, and the throttle setting. The aircraft system has the following constraints

\[
\begin{align*}
-\alpha & \leq 0 \\
n(\alpha, h_a, v_a) - N_{\text{max}} & \leq 0 \\
C_L(\alpha, M(h_a, v_a)) - C_{L,\text{max}}(M(h_a, v_a)) & \leq 0 \\
|\dot{\mu} - \chi_a \sin \gamma_a| - K_{\text{max}}(h_a, M(h_a, v_a), \alpha) & \leq 0 \\
|\dot{\alpha} + \dot{\gamma}_a \cos \mu + \dot{\chi}_a \cos \gamma_a \sin \mu| - P_{\text{max}} & \leq 0 \\
|h_{\text{min}} - h_a| & \leq 0 \\
q(h_a, v_a) - q_{\text{max}} & \leq 0.
\end{align*}
\]

These are the angle of attack limit, the load factor limit, lift coefficient limit, the bank angle and the angle of attack rate limits, altitude limit, and velocity limit. For description of these, see [4].

### 3.2 Problem Formulation

The objective function is the LOS angle of the missile. For numerical reasons, we calculate the cosine of LOS angle \(\lambda\) rather than the LOS angle itself. This can be calculated at any time instant \(t\) as

\[
\cos \lambda = \frac{v_{m,\alpha} \cdot r}{\|v_{m,\alpha}\| \|r\|}
= \left( \cos(\gamma_m + \alpha_1) \cos(\chi_m + \alpha_2)(x_a - x_m) + \
\cos(\gamma_m + \alpha_1) \sin(\chi_m + \alpha_2)(y_a - y_m) + \
\sin(\gamma_m + \alpha_1)(h_a - h_m) \right) / \sqrt{(x_a - x_m)^2 + (y_a - y_m)^2 + (h_a - h_m)^2}.
\]
3 EXAMPLE PROBLEM

Here, \( v_{m,a} \) is the missile centerline vector, \( r \) is the position vector of the aircraft, seen from the missile. \( \alpha_1 \) and \( \alpha_2 \) are the vertical and horizontal components of the missile’s angle of attack.

To solve the problem numerically, the continuous variables need to be discretized. With the moving horizon control, the discretization need only be performed for one interval at a time. Therefore, for each subproblem in interval \( T_p = [t_p, t_p + n\Delta t] \), the state vector \( \mathbf{x} = [h_a, \gamma_a, \chi_a, v_a, x_m, y_m, h_m, \gamma_m, \chi_m, v_m, a_1, a_2]^T \) is discretized with \( \Delta t \) steps. The control vector \( \mathbf{u} = [\alpha, \mu, \eta]^T \) is discretized with finer accuracy. For each single discretization step in the variables, \([t_p, t_p + \Delta t], n_c \) steps of discrete controls are included, i.e., the control vectors for each subinterval \([t_p, t_p + \Delta t]\), are \( \mathbf{u}_1^p, \mathbf{u}_2^p, \ldots, \mathbf{u}_{n_c}^p \). So, for one moving horizon subproblem, the state variables are \( x_1, x_2, \ldots, x_n \), and the controls are \( (\mathbf{u}_1^0, \mathbf{u}_2^0, \ldots, \mathbf{u}_m^0), \ldots, (\mathbf{u}_1^{n-1}, \mathbf{u}_2^{n-1}, \ldots, \mathbf{u}_m^{n-1}) \), \( \mathbf{u}_1^p \). By combining these together, we get the vector \( \mathbf{y} \) of decision variables. Thus, for each optimization problem in a single interval \( T_p \), the state variables to be optimized are

\[
\mathbf{y} = \begin{bmatrix}
    h_1^0 & h_2^0 & \cdots & h_n^0 \\
    \gamma_1^0 & \gamma_2^0 & \cdots & \gamma_n^0 \\
    \vdots & \end{bmatrix} \begin{bmatrix}
    a_1^1 & a_2^1 & \cdots & a_n^1 \\
    a_1^2 & a_2^2 & \cdots & a_n^2 \\
    \alpha_1^0 & \cdots & \alpha_{n_c}^0 & \alpha_1^1 & \cdots & \alpha_{n_c}^1 & \cdots & \alpha_1^{n-1} & \cdots & \alpha_{n_c}^{n-1} & \alpha_1^n \\
    \mu_1^0 & \cdots & \mu_{n_c}^0 & \mu_1^1 & \cdots & \mu_{n_c}^1 & \cdots & \mu_1^{n-1} & \cdots & \mu_{n_c}^{n-1} & \mu_1^n \\
    \eta_1^0 & \cdots & \eta_{n_c}^0 & \eta_1^1 & \cdots & \eta_{n_c}^1 & \cdots & \eta_1^{n-1} & \cdots & \eta_{n_c}^{n-1} & \eta_1^n \\
    t_f & \end{bmatrix}^T.
\] (3.23)

The constraints (3.15)- (3.21) are discretized in a similar fashion. This yields a nonlinear programming problem in standard form, which is then solved with the SQP method introduced in Section 2.3.
4 Numerical Results

The moving horizon control algorithm and the model dynamics were implemented on Fortran 77/90 language. For the numerical optimization, we used NPSOL 5.0 optimization package for Fortran (see [3]). NPSOL uses SQP algorithm to optimize nonlinear problems. The numerical computations were performed on a 1.8 GHz Pentium M computer. We first examined how well the MHC method solves the introduced problem as such. Then, we compared the results given by the MHC in comparison to traditional open-loop optimization, where the initial solution was generated with genetic algorithm and then discretized and optimized with SQP.

4.1 Results of the MHC method

We examined different trajectories resulting from different initial states. One such trajectory is depicted in Fig. 4.1. The figure presents the trajectories of the aircraft and the missile, as well as the projections of those on the \((x, y)\)-, and \((x, h)\)-planes. In the illustrated scenario, the missile is launched head-on towards the aircraft. The initial distance between the missile and the aircraft was 20 km, the altitude of both was 10 km and the final distance \(d_f\) was 1000 meters. The discretization parameters used were \(\Delta t = 1\) s, \(n = 8\), and \(n_c = 2\). The final time was roughly 18 seconds.

The results show a downward-sloping trajectory, as in [4]. By turning downward, the aircraft obtains more velocity, which enables faster maneuvers and therefore greater angles between the missile’s nose and the direction vector towards the aircraft. Also, the farther the aircraft can get from the missile, the less airspeed the missile has and thus the more it relatively needs to turn in order to reach the fighter aircraft. It can be noted that the most important evasion takes place during the final seconds. During the initial part of the trajectory, the flight path of the plane is quite stable. The aircraft flips inverted and turns downward somewhat late in the trajectory. The movement happens mainly in the \((x, h)\)-plane. In the \(y\)-direction, the path is almost a straight line.
Figure 4.1: Example trajectory generated by moving horizon control algorithm. The blue ribbon represents the aircraft and the twists in the ribbon illustrate the aircraft bank angle.

More generally, it was noticed that the moving horizon optimization does not alter the initial solution given to it in all cases, although a better solution would be available, as is seen in the next section. This of course depends on the type of the initial solution given to the optimizer. In our algorithm, the initial solution of the new interval part was based on integrating the system forward from the last final state with the final controls. This might not be the best possible way to generate the initial guess, but was used because of its simplicity. In the later parts of the optimization, the used initial guess seemed to work better. This can also be partly because of the nature of the problem, since in this problem, the main effect of the controls shows up quite late in the total optimization interval. Also the optimization horizon naturally affects the quality of the solution, but extending
the optimization interval longer in to the future also increases the time required
to compute each part of the trajectory.

4.2 Comparison of Methods

Next, the results given by the MHC method were compared against the open-loop
optimization. We compare the trajectory represented in Fig. 4.1 with open-loop
results with a similar initial state. For the open-loop method, a genetic algorithm
was used to generate the initial solution to the problem. The resulting trajectories
for both methods are illustrated in Figs. 4.2 and 4.3.

The both trajectories are qualitatively somewhat similar in nature. Turning down-
ward helps the aircraft to gain airspeed and evade the missile for a longer time
while the missile uses up its fuel and begins to lose airspeed. It is noticeable that
neither one of the methods much utilize the $y$-direction in the evasion. In these
types of problems, the examination could be simplified by confining to $(x, h)$-plane.
This is naturally not the case if the initial directions of the aircraft and the missile
are not head-on.

The maneuvers given by the open-loop optimization however seem to be more
aggressive and begin earlier than those given by the moving horizon control. While
the aircraft slowly turns slightly downward in the MHC solution, the open-loop
solution is more dramatic, turning straight down right from the beginning of the
optimization. During the final seconds, the optimal solution also turns slightly to
the left, while the MHC solution continues almost straight ahead.

The differences in trajectories in Figs. 4.2 and 4.3 also affect the achievable final
line of sight angle. The maximum achieved angle naturally depends on the final
distance, as the $\lambda(t)$ rises sharply during the final seconds of the flight trajectory.
$\lambda(t)$ for the MHC and open-loop methods is illustrated in Fig. 4.4. The reached
final value for the LOS angle $\lambda(t_f)$ is 23.8 degrees for the moving horizon control
and 35.3 for the open-loop optimization. The optimal value given by the open-
loop optimizer is thus considerably better than the LOS angle achieved with the
MHC. The slow reaction of the moving horizon method in the early phase shows
Figure 4.2: Aircraft and missile trajectories in the \((x, y)\)-plane with full optimization and moving horizon control.

Figure 4.3: Aircraft and missile trajectories in the \((x, h)\)-plane with full optimization and moving horizon control.
Figure 4.4: Comparison of the LOS angle $\lambda(t)$ between the full optimization and the moving horizon control.

up later in the flight where the aircraft ends up closer to the missile and with smaller airspeed than with the open-loop solution.

It is noticeable, however, that there is also a considerable difference in the computation time used by the two methods. The computation of the optimal open-loop solution took roughly 10 minutes for the genetic algorithm and the final optimization, while the moving horizon control was finished in about 30 seconds. Therefore, if the time used on the solution or the computational burden of the problem are critical factors, the MHC has an advantage over the genetic algorithm and open-loop solver combination in that respect.
5 Summary and Conclusions

In this study, we have applied moving horizon control approach to an aircraft trajectory optimization problem. We presented an outline of the MHC algorithm and discussed how the moving horizon could be implemented. We constructed the MHC steps as partly overlapping, so that we can use the solution from the previous interval as an initial solution to the major part of the next interval. These overlapping steps would then be continuously solved until it could be seen that the optimization would end at the following step. We discussed how the initial solution to the new parts of the optimization horizon could be generated. We used the final controls from the previous interval to integrate the missing part of the initial solution. The SQP algorithm as a part of the moving horizon control was described and the determination of the end state to the problem was discussed. We applied the moving horizon control to the missile evasion problem. With discretization, we transformed the MHC subproblems to regular nonlinear programming problems. The numerical results from the MHC algorithm showed that the method gives qualitatively similar results to the optimal open-loop method, but the MHC optimization in the early phase might not take the whole problem satisfactory into consideration.

The most serious problem with the algorithm in the study was the nonlinear solver’s tendency to converge to a local optimum that was very near to the generated initial solution. This lead to the fact that very little changes were made to the trajectory in the early phases of the MHC. To remedy this, we could of course use more complicated initial solutions or have the initial controls randomly selected. However, since we were interested in generating quick and simple initial solutions that would not have a stochastic nature, this approach is rather questionable. Another solution to this problem would be to extend the examination horizon or to change the nonlinear optimization algorithm or the specific implementation of it.

Compared to the moving horizon control method represented in [4], the MHC method presented here is considerably faster and does not suffer from same kind of curse of dimensionality problems as the method used in the earlier study. These problems were caused by the discrete alternative values for the controls and the
recursive nature of the algorithm. Method presented here has also the advantage that the final result is not as sensitive to the parameters of the algorithm.

The natural use for the moving horizon control is in the generation of an initial solution to the open-loop solver. We saw that the MHC was considerably faster than the genetic algorithm. As stated before, moving horizon control is also a suitable method when the optimization is performed in a real-time fashion. According to the results, the time requirement for MHC is much smaller than that for a full optimization. MHC can therefore used in time-critical applications or situations where the full open-loop optimization is not a viable option due to its computational complexity. Furthermore, the advantage of MHC its ability to incorporate new information to the optimization as the horizon moves. These types of algorithms can also be used in other types of applications, such as collision avoidance (see [7]).
References


REFERENCES
