Unmanned Helicopter Waypoint Trajectory Tracking Using Model Predictive Control

C. L. Castillo*, W. Moreno*, K. P. Valavanis**
* Electrical Engineering
**Computer Science and Engineering
Unmanned Systems Laboratory
USF, Tampa, FL 33620, USA

Abstract—A Model Predictive Control Based Trajectory Tracking (MPCTT) system for small unmanned helicopters is presented. A linear model predictive controller is used to take advantage of the fast algorithms available to solve convex optimization problems. The proposed MPCTT system is compared with a velocity tracking and a position tracking system implemented with classical PIDs from previous research work. Obtained simulation results demonstrate the superiority of the proposed MPCTT approach, generating a substantially less control effort in order to track waypoint trajectories. The unmanned helicopter used for the proposed MPCTT system is considered as a (linearized) linear state-space model obtained for hovering and slow motion; however, as shown, the MPCTT is robust enough to perform trajectory tracking under a cruise flight mode, too.

Keywords – unmanned helicopter, model predictive control, waypoint navigation, trajectory tracking.

I. INTRODUCTION

Research in the area of Unmanned Aerial Vehicles (UAV) has seen recently unprecedented levels of growth, even though it dates back to as early as World War I where an unmanned aircraft was considered the "aerial equivalent to the naval torpedo" [1]. Currently, most of the fixed wing UAV related military applications are in the area of intelligence, surveillance and reconnaissance (ISR) [1, 2], performing missions at high altitudes to avoid detection and threats from the enemy.

However, as part of the Future Combat Systems (FCS) initiative, formerly known as Future Ground Combat Systems program [3], it has been recommended that several types of vertical take-off and landing (VTOL) aircrafts be developed and used. Such VTOL vehicles will provide reconnaissance, surveillance and target acquisition assistance to the ground troops [4], offering major advantages over fixed-wing unmanned aircrafts, like flying in very low altitudes, taking off and landing almost everywhere.

Besides military applications, there are civil and commercial applications, too, such as search and rescue, traffic monitoring, demining, forest fire detection, border patrol, filming industry, dam inspections, etc. For these applications, miniature or small-scale man portable unmanned helicopters offer additional advantages including easy transportation and cost effectiveness. If an appropriate and robust control system may be designed to navigate such small helicopters, then they can perform aggressive and non-aggressive flights and maneuvers for a wide range of applications.

Helicopters are multiple input multiple output (MIMO), nonlinear, underactuated, unstable and highly coupled systems [5, 6]. Because of these characteristics, design of a control system is a rather challenging task. Small-scale helicopters present unique dynamics behavior because of typical large thrust-to-weight ratios and peak roll and pitch rate sensitivities of about 200 deg/s [2].

The most widely known and accepted research results by the community are from the "Software Enabled Control" (SEC) program sponsored by the Defense Advanced Research Projects Agency (DARPA). This research has pushed the state of the art of helicopter control to higher levels implementing advanced control techniques and sophisticated software platforms [7], including hardware-in-the-loop and software-in-the-loop validation and verification. Receding horizon control (RHC) was one of the advanced control techniques implemented and improved during the five-year program [8]. RHC is also called Model-based Predictive Control (MPC) or simply Predictive Control (PC).

The viability of using model predictive control for tracking trajectories in rotorcraft based UAV has received some attention in the past. Reference [9] presents a RHC approach for trajectory generation and flight control. Its approach is limited to differentially flat model needed for the generation of feed forward control trajectory. Its model is a planar, Cartesian (X-Y), 3-DOF model. Reference [10, 11] presents a combination of a neural network (NN) feedback controller and a state-dependent Riccati equation (SDRE) controller. The NN is optimized within a MPC framework. Some tracking trajectories with good results were presented but the computational effort is very high and the method is unlikely to be directly employed as on-line controller for UAVs [12]. Reference [12] presents a nonlinear model predictive tracking control (NMPTC). It outperforms a multi-loop proportional-derivative (MPLD) controller with which NMPTC is compared. This NMPTC approach shows robustness to parameter uncertainty.

This paper focuses on the design and implementation of linear model predictive controllers for waypoint trajectory tracking of small-scale unmanned helicopters. The controller developed shows a good robustness to parameter uncertainty. This MPC is based in a linear model considering input constraints. States constraints are also easily incorporated. The use of a linear model allows reducing the computational burden associated with the
nonconvex, nonlinear programming problem, which is generated when the quadratic components of the cost functions are substituted by the nonlinear model.

The remaining parts of this paper are divided as follows: Section II describes the helicopter model for control design. Section III presents the basic concepts of model predictive control and the controller implementation. Section IV presents the trajectory tracking system developed using the MPC. Section V presents the simulation results of tracking two trajectories and brief preliminary timing results. Section VI concludes the paper.

II. HELICOPTER MODEL

The reference point is a 13th order model in state space, derived for an R-50 helicopter, given by (1). This model is based on the stabilities derivatives model, a parameterized linearized form of equations of motions where external forces and moments are represented through the product of derivatives and the rigid-body vehicle’s states and control inputs [6]. It extends the stabilities derivatives model to account for coupled rotor-fuselage dynamics and control inputs [6]. It extends the stabilities derivatives model, a parameterized linearized form of equations of motions where external forces and moments are represented through the product of derivatives and the rigid-body vehicle’s states and control inputs [6]. The equations used in the original model to relate Euler rates (φ, θ) and rotational velocities (p, q) in the body-frame are simplifications of the exact relationship between Euler rates and rotational velocities and they are shown in (4):

\[
\dot{\phi} = p \\
\dot{\theta} = q
\]

\[
B = \begin{bmatrix}
A_{uu} & A_{uv} & 0 & 0 \\
A_{vu} & A_{vv} & 0 & 0 \\
B_{u} & B_{v} & 0 & 0 \\
0 & 0 & Z_{uu} & Z_{uv} \\
0 & 0 & N_{vud} & N_{vvd} \\
0 & 0 & 0 & 0 \\
0 & 0 & C_{iu} & 0 \\
D_{u} & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & Y_{uu} & 0 \\
0 & 0 & Y_{uv} & 0 \\
0 & 0 & 0 & M_{uu} \\
0 & 0 & 0 & M_{uv} \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
\tau_f & \tau_r \\
\tau_f & \tau_r \\
\tau_f & \tau_r \\
\tau_f & \tau_r \\
\tau_f & \tau_r \\
\end{bmatrix}
\]

The complete expression for these relationships is presented in equation (5) as follows:

\[
\dot{x} = Ax + Bu
\]

\[
A = \begin{bmatrix}
X_s & 0 & 0 & 0 & 0 & g & X_s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & Y_r & 0 & 0 & g & 0 & 0 & Y_r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
L_s & L'_s & 0 & 0 & 0 & 0 & L_s & L'_s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
M_s & M'_s & 0 & 0 & 0 & 0 & M_s & M'_s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

A. Nonlinearities included in the model

The model presented in [6] is used here with some modifications. The equations used in the original model to relate Euler rates (φ, θ) and rotational velocities (p, q) in

\[
\begin{array}{c}
\text{TABLE 1. MODEL STATES} \\
\text{Part} & \text{State} & \text{Description} \\
\hline
\text{Fuselage Linear Motion} & u & X-velocity \\
\text{Fuselage Angular Motion} & p & Roll Angular Rate \\
\text{Rotor Tip-Path-Plane} & q & Pitch Angular Rate \\
\text{Stabilizer Bar Tip-Path-Plane} & r & Yaw Angular Rate \\
\text{Augmented Yaw Dynamics} & r_{th} & Yaw Rate Gyro Feedback \\
\text{Roll} & \delta & \text{Roll Euler Angle} \\
\text{Pitch} & \theta & \text{Pitch Euler Angle}
\end{array}
\]

\[
\begin{array}{c}
\delta_{col} & \text{Collective Control Input} & \text{Dimensionless [-1,1]} \\
\end{array}
\]

\[
\begin{array}{c}
\delta_{ped} & \text{Pedal Control Input} & \text{Dimensionless [-1,1]} \\
\end{array}
\]

\[
\begin{array}{c}
\delta_{lon} & \text{Longitudinal Cyclic Deflection} & \text{Dimensionless [-1,1]} \\
\end{array}
\]

\[
\begin{array}{c}
\delta_{lat} & \text{Lateral Cyclic Deflection} & \text{Dimensionless [-1,1]} \\
\end{array}
\]

\[
\begin{array}{c}
\delta_{col} & \text{Collective Control Input} & \text{Dimensionless [-1,1]} \\
\end{array}
\]
It may be observed that Euler angles are assumed to be very small or zero, a justifiable assumption for non-aggressive flights that allows for model simplification using small angle approximation in order to design flight control systems [13, 14].

However, in this case, the complete relationship shown in (5) is used and implemented because it reproduces the helicopter dynamics more accurately than the simplified representation. The drawback is that by using this complete expression, nonlinearities in the helicopter model are introduced that must be taken into account during the controller design phase.

III. MODEL PREDICTIVE CONTROLLER DESIGN

A. MPC Fundamentals

MPC, used in industry for more than 25 years, may be considered as the most general approach to tackle a control problem in the time domain [15]. MPC refers to a set of control strategies that are based on the same basic ideas or concepts [15] as follows:

- Explicit use of a plant model to predict the behavior (states and outputs) of the plant at future time instants;
- Computation of a control sequence minimizing a cost or objective function that takes into account the output/states errors and control effort;
- Receding horizon strategy, in which at each instant the predicted behavior is displaced towards the future, and only the first value of the control sequence calculated at each instant is applied.

Main advantages for using/implementing MPC, also called Receding Horizon Predictive Control (RHPC), are:

- Ability to support constrains of variables associated with the control problem under study such as input, output or states variables;
- Its basic formulation may be extended to multivariable plants with almost no modification;
- Intrinsic compensation for dead time and no minimum phase dynamics;
- Ability to use future values of references when they are available, allowing MPC to improve performance in navigation such as waypoint trajectory tracking.

B. Implementation using Model Predictive Control Toolbox

The MPC has been designed using the MATLAB Model Predictive Control Toolbox (MPCTB), which has a graphical user interface (GUI) that allows for an interactive design of controllers.

The MPCTB uses a linear model for prediction and optimization [16]. Considering that: \( x(k) \) is the \( n_x \)-dimensional state vector, \( u(k) \) is the \( n_u \)-dimensional vector of manipulated variables (MV), \( y(k) \) is the \( n_y \)-dimensional vector of measured disturbances (MD), \( d(k) \) is the \( n_d \)-dimensional vector of unmeasured disturbances (UD), \( y_m(k) \) is the vector of measured outputs (MO), \( y_v(k) \) is the vector of unmeasured outputs (UO) [16], the relevant equations are shown in (6):

\[
\begin{align*}
x(k+1) &= Ax(k) + B_u u(k) + B_v y(k) + B_d d(k) \\
y_m(k) &= C_n x(k) + D_m y(k) + D_{du} d(k) \\
y_v(k) &= C_v x(k) + D_v y(k) + D_{dv} d(k) + D_{uv} u(k)
\end{align*}
\]

The overall output vector \( y(k) \) is composed of \( y_m(k) \) and \( y_v(k) \) [16].

The cost or objective function used in the optimization is given by (7):

\[
J = \sum_{i=0}^{n_x-1} \sum_{j=1}^{n_u} w_{ij} \left( y_j(k+i+1) - r_j(k+i+1) \right)^2 + \\
+ \sum_{j=1}^{n_u} w_{ij} \sum_{i=1}^{n_v(\text{current})} \Delta u_j(k+i)^2 + \\
+ \sum_{j=1}^{n_v(\text{current})} w_{ij} \sum_{i=1}^{n_v(\text{current})} \Delta y_j(k+i)^2 + \\
+ \sum_{j=1}^{n_v(\text{current})} w_{ij} \sum_{i=1}^{n_v(\text{current})} (y_j(k+i) - y_{\text{ref}}(k+i))^2 + \rho \epsilon^2
\]

where the subscript "(\( k+i \))" denotes the \( j \)-th component of a vector, "(\( k+i \))" denotes the value predicted for the time \( k+i \) based on the information available at time \( k \); \( r(k) \) is the current sample of the output reference, \( p \) is the prediction horizon, \( n_u \) is the number of inputs, \( n_v(\text{current}) \) are nonnegative weights for the corresponding variable. \( \Delta u_j(k) \) is the control effort at time \( k \).

In the case of existence of constrains, the optimization of the function \( J \) is subject to (8):

\[
\begin{align*}
u_{\text{min}}(i) &\leq u_j(k+i) \leq u_{\text{max}}(i) & s.t. \Delta u_k(j) &\leq u_{\text{max}}(i) + \Delta u_k(j) \\
\epsilon_k(j) &\leq \epsilon_k(j) & s.t. \Delta u_k(j) &\leq u_{\text{max}}(i) + \Delta u_k(j) \\
\end{align*}
\]

The minimum horizon prediction is always zero.

For the design under consideration, only constraints of the manipulated variables have been considered. The minimum value allowed in the inputs is -1, the maximum values is +1, the maximum down rate is -1, and the maximum up rate is +1. As mentioned above, states constrains can be easily incorporated.
The body-frame translational velocities and the heading were selected as the variable to be controlled for the MPC. The reason for this selection is explained in Section IV.

The design has been implemented using the GUI of the MPCT as follows:

The body-frame helicopter model previously described is first implemented in Simulink;

The complete relationship between Euler angle rates and body-frame rotational velocities has been used and implemented, and the model has been modified to use this relationship. Figure 1 shows the Simulink implementation of the body-frame linear helicopter model and the body to inertial transformation that includes (5).

The MPCTB uses a linear model for prediction and optimization, thus, the Simulink model is linearized using the Control and Estimation Tool.

The linearization produced a state-space model, which is controllable but unstable. Table 3 shows the modes of the linearized system. These modes are the same with the ones presented in [6, 14] for hovering /slow flight scenarios. Because linearization of the helicopter model was at the equilibrium point of \((u, v, w) = (0, 0, 0)\), equation (5) becomes essentially (4) due Euler angles are around zero.

<table>
<thead>
<tr>
<th>Modes</th>
<th>Eigenvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode 1</td>
<td>(\lambda_1 = 0.3061 \pm 0.6939i)</td>
</tr>
<tr>
<td>Mode 2</td>
<td>(\lambda_2 = -0.4007 \pm 0.0862i)</td>
</tr>
<tr>
<td>Mode 3</td>
<td>(\lambda_3 = -0.6078)</td>
</tr>
<tr>
<td>Mode 4</td>
<td>(\lambda_4 = -1.6977 \pm 8.1884i)</td>
</tr>
<tr>
<td>Mode 5</td>
<td>(\lambda_5 = -2.6605 \pm 11.5571i)</td>
</tr>
<tr>
<td>Mode 6</td>
<td>(\lambda_6 = -6.1981 \pm 8.1967i)</td>
</tr>
<tr>
<td>Mode 7</td>
<td>(\lambda_7 = -20.3125 \pm 4.7429i)</td>
</tr>
</tbody>
</table>

Using the linearized model obtained previously, the model predictive controller was tuned manually to obtain the responses shown in Figure 2. The sampling period or control interval [16] is 0.01 seconds, the prediction horizon is 100 and the control horizon is 10.

### IV. VELOCITY / POSITION WAYPOINT TRAJECTORY TRACKING ARCHITECTURES

Different names have been used to describe controller configurations used in flight control systems, such as multi-loop [17], cascade or nested [6] controllers. Typical architectures of a waypoint trajectory tracking system or flight control system may be broadly classified as Velocity Tracking Control and Position Tracking Control architectures as shown in Figure 3.

Trajectories to be tracked normally consist of a sequence of waypoints at specified inertial coordinates \((x, y, z)\), heading and time.

The implementation, based on classical control techniques, of an attitude (inner loop) controller consists of two or more SISO PID controllers. In [6], the attitude controller is formed only by the attitude angle (roll, pitch) loops. In [13, 14] the attitude controller is formed with the attitude angles, heading and height. In [17] Euler angles are used only. The body-frame velocity controller is also known as outer-loop controller for the case of the velocity tracking control configuration; it is called mid-loop controller [17] for the case of the position tracking controller configuration.

The inertial-frame position controller involves the inertial frame to body frame transformation to obtain the body-frame velocity set points needed for the mid-loop controller. In addition, the body-frame to inertial frame transformation is needed to obtain the actual inertial
position of the helicopter, see Figure 3b. The inertial frame to body frame transformation is presented in (9):

\[
T^b = \begin{bmatrix}
0 & c \theta \psi & -s \theta \\
s \psi & c \phi & s \phi \\
-c \phi & s \phi \psi & c \phi \theta
\end{bmatrix}
\]  \hspace{1cm} (9)

The transformation from body-frame to inertial frame is given by (10):

\[
T^i = (T^b)^T
\]  \hspace{1cm} (10)

The first type of tracking system, shown in Figure 3a, has the advantage of simplicity and the disadvantage that it cannot compensate errors in the inertial coordinates of the helicopter due to disturbances (like gust winds) and it is very susceptible to heading measurement errors.

The second type of tracking system shown in Figure 3b has the advantage of compensating for errors in the inertial position. The use of GPS is normally used to compensate inertial position measurement errors. The use of the coordinate transformation blocks, inserts nonlinearities in the inertial-frame position controller. These nonlinearities may be neglected for the case of non-aggressive flights [13, 14].

The above designed MPC is used in the trajectory tracking system shown in Figure 4. In this configuration, the MPC is implemented as a body-frame velocities controller. The multi input multi output (MIMO) MPC implements the functions of both attitude and body-frame velocity controllers shown in Figure 3. The waypoint trajectory tracking system has been implemented using Simulink.

**V. SIMULATION RESULTS**

The proposed MPCTT system has been tested on trajectories and compared with the tracking system used in [13, 14] (called the PID velocity tracking system), shown in Figure 3a. In addition, the proposed system is also compared with a waypoint trajectory tracking system like the one shown in Fig. 3b obtained through the use of the inner and outer loop controller developed in [13, 14], called the PID position tracking system. Next, the MPCTT system implemented is tested again two trajectories

A. Ascending Spiral Trajectory

This trajectory is used in [12] to test the ability of the controller to handle nonlinear kinematics as well as the multivariable coupling in system dynamics. It can be observed in the next five figures that the MPCTT performs very well compared to the PID velocity tracking system and it is better than the PID position tracking. Figure 10 to Figure 13, show that the variations or changes in control signals of the MPCTT are minimal compared to changes needed in the PID velocity and position controllers. The same trajectory was tracked using the model parameters for the cruise mode. This change of operational mode requires substantial changes in the parameters. The MPCTT was able to track the trajectories with almost no changes. The range of the parameter changes were from 0% to more than 100% for some parameters. This indicates that the MPC is robust to those parameter changes.
B. Double circle with constant height trajectory

Figure 14 to Figure 22 show again that the MPCTT outperforms substantially the PID velocity tracking control and marginally the PID position regarding with respect to error in the coordinates. However, again the MPCTT produces control signals that require substantially less control effort or changes.
C. Preliminary Timing Results

The issue of real-time implementation of the MPC model developed was studied using the Profile tool available in MATLAB/Simulink. Even though MATLAB does not provide a function that gives the number of floating point operations per second (FLOPS), the profile tool provides the CPU time needed to execute the model predictive control function implemented in Simulink as shown in Table 4.

<table>
<thead>
<tr>
<th>Trajectory</th>
<th>CPU time per function call</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascending</td>
<td>1.29 ms</td>
</tr>
<tr>
<td>Double Circle</td>
<td>0.8 ms</td>
</tr>
</tbody>
</table>

Figure 16. Comparison of the y-coordinate response

Figure 17. Comparison of the height response

Figure 18. Comparison of the heading response

Figure 19. Comparison of the lateral cyclic control signals

Figure 20. Comparison of the longitudinal cyclic control signals

Figure 21. Comparison of the pedal cyclic control signals

Figure 22. Comparison of the collective cyclic control signals
These profiles were obtained from an AMD Athlon 64 3200, clock of 2.2 GHz, 1 GB ram. Assuming that the CPU needed to compute the MPC function is going to be proportional to the clock speed. A CPU with a clock of 440 MHz would be able to run the function in 6.45ms and 4 ms respectively. These timing results, even though not precise, are encouraging.

VI. CONCLUSIONS

This paper has presented the design of a model predictive control based waypoint trajectory tracking system. Comparison of the results obtained shows that our system performs better than the other evaluated systems. The control effort needed to track trajectories using the system developed is substantially less. Robustness of the MPCTT was tested by modifying the model’s parameters in order to represent modeling errors, producing almost similar responses to the trajectories used. Preliminary timing results suggest that the real-time implementation is perfectly possible considering the speed of actual computers.

ACKNOWLEDGMENT

This work was also supported partially by two grants ARO W911NF-06-1-0069 and SPAWAR N00039-06-C-0062. We also wish to thank Miss Wendy Alvis for her collaboration in the revision of this paper.

REFERENCES