Proceedings of the 2002 IEEE International
Conference on Control Applications
September 18-20, 2002 • Glasgow, Scotland, U.K.

Application of predictive control to a toy helicopter

Jonas Balderud and David I. Wilson
Department of Electrical Engineering
Karlstad University, Sweden
jonas.balderud@kau.se

Keywords: Matlab/Simulink, optimal control, helicopter, MPC

Abstract

A toy helicopter makes for an impressive classroom control demonstration. However the challenge to obtain a robust, stable, multivariable controller is non-trivial given the nonlinear nature of the helicopter, the stochastic nature of the disturbances, and the performance limitations of PCs. This paper applies a model predictive controller using the Mathworks xPC target to obtain significantly improved results over that achievable using classical control algorithms.

1 Predictive control in an education context

In 1997, Karlstad University purchased, at not inconsiderable expense, a toy helicopter with two degrees of freedom and with three inputs as shown in Fig. 1. The intention was to use this bench-scale model in our automatic control education to supplement our existing collection of laboratory equipment. Our wish was for a multi-input/multi-output (possibly non-square) interacting laboratory plant that exhibited both stable and unstable modes, possessing time constants in the order of seconds, and most importantly, to be visually arresting. This latter feature in particular is due to the simple fact that the helicopter is practically uncontrollable under manual operation. The helicopter from Humusoft[1] satisfied most of these demands and we demonstrated it successfully in introductory control courses and during university open days.

However some debilitating drawbacks for this laboratory plant quickly became apparent such as severe stiction nonlinearities, and the difficulty in developing a white, or even grey-box model of the plant. These drawbacks are such that apart from the initial demonstration, we found that we could not use the helicopter productively until late in our advanced control course. This is, of course, poor utilisation of an expensive resource. Some of these issues are independent of the helicopter and are further discussed in [1].

Figure 1: The bench-scale 2DOF helicopter from Humusoft.

The helicopter challenged us in the MATLAB/SIMULINK environment using a modest 266MHz computer. While the Real-time Toolbox (also from Humusoft) performed adequately at modest sampling rates (0.1s and upwards), this was inadequate for the helicopter.

The remainder of the paper is as follows. Section 2 reviews state-space model predictive control with constraints and section 3 outlines the model development and identification of key parameters and the rational behind the implementation of MPC on a PC running a dedicated real-time operating system. Section 4 shows the performance of the MPC control scheme and section 5 finishes with some conclusions.

2 Model predictive control

Model Predictive Control (MPC) refers to a family of controllers that use a model to compute an input trajectory in order to optimize the future behaviour of the plant. The optimization is repeated each sample, as only the first future manipulated variable adjustment

[1] www.humusoft.cz

is actually implemented on the plant. MPC while perhaps an odd choice for an aerospace application, is very popular in the chemical processing industry due to its ability to handle saturation constraints, [2–4]. This paper also explores the possibility to apply MPC on a high-speed plant which would traditionally be controlled by a much simpler algorithm implemented in an embedded system.

The MPC algorithm is well known and is reviewed in [5]. For this application we use a state-space approach, [6], with a differenced input

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{B}\Delta\mathbf{u}(k) \quad (1)$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{x}(k) \quad (2)$$

where Δ is the difference operator defined as $1 - q^{-1}$, and q^{-1} denotes the backward shift operator. In the linear SISO case, the predicted output j samples in the future is given by,

$$y_p(k+j) = y_f(k+j) + \sum_{i=0}^{j-1} \mathbf{C}\mathbf{A}^{j-i-1}\mathbf{B}\Delta u(k+i) \quad (3)$$

where $y_f(k+j)$ is the so-called free response, which accounts for the initial conditions, at time k, and the input adjustments done in the past.

Over a prediction horizon, N_p, using a control horizon, N_c, the stacked predicted output vector, $\mathbf{y_p} = [y_p(k+1), \cdots, y_p(k+N_p)]^T$ is given by

$$\mathbf{y_p} = \mathbf{y_f} + \mathbf{H}\Delta\mathbf{u} \quad (4)$$

where the dynamic matrix $\mathbf{H}$ is defined as,

$$\mathbf{H} = \begin{bmatrix} h_{1,1} & \cdots & h_{1,N_c} \\ \vdots & \ddots & \vdots \\ h_{N_p,1} & \cdots & h_{N_p,N_c} \end{bmatrix} \quad (5)$$

with the scalar elements $h_{j,i}$ given by,

$$h_{j,i} = \begin{cases} \mathbf{C}\mathbf{A}^{j-i}\mathbf{B}, & j \geq i \\ 0, & j < i \end{cases} \quad (6)$$

The unconstrained cost function to be minimized is a weighted sum of output deviations from a desired trajectory, $\mathbf{r}$, and input trajectory,

$$J = (\mathbf{r} - \mathbf{y_f} - \mathbf{H}\Delta\mathbf{u})^T(\mathbf{r} - \mathbf{y_f} - \mathbf{H}\Delta\mathbf{u}) + \Delta\mathbf{u}^T\mathbf{\Lambda}\Delta\mathbf{u} \quad (7)$$

with solution

$$\Delta\mathbf{u} = \left(\mathbf{H}^T\mathbf{H} + \mathbf{\Lambda}\right)^{-1}\mathbf{H}^T(\mathbf{r} - \mathbf{y_f}) \quad (8)$$

Since the current state appears implicitly in the term $\mathbf{y_f}$ of Eqn. 8, a state observer is needed to complete the control algorithm. The state observer, such as a Kalman Filter, provides direct feedback into the control law, which is different from the somewhat ad-hoc solution used by the DMC, [7], algorithm.

This SISO control law is extended to the multivariable case by stacking the input and output vectors,

$$\Delta\mathbf{u} = [\Delta\mathbf{u}_1, \cdots, \Delta\mathbf{u}_n]^T, \; \mathbf{y_p} = \left[\mathbf{y_p}_1, \cdots, \mathbf{y_p}_n\right]^T \quad (9)$$

so now dynamic matrix becomes,

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,1} & \cdots & \mathbf{H}_{1,n} \\ \vdots & \ddots & \vdots \\ \mathbf{H}_{n,1} & \cdots & \mathbf{H}_{n,n} \end{bmatrix} \quad (10)$$

where each block matrix $\mathbf{H}_{i,j}$ predicts the effect of input j on the output i.

Owing to the significant nonlinearities of the helicopter, (refer Fig. 1), the accepted approach in such cases is to use a full nonlinear model to predict the free response, while using a linearised plant model to compute the control moves, [8]. The alternative of solving the full optimisation problem while avoiding the explicit linearisation step, is extremely computationally demanding, and rarely used, [9].

In addition to nonlinear dynamics, the helicopter is constrained in motor power (input), and unlike some competitor's products, it is constrained in both outputs rotations. Since adding constraints to the optimisation problem precludes the possibility of an analytical solution, most commercial MPC formulations use a quadratic programming approach. However in our case, we could not afford the computation time to solve a QP, so we simply saturated the inputs if the optimised solution violated the hard constraints. A minor algorithmic improvement could be to remove those inputs that violated the constraints from the decision vector, and repeat the now smaller optimisation problem. In practice, as shown in section 4, constraint violation was not a problem under typical operation.

3 Implementation of MPC on the helicopter

3.1 Helicopter model

The helicopter has two degrees of freedom, (elevation, azimuth), and three inputs (main and side rotors, and a moveable counter weight). Four states are needed to model the position and velocity of the body in the two axes,$(\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$, and two further states, (ω_1, ω_2), are needed to model the two motor dynamics giving a total of six, coupled, highly-nonlinear states. A semi-

rigorous model incorporating stiction,

$$\begin{aligned}
I_m\ddot{\theta}_1 =& K_1\omega_1^2 - (\beta_{11}|\omega_1| + \beta_{21})\dot{\theta}_1 \\
&- T_{c_1}\text{sign}(\dot{\theta}_1)\left(1 - \text{e}^{-(|\dot{\theta}_1|/\dot{\theta}_{1_0})}\right) \\
&- T_g \sin(\theta_1 + \alpha_1) - K_G\dot{\theta}_2\omega_1\cos(\theta_1) \\
&+ K_C\dot{\theta_2}^2\cos(\theta_1) \\
I_{r_1}\dot{\omega}_1 =& u_1 - a_1\omega_1^2 - b_1\omega_1 \\
\sin(\theta_1)I_{s_0}\ddot{\theta}_2 =& \sin(\theta_1)K_2|\omega_2|\omega_2 \\
&- (\beta_{12}|\omega_2| + \beta_{22})\dot{\theta}_2 \\
&- T_{c_2}\text{sign}(\dot{\theta}_2)\left(1 - \text{e}^{-(|\dot{\theta}_2|/\dot{\theta}_{2_0})}\right) \\
&- \sin(\theta_1 + \alpha_2) \\
&K_{r_1}\left(u_1 - K_{r_2}\left(a_1\omega_1^2 + b_1\omega_1\right)\right) \\
I_{r_2}\dot{\omega}_2 =& u_2 - a_2|\omega_2|\omega_2 - b_2\omega_2
\end{aligned}$$

was developed in [10] which also details the identification of the 23 plant parameters. The two outputs are the axis rotations θ_1 and θ_2. The 95% settling time for the elevation dynamics is around 3–6s while the motor dynamics are an order of magnitude faster, so a suitable sampling time is 0.05s giving prediction and control horizons of $N_p = 60$ and $N_c = 30$ respectively.

3.2 Implementation using the target hardware

As is typical in these undertakings, the simulations in MATLAB worked perfectly. However the controller computation time on a 1GHz machine was in the order of 0.08s, which clearly indicated that running under a more modest 266MHz machine, with the added load of servicing the real-time components in a timely manner was unlikely to suffice. Table 1 lists our implementation alternatives for an MPC controller in increasing order of development complexity. The Humusoft real-time environment (options 1–3) has both the disadvantage of requiring an excessive sample time, and an enforced one-step delay. In addition, the at times sporadic sampling timings played havoc with the model-predictive controllers whereas the PI type are relatively immune.

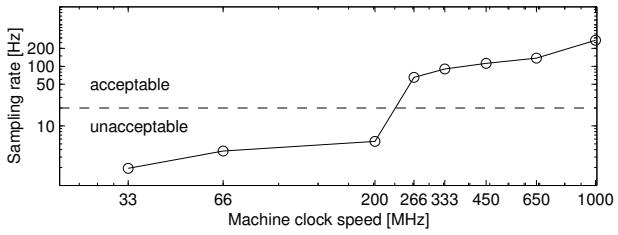

Figure 2: Maximum sampling rate using xPC as a function of clockspeed.

Implementing the controller on a dedicated PC running the xPC target real-time kernel solves the timing problems that plague real-time windows applications, but also considerably increases the development complexity and cost. This is partly offset by the minimal hardware demands as illustrated in Fig. 2 which gives the achievable sampling rate for the MPC controller given machine clock speed measured on a variety of PC hardware. The maximum achievable sampling rate is both a function of clock-speed and processor architecture which is why the value for the 200MHz machine is lower than expected. Of the next three options listed in Table 1, only the last is feasible given the limitations of the xPC development environment (e.g. MATLAB m s-functions and MATLAB matrix library routines are unsupported, limitations on the size of variables used, the computational overhead of SIMULINK passing large matrices between blocks etc).

The main algorithmic stumbling blocks for the MPC calculations are the re-linearisation of the nonlinear system, and the nonlinear integration step required for the prediction. These operations are not suited for the block diagram nature of SIMULINK, and even if attempted, are comparatively slow compared to C code. The disadvantage of C code is that it blurs the transparency of the application, an important consideration in an education environment. The streamlining of the computation required the following three steps: (1) employing simple explicit integrators (as proposed in [11]), (2) using analytical Jacobians and a Padé approximation method for the linearisation and discretisation steps, and (3) only partially computing the least-squares solution to the optimisation using a Cholesky decomposition. In all these cases, we could not use the internal MATLAB routines since substantial modifications to the C source code was required. At large sampling times, the optimisation problem becomes illconditioned where a singular value, rather than a Cholesky, decomposition is more appropriate. However this drastically increases the already demanding computational load, and was not deemed necessary in this application.

Table 1: Hardware implementation alternatives

Environment	*Max. sampling rate [Hz]* 266MHz	1GHz
MATLAB/RT	3	12
SIMULINK/RT & Chol	N/A	143
SIMULINK/RT sfunc.	N/A	146
xPC/C sfunc. & Chol	63	234
xPC/C sfunc. (own)	65	274

Since the MPC algorithm uses only the first of the computed control moves for the control horizon (in our case 30), it makes sense only to compute that one. Savings can be made by re-ordering the system matrices and terminating the back-substitution step after $\mathbf{u}(1)$ is computed.

A further issue characteristic of receding horizon con-

trollers is that if we increased the sampling rate, perhaps to better capture some of the nonlinearities in the linearised model, we also increase the computational load of the control law. This is because the horizon length, which is governed by the open-loop time constants remains the same. Fig. 3 shows the breakdown of the computational load of the MPC algorithm. The expensive operations are the Cholesky decomposition, and particularly the $2N_c \times 2N_p$ matrix multiplication $\mathbf{H}^T\mathbf{H}$ which have order at least $\mathcal{O}(N_p N_c^2)$. The nonlinear prediction step, the linearisation, and the creation of the dynamic matrix dominate only at the shorter control horizons.

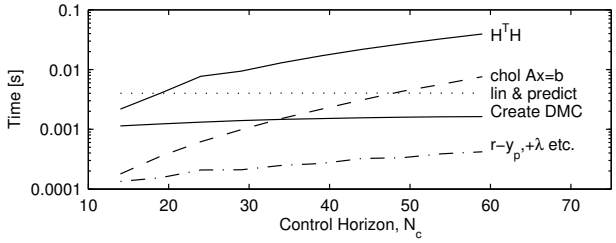

Figure 3: Execution time for key numerical operations as a function of control horizon, N_c with a prediction horizon of $N_p = 60$.

4 Results

Fig. 4 shows the controlled response of the helicopter using two PID controllers. This control is only barely acceptable, and exhibits excessive stiction and, at times, excessive oscillation.

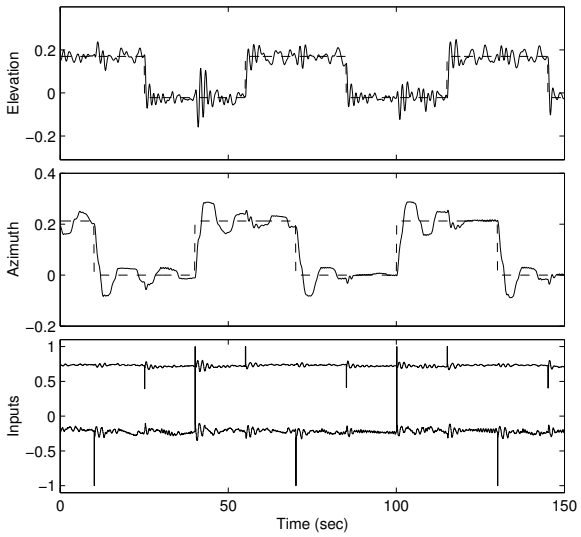

Figure 4: Closed-loop response of the helicopter using 2 PID controllers with a Δt = 10ms.

Fig. 5 shows the improved closed-loop response of the helicopter under MPC following a command signal and subjected to manual disturbances. While the PID elevation control is comparable to the predictive controller, the azimuth response, which is considerably more challenging, is substantially worse. The figure starts with prediction horizon of $N_p = 60$, and control horizons $N_c = 30$, then increasing to $N_p = 120, N_c = 40$, and finally $N_p = 60, N_c = 20$. Control horizons of about 50 is an upper limit.

It is interesting to note that integration of the full nonlinear model is *required* for the prediction step in the estimator (as well as for the computation of the free response in the MPC algorithm). If we relaxed any of the model demands (such as using a linearised model, or skip over the hard discontinuities), the estimated states from the Kalman filter rapidly diverged, and consequently the controller went unstable.

4.1 Model robustness

The model fidelity of the MPC can easily be compromised by moving the small counter weight on the helicopter as an unmeasured disturbance. Fig. 6 shows that the plant model does play a vital role in the controller. The controlled performance is noticeably deteriorated when moving the weight from the design position (100%) to 90% and 75% of the full-scale range, while a 50% change is on the brink of instability. In this disturbance however, the weight position is known and could be incorporated in the model.

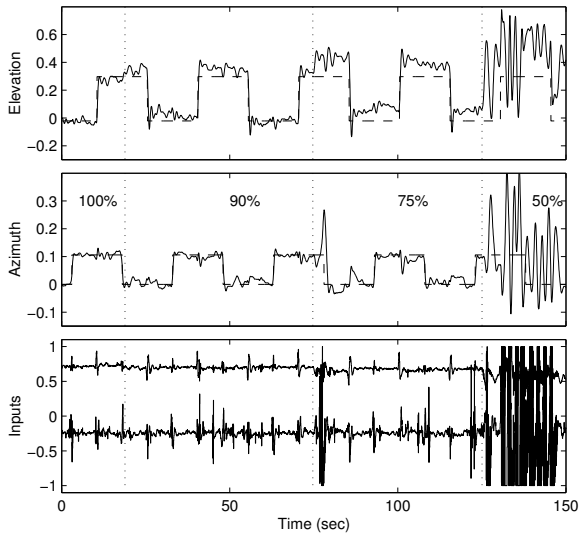

Figure 6: Closed-loop response of the helicopter using MPC with a significant model/plant mis-match introduced by moving the counter-weight.

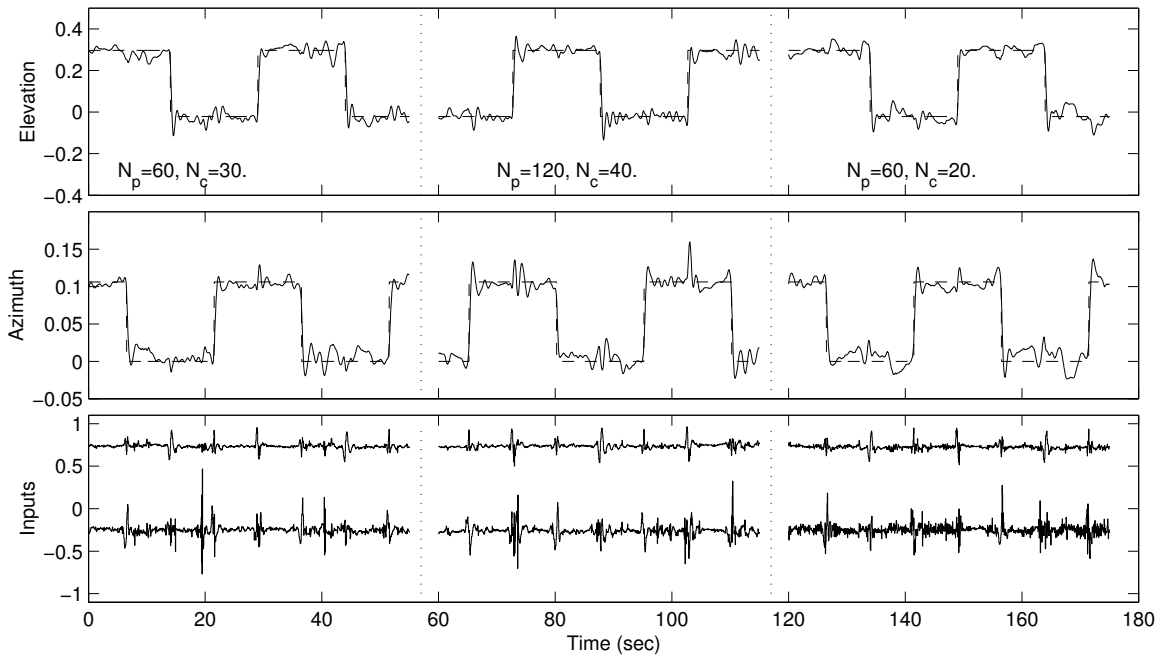

Figure 5: Closed-loop response of the helicopter using MPC with different prediction and control horizons

5 Conclusions

The bench-scale helicopter supplied by Humusoft makes for a marvellous and impressive control demonstration. Model predictive control is popular in the process industries, but does require substantial inter-sample computation. Due to the nonlinear model, the demand for relatively short sampling times, coupled with long prediction, and in particular, control horizons, SIMULINK or even raw MATLAB with the real-time toolbox did not suffice. In order for the demonstration to be sufficiently robust, we found that the controller must be implemented primarily in C with many algorithmic shortcuts and streamlined optimisations, and for it to be implemented on a stand-alone PC running xPC target. While the development and cost far outweighs what we had originally expected, the controlled performance is significantly improved over that achievable using classical control algorithms.

References

[1] David Wilson. Optimal control teaching using Matlab: Have we reached the turning point yet? In Tore Bjørnarå, editor, *Nordic Matlab Conference*, pages II–246 – II–251, Oslo, Norway, 17–18 October 2001. ISBN 82-995995-0-9.

[2] David Clarke. *Advances in Model-Based Predictive Control*. Oxford University Press, 1994.

[3] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In J.C. Kantor, C.E. Garcia, and B. Carnahan, editors, *Proc. Fifth International Conference on Chemical Process Control*, number 316 in AIChE Symposium Series, pages 232–256, Tahoe City, January 1996. AIChE.

[4] J. M. Maciejowski. *Predictive Control with Constraints*. Prentice–Hall, 2002.

[5] M. Morari and J.H. Lee. Model predictive control: Past, present and future. *Computers and Chemical Engineering*, 23:667–682, 1999.

[6] P. Krauss, K. Dass, and H. Rake. Model-based predictive controller with Kalman filtering for state estimation. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 69–83. Oxford University Press, New York, 1994.

[7] C. R. Cutler and B.L. Ramaker. Dynamic matrix control - a computer control algorithm. In *Proc. 1980 Joint automatic Control Conference*, San Fransisco, 1980. American Institute of Chemical engineers.

[8] J.H. Lee and N.L. Ricker. Extended Kalman Filter Based Nonlinear Model Predictive Control. *Ind. Eng. Chem. Res.*, 33(6):1530–1541, 1994.

[9] M.A. Henson. Nonlinear Model Predictive Control: current status and the future directions. *Computers and Chemical Engineering*, 23:187–202, 1998.

[10] Jonas Balderud. Modelling and Control of a Toy Helicopter. Master's thesis, Karlstad University, Electrical Engineering Department, December 2001.

[11] David Wilson. End-Use–Oriented Discretisation of Typical Chemical-Process Models. In *Proceedings of the IASTED Modelling, Identification and Control*, Innsbruck, Austria, Feb 18–21 2002.