Proceedings of the Canadian Society for Mechanical Engineering International Congress 2020
CSME Congress 2020
June 21-24, 2020, Charlottetown, PE, Canada

Fast Nonlinear Model Predictive Control of Quadrotors: Design and Experiments

Hadi M. Daniali[1], Nasser L. Azad[1]
[1] Department of Systems Design Engineering, University of Waterloo, Waterloo, Canada

Abstract— This paper proposes a computationally fast scheme for implementing Nonlinear Model Predictive Control (NMPC) as a high-level controller for unmanned quadrotors. The NMPC-based controller is designed using a more realistic highly nonlinear control-oriented model which requires heavy computations for practical implementations. To deal with this issue, the Newton generalized minimal residual (Newton/GMRES) method is applied to solve the NMPC's real-time optimizations rapidly during the control process. The Kalman filter and Luenberger observer algorithms are used, as well as compared, to estimate unknown states. The NMPC-based controller operation is simulated and compared with a proportional controller which shows great improvements in the response of the quadrotor. Experimental results using a commercial drone, called AR.Drone, in our laboratory instrumented by a Vicon motion capture system demonstrate that our control method is sufficiently fast for practical implementations and it can solve the trajectory tracking problem properly.

Keywords-predictive control of nonlinear systems; optimal control; autonomous robots

I. INTRODUCTION

Quadrotor (or quadcopter) is a type of Unmanned Aerial Vehicle (UAV). Due to the capabilities of quadrotors to hover, vertical take-off, and landing, they have become popular platforms and are applied to civilian and military duties, such as photography, security, search and rescue, drone-delivery, art, etc. [1-3]. To accomplish these tasks properly, quadrotors should be able to fly autonomously. To do so, they should have a capability to fly in multi-agent systems, plan paths, and track trajectories [4-6]. Accordingly, path following is one of the most significant problems discussed in the literature in recent years and a wide variety of control algorithms have been applied to quadrotors to solve this problem.

Although the quadrotor's equations of motion are generally nonlinear, several types of linear control techniques have been investigated to find a solution to the path tracking problem. The reported results of linear controllers, such as PID control, Linear Quadratic Regulator (LQR) control [7], model reference adaptive control [8], and linear Model Predictive Control (MPC) [9], reveal that these kinds of control approaches are not capable of manipulating highly nonlinear systems like quadcopters appropriately. One of the most advanced robust model-based control techniques is Learning-Based Model Predictive Control (LBMPC). This approach has been applied to quadrotors in [10-12]. Similar to any other type of MPC controllers, this method predicts the state of the system and chooses the optimum input. Therefore, it is highly dependent on the accuracy of the control-oriented (prediction) model used inside the controller [1]. To use the best linear model at any instant, LBMPC updates the model's parameters online, while linear models are not able to represent such a nonlinear platform accurately.

Also, the sliding mode control technique based on backstepping for quadrotors and feedback-linearization technique discussed in [13] and [14], are different versions of nonlinear control methods. However, unlike the NMPC method, these approaches cannot impose directly constraints on inputs and states. Moreover, as it will be discussed, NMPC tries to minimize errors with minimal efforts. So far, various NMPC-based controllers have been designed to solve the trajectory tracking problem for quadrotors. A switching model predictive controller is introduced and simulated in [15], and in [16], the NMPC method is used to control a human-sized quadrotor. These approaches have been all used to act as a low-level controller, while another alternative is to design a two-level control architecture for drones with an NMPC-based high-level controller which is presented in this study, as follows.

The low-level controller inputs of any quadrotor are left/right, forward/backward, up/down, and rotation around itself (namely z-axis). The task of this controller is to receive these inputs from the user and calculate the desired rotor inputs in such a way that the drone follows the desired path. The left/right and forward/backward commands can be denoted as ϕ and θ references, where ϕ is the roll angle and θ is the pitch angle. Besides, up/down and rotation commands can be

This work is partly supported by the Natural Sciences and Engineering Research Council of Canada.

recognized as reference values for $\dot{z}$ (vertical velocity) and $\dot{\psi}$ (yaw rate). As depicted in Figure 1, to control the drone at a particular position or path, a high-level controller obtains the desired position (x_d, y_d, and z_d) and heading (ψ_d), and send $\dot{z}$, ϕ, θ, and $\dot{\psi}$ reference values to the low-level controller.

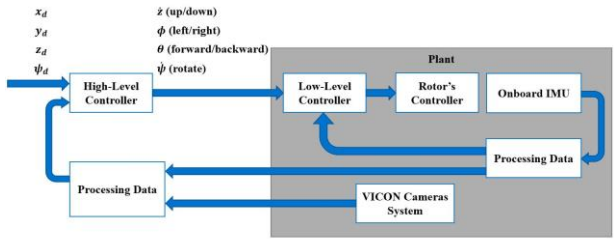


Figure 1. Two-level control architecture for drones

Due to the computational cost of NMPC real-time optimization process, which also depends on the complexity of the control-oriented model, the use of fast optimizers is critical to implement this approach in practice. As a result, in previous studies, for instance [17], where a condensed multiple shooting continuation generalized minimal residual (CMSCGMRES)-based NMPC is proposed as a high-level controller for a commercial UAV, to make the resulting controller real-time implementable, significant simplifications have been considered. In this study, a highly nonlinear model of quadrotors with more realistic behavior is introduced and used to develop an NMPC-based high-level controller. To the best of our knowledge, such a realistic control-oriented model has not been considered before to develop high-level NMPC-based controllers for quadrotors. To tackle the NMPC's real-time optimization problem, the Newton generalized minimal residual (Newton/GMRES) technique is employed. The Newton/GMRES method offers a quick scheme to calculate the Hamiltonian function's solution which can handle the real-time optimization problem properly [18-20]. To estimate the unknown states of the system, Kalman filter [21] and Luenberger Observer [22] are used, as well as compared against each other. MATLAB/Simulink has been employed to simulate the performance of the proposed NMPC-based controller and the state estimators. Finally, some experiments have been conducted with a commercial quadrotor (AR.Drone 2.0) by employing the Simulink Toolbox introduced in [23]. To do these tests, a set of off-board cameras (Vicon motion capture system) and an onboard IMU have been used to track the quadcopter's states.

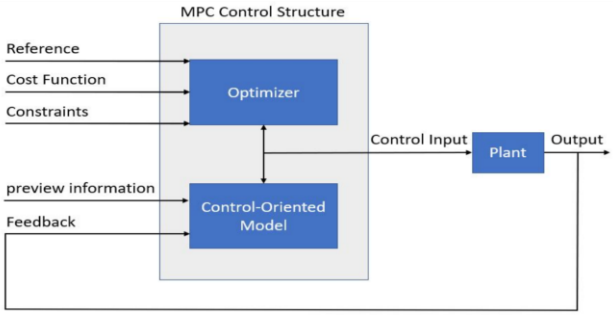


Figure 2. Block diagram of MPC [18]

The remainder of the paper is organized, as follows. In Section II, the quadrotor's high-level model is introduced. Section III discusses NMPC and Newton/GMRES in detail. Next, Section IV describes the state estimation approaches. Section V presents the simulation results, and in Section VI, the experimental setup is described, and the test results are discussed. Finally, Section VII concludes the paper.

II. System Modelling

The free body diagram of the drone is depicted in Figure 2. As shown, each rotor produces thrust which is represented by F_1, F_2, F_3, or F_4. Accordingly, F can be expressed as

$$F = F_1 + F_2 + F_3 + F_4 \tag{1}$$

Therefore,

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = -m\begin{bmatrix}0\\0\\g\end{bmatrix} + R \times \begin{bmatrix}0\\0\\F\end{bmatrix} \tag{2}$$

where g and R are the gravity acceleration and rotation matrix.

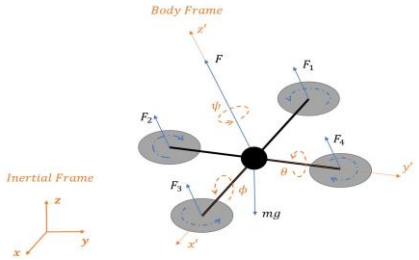


Figure 3. Free body diagram of quadrotor

By expanding the rotation matrix, the following can be written:

$$m\ddot{z} = -mg + (cos(\theta)\,cos(\phi))F \tag{3}$$

Thus, F can be expressed as a function of $\ddot{z}$.

$$F = \frac{m(\ddot{z}+g)}{cos(\theta)\,cos(\phi)} \tag{4}$$

Consequently, by using (2) and (4)

$$\ddot{x} = \left(\frac{sin(\psi)\,tan(\phi)}{cos(\theta)} + cos(\psi)tan(\theta)\right)(\ddot{z}+g)$$
$$\ddot{y} = \left(-\frac{cos(\psi)\,tan(\phi)}{cos(\theta)} + sin(\psi)tan(\theta)\right)(\ddot{z}+g) \tag{5}$$

Finally, the system model is represented in (6).

$$\dot{\boldsymbol{x}}(t) = \begin{bmatrix} x_2 \\ \left(\frac{sin(\psi)\,tan(u_2)}{cos(u_3)} + cos(\psi)tan(u_3)\right)(u_1+g) \\ x_4 \\ \left(-\frac{cos(\psi)\,tan(u_2)}{cos(u_3)} + sin(\psi)tan(u_3)\right)(u_1+g) \\ x_6 \\ u_1 \\ u_4 \end{bmatrix} \tag{6}$$

where $\boldsymbol{u}(t)$ is $[u_1, u_2, u_3, u_4]^T = [\ddot{z}, \phi, \theta, \dot{\psi}]^T$, and $\boldsymbol{x}(t)$ is $[x_1, x_2, x_3, x_4, x_5, x_6, x_7]^T = [x, \dot{x}, y, \dot{y}, z, \dot{z}, \psi]^T$.

III. Control Approach

Using the control-oriented model of the system presented in Section II, the NMPC problem can be formulated, as follows:

Minimize: $J = \Phi(\boldsymbol{x}_N(t), \boldsymbol{p}_N(t)) + \sum_{i=0}^{N-1} L(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t))\Delta\tau$

$$\text{Subject to: } \begin{cases} \boldsymbol{x}_{i+1}(t) = \boldsymbol{x}_i(t) + f\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t)\big)\Delta\tau, \\ \boldsymbol{x}_0(t) = \boldsymbol{x}(0) \\ g\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t)\big) = 0 \\ C\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t)\big) < 0 \end{cases} \quad (7)$$

Where $\boldsymbol{x}(0)$ is the current state of the UAV provided by sensors and observers, $\Delta\tau$ is the stepping time, N denotes the number of prediction horizon steps, $\boldsymbol{p}(t)$ expresses a vector of given time-dependent parameters, $f(.)$ is the dynamics of the system, $g(.)$ refers to equality constraints, and $C(.)$ expresses inequality constraints.

Similar to LQR, the purpose of NMPC is to select $\boldsymbol{u}(t)$ in a way that minimizes the errors between the actual and desired states with minimal effort by defining well-tuned $\Phi(.)$ and $L(.)$. However, they have major differences. Firstly, the NMPC's cost function can take different forms other than a quadratic function. Moreover, constraints can be added to the NMPC problem definition such that keep the system's states, outputs, or inputs within specific boundaries.

NMPC optimization problems can be solved by Hamiltonian and Newton/GMRES methods [18-20]. Based on the Hamiltonian method, the following can be stated:

$$\begin{cases} \boldsymbol{x}_{i+1}(t) = \boldsymbol{x}_i(t) + f\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t)\big)\Delta\tau \\ \lambda_i(t) = \lambda_{i+1}(t) + H_x^T(\boldsymbol{x}_i(t), \lambda_{i+1}(t), \boldsymbol{u}_i(t), \nu_i(t), \boldsymbol{p}_i(t)) \\ H_u\big(\boldsymbol{x}_i(t), \lambda_{i+1}(t), \boldsymbol{u}_i(t), \nu_i(t), \boldsymbol{p}_i(t)\big) = 0 \\ g\big(\boldsymbol{x}_i(t), \boldsymbol{u}_i(t), \boldsymbol{p}_i(t)\big) = 0 \\ \boldsymbol{x}_0(t) = \boldsymbol{x}(0) \\ \lambda_N(t) = \Phi_x^T(\boldsymbol{x}_N(t), \boldsymbol{p}_N(t)) \end{cases} \quad (8)$$

H expresses the Hamiltonian function, and λ and ν denote the co-states and Lagrange multipliers. By calculating the states forward in time and the co-states backward in time from the first and second equations of (8), $\boldsymbol{x}_i$ and λ_i can be found as sequences of $U(t) = [\boldsymbol{u}_0^T, \nu_0^T, \boldsymbol{u}_1^T, \nu_1^T, \dots, \boldsymbol{u}_{N-1}^T, \nu_{N-1}^T]$. Then, (8) can be written as one equation denoted by $F(U, x, t)$:

$$F(U,x,t) = \begin{bmatrix} H_u\big(\boldsymbol{x}_0(t), \lambda_1(t), \boldsymbol{u}_0(t), \nu_0(t), \boldsymbol{p}_{N-1}(t)\big) \\ g\big(\boldsymbol{x}_0(t), \boldsymbol{u}_0(t), \boldsymbol{p}_{N-1}(t)\big) \\ H_u\big(\boldsymbol{x}_1(t), \lambda_2(t), \boldsymbol{u}_1(t), \nu_1(t), \boldsymbol{p}_{N-1}(t)\big) \\ g\big(\boldsymbol{x}_1(t), \boldsymbol{u}_1(t), \boldsymbol{p}_{N-1}(t)\big) \\ \vdots \\ H_u\big(\boldsymbol{x}_N(t), \lambda_N(t), \boldsymbol{u}_{N-1}(t), \nu_{N-1}(t), \boldsymbol{p}_{N-1}(t)\big) \\ g\big(\boldsymbol{x}_{N-1}(t), \boldsymbol{u}_{N-1}(t), \boldsymbol{p}_{N-1}(t)\big) \end{bmatrix} = 0 \quad (9)$$

To solve (9), Newton's method will be useful:

$$F_U(U^k(t), x^k(t), t)\delta U(t) = -F(U^k(t), x^k(t), t) \quad (10)$$

$$U^{k+1}(t) = U^k(t) + \delta U(t) \quad (11)$$

Due to the complexity of calculation of Jacobian of F_U, Newton/GMRES method involves the use of Forward-Difference GMRES (fdgmres) technique which approximates the Jacobian of F_x by a forward difference approximation:

$$F_x(x)w \approx \frac{F(x+hw) - F(x)}{h} \quad (12)$$

Although by using Newton/GMRES method some approximations are made, the NMPC optimization problem can be solved quickly while the result will be accurate enough.

Result: $\delta U = fdgmres(\delta U, x, p, F, k_{max}, \eta, \rho)$
$\delta U = 0$, $r = F(U, x, t)$, $\upsilon_1 = r/||r||$, $\beta = \rho$, $l = 0$;
while $\rho > \eta||F(U, x, t)||$ *and* $l < l_{max}$ **do**
  $l = l + 1$;
  $\upsilon_{l+1} = D_h F(U, x, t : \upsilon_l, 0, 0)$;
  for $j = 1, 2, ..., l$ **do**
    $h_{j,l} = \upsilon_{l+1}^T \upsilon_j$;
    $\upsilon_{l+1} = \upsilon_{l+1} - h_{j,l}\upsilon_j$;
  end
  $h_{l+1,l} = ||\upsilon_{l+1}||$;
  $\upsilon_{l+1} = \upsilon_{l+1}/||\upsilon_{l+1}||$;
  $e_1 = (1, 0, ..., 0)^T \in \mathbb{R}^{l+1}$;
  $H_l = [h_{ij}] \in \mathbb{R}^{(l+1)\times l}$ (if $i > j + 1$ then $h_{ij} = 0$);
  Minimize $||\beta e_1 - H_l y^l||$ to find $y^l \in \mathbb{R}^l$;
  $\rho = ||\beta e_1 - H_l y^l||$;
  $V_l = [\upsilon_i] \in \mathbb{R}^{mN\times l}$;
end
$\delta U = V_l y^l$;

Algorithm 1. Forward-Difference GMRES algorithm [18,20]

IV. State Estimation

NMPC requires to have access to the states of the system. As stated before, two sensors have been employed to detect the states of the considered UAV: Vicon camera system and onboard IMU. The Vicon determines x, y, z, and ψ, and the IMU observes $\dot{x}$ and $\dot{y}$. Therefore, $\dot{z}$ is the only state which cannot be measured directly. Because of the Vicon system's noisy data, taking derivative of z for calculations is inaccurate. Also, as shown in Figure 1, $\dot{z}$ is the first input to the low-level controller. On the other hand, $\ddot{z}$ is the first output of NMPC. Hence, a method should be employed to estimate $\dot{z}$ based on the inaccurate derivative of z and the output of NMPC which is $\ddot{z}$. To do so, firstly, the observability should be checked.

A. Observability

As discussed, all of the states except for $\dot{z}$ are measurable. Therefore, according to the definition of observability, if $\dot{z}$ can be determined by knowing the input and the output of the plant over a finite time, then the system is completely observable [22]. Hence,

$$\boldsymbol{x}' = \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} z \\ \dot{z} \end{bmatrix} \quad (13)$$

$$\dot{\boldsymbol{x}}' = \begin{bmatrix} x_6 \\ u_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x' + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_1 \quad (14)$$

When z can be observed by the sensors, the observability matrix will be full rank. As a result, $\dot{z}$ can be determined from observations of x_5 (z) and u_1 ($\ddot{z}$), and the system is observable.

B. Kalman Filter

To estimate $\dot{z}$, because of the noise of z measured by the Vicon system, finding the derivative of z will not be accurate. Therefore, a Kalman filter is used to obtain a more reliable vertical velocity value. Moreover, the first NMPC's output (u_1) is $\ddot{z}$, however, the first low-level controller input is $\dot{z}$. Hence, the predicted vertical velocity obtained by the Kalman filter based on the system model can be sent to the low-level controller as a reference signal $\dot{z}$.

To design the Kalman filter, the model can be converted to a discrete form by using the forward Euler approach and

finding A, B, and C which are system's parameters [21]. Then, the Kalman estimator can be expressed as:

$$\hat{x}_6^-[k] = A\hat{x}_6[k-1] + Bu_1[k-1]$$
$$P^-[k] = AP[k-1]A + Q$$
$$\hat{x}_6[k] = \hat{x}_6^-[k] + K[k](\dot{z}[k] - C^T\hat{x}_6^-[k])$$
$$K[k] = P^-[k]C(C^TP^-[k]C + R)^{-1}$$
$$P[k] = (I - K(k)C^T)P^-[k] \quad (15)$$

where $\dot{z}$ is the derivative of the measured z at each time-step, and Q and R are the covariances of the process and measurement noises which are assumed to have white Gaussian form.

C. *Luenberger Observer*

Luenberger observer is another method to estimate the unknown states, and for the sake of comparison, it will be designed in study as follows [22]:

$$\dot{\hat{x}} = A\hat{x} + Bu + K_o(y - \hat{y}), \hat{x}(0) = \hat{x}_0, \hat{y} = C\hat{x} \quad (17)$$

where,

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, C^T = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

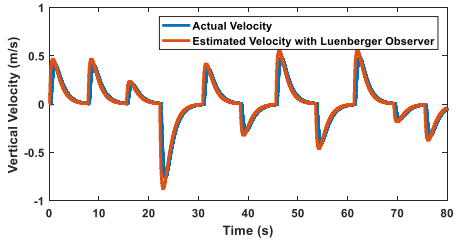
K_o is chosen such that $A - K_oC$ is a stable matrix which means:

$$\det\big(sI - (A - K_oC)\big) = (s - \lambda_0)^2 = 0, \ (\lambda_0 > 0) \quad (18)$$

V. SIMULATIONS

The discussed controller and observers have been implemented in the ARDrone Simulink Development Kit V1.1 [23]. This Simulink toolbox contains a simulation environment that simulates AR.Drone 2.0 which is parameterized by parameter identification techniques. Also, the MPsee toolbox [18] which is based on our group's previous works, is used to implement the designed NMPC controller. After modifying the model and controller's parameters, such as receding horizon length and time steps, N, $\Phi(.)$, and $L(.)$, this toolbox generates a Newton/GMRES-based NMPC controller block which is very fast.

As Figure 4 shows, the both observers can estimate the actual velocity accurately, and as depicted in Figure 5, the NMPC controller which needs vertical velocity as a feedback signal from an estimator, can control the UAV platform at different height set-points properly without any overshoot. Also, Figure 6 shows that applying these estimators will greatly improve the performance of the altitude controller. It will converge to the desired heights quickly without any overshoot.



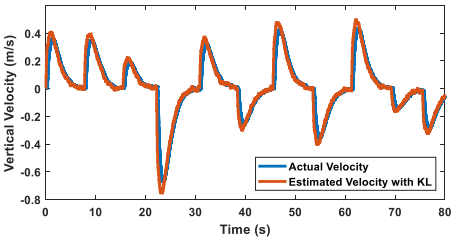


Figure 4. Estimated vertical velocity by observers in simulation

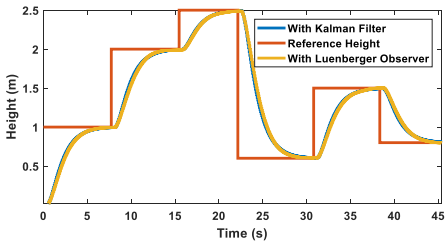


Figure 5. Simulation of the height control with applying Luenberger observer and Kalman filter

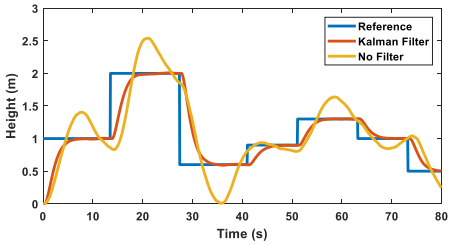
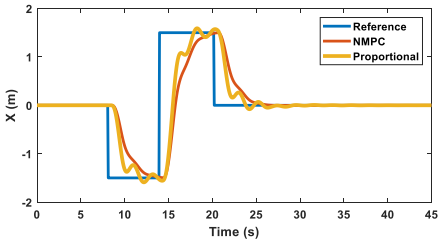


Figure 6. Effect of Kalman filter on altitude control

The proposed NMPC approach is compared with a proportional controller designed in the ARDrone Simulink Development Kit V1.1. The formulation of this controller can be expressed as:

$$\dot{z} = k_1(z_d - z)$$
$$\phi = k_2(\dot{y}_d - \dot{y}), \dot{y}_d = y_d - y$$
$$\theta = k_3(\dot{x}_d - \dot{x}), \ \dot{x}_d = x_d - x$$
$$\dot{\psi} = k_4(\psi_d - \psi) \quad (19)$$

As shown in Figure 7, the NMPC method tries to converge to the reference input smoothly while with the proportional controller, the UAV behaves much more aggressively and usually with overshoot. However, while the proportional controller performs better for altitude control, NMPC leads to a smooth response with less overshoots for other coordinates.



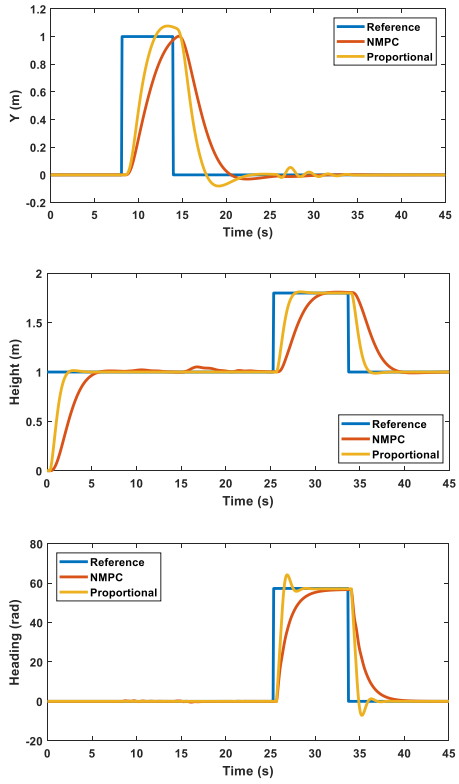


Figure 7. NMPC vs. proportional controller

VI. EXPERIMENTS

The proposed controller and observers are applied to AR.Drone in our laboratory which is equipped by a Vicon Vantage motion capture system including eight cameras (see Figure 8). This system can track spherical markers which are attached to the quadrotor and provide its pose at a rate of 100 Hz. For real-time applications, this rate can be increased to 400 Hz. This system is interconnected to a ground station computer on a local area network (LAN) and transfers the position data over UDP. Also, the ARDrone Simulink Development Kit V1.1 provides a toolbox which can communicate with AR.Drone's onboard computer via WiFi, which act as a low-level controller. The toolbox is installed on the ground station computer. The control and estimation approaches discussed in this study are implemented on this station, which can use the UDP data as feedback and send the optimal inputs to the UAV over WiFi. Also, an IMU is attached to the UAV's onboard computer to send reliable observations of $\dot{x}$ and $\dot{y}$.

Figure 8. Experiment environment

Figure 9 represents a sample of the performance of estimators. These graphs show that the estimated values can represent the actual vertical velocity with little errors, and in Figures 10, it is demonstrated that, even with such errors, the designed NMPC controller can follow the reference height properly. It should be mentioned that, without the Kalman filter, the devised NMPC controller cannot be implemented because the system will be unstable in practice.

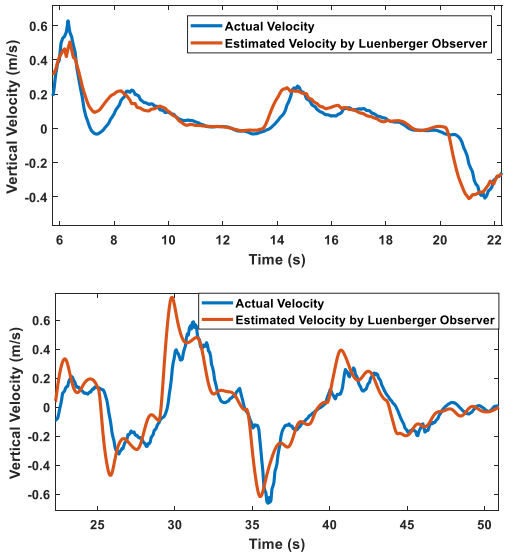


Figure 9. A sample of estimated vertical velocity by observers in practice

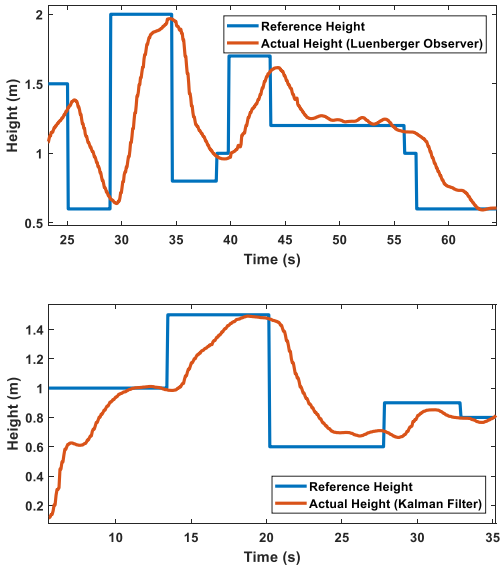


Figure 10. Experiment of height control with applying Luenberger observer and Kalman filter

Also, in Figure 11, the UAV is attempting to follow a square path. The blue and red lines are the desired and actual paths. As shown, the quadrotor remains close to the desired path with little errors.

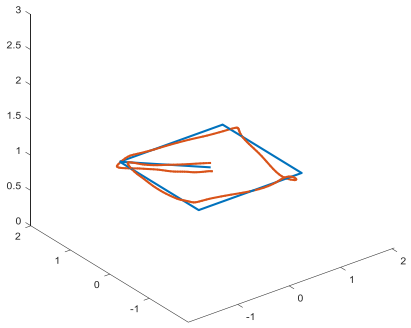


Figure 11. Square path following experimental results

VII. Conclusion

In this study, a fast-implementable NMPC-based controller was designed to perform as a high-level control module for quadrotors. The control-oriented model was highly nonlinear but more realistic, making the NMPC's real-time optimizations very time-consuming for practical applications. To resolve this issue, the controller was embedded with a fast optimizer, namely the Newton/GMRES scheme. A Kalman filter and Luenberger observer were also designed and employed to determine the unknown vertical velocity for real-life implementations. The proposed NMPC control system was compared with a proportional controller in simulations, which indicated significant improvements in the control performance. Besides, this technique was successfully implemented using a commercial UAV, AR.Drone 2.0, while a Vicon Vantage motion capture system was used to track the drone. The test results demonstrated superior performance of the proposed NMPC-based controller.

Future work will examine whether the controller can follow other complicated trajectories, such as Elliptical and Lorenz paths. Also, solving the obstacle avoidance problem using the proposed control scheme will be investigated. This can be done by adding an inequality constraint which keeps the quadrotor away from pre-defined obstacles. Moreover, the performance of the NMPC-based controller will be compared with another advanced method like Reinforcement Learning.

Acknowledgment

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada for supporting this study.

References

[1] Cao, G., Lai, E. M. K., & Alam, F. (2017). Gaussian process model predictive control of an unmanned quadrotor. *Journal of Intelligent & Robotic Systems*, *88*(1), 147-162.

[2] Abdolhosseini, M., Zhang, Y. M., & Rabbath, C. A. (2013). An efficient model predictive control scheme for an unmanned quadrotor helicopter. *Journal of intelligent & robotic systems*, *70*(1-4), 27-38.

[3] Hoffmann, G., Waslander, S., & Tomlin, C. (2006, August). Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications. In *AIAA Guidance, Navigation, and Control Conference and Exhibit* (p. 6576).

[4] Xiao, T. (2016). A Literature Review of Learning and Optimization Methods Applied to Quadrotor Control.

[5] Dong, X., Zhou, Y., Ren, Z., & Zhong, Y. (2016). Time-varying formation tracking for second-order multi-agent systems subjected to switching topologies with application to quadrotor formation flying. *IEEE Transactions on Industrial Electronics*, *64*(6), 5014-5024.

[6] Wang, D., Pan, Q., Hu, J., Zhao, C., & Guo, Y. (2019, June). MPCC-based Path Following Control for a Quadrotor with Collision Avoidance Guaranteed in Constrained Environments. In *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)* (pp. 581-586). IEEE.

[7] Bouabdallah, S., Noth, A., & Siegwart, R. (2004, September). PID vs LQ control techniques applied to an indoor micro quadrotor. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No.* 04CH37566*)* (Vol. 3, pp. 2451-2456). IEEE.

[8] Selfridge, J. M., & Tao, G. (2014). A multivariable adaptive controller for a quadrotor with guaranteed matching conditions. *Systems Science & Control Engineering: An Open Access Journal*, *2*(1), 24-33.

[9] Iskandarani, M., Givigi, S. N., Rabbath, C. A., & Beaulieu, A. (2013, June). Linear model predictive control for the encirclement of a target using a quadrotor aircraft. In *21st Mediterranean Conference on Control and Automation* (pp. 1550-1556). IEEE.

[10] Bouffard, P., Aswani, A., & Tomlin, C. (2012, May). Learning-based model predictive control on a quadrotor: Onboard implementation and experimental results. In *2012 IEEE International Conference on Robotics and Automation* (pp. 279-284). IEEE.

[11] Aswani, A., Bouffard, P., & Tomlin, C. (2012, June). Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In *2012 American Control Conference (ACC)* (pp. 4661-4666). IEEE.

[12] Aswani, A., Gonzalez, H., Sastry, S. S., & Tomlin, C. (2013). Provably safe and robust learning-based model predictive control. *Automatica*, *49*(5), 1216-1226.

[13] Bouadi, H., Bouchoucha, M., & Tadjine, M. (2007). Sliding mode control based on backstepping approach for an UAV type-quadrotor. *World Academy of Science, Engineering and Technology*, *26*(5), 22-27.

[14] Voos, H. (2009, April). Nonlinear control of a quadrotor micro-UAV using feedback-linearization. In *2009 IEEE International Conference on Mechatronics* (pp. 1-6). IEEE.

[15] Alexis, K., Nikolakopoulos, G., & Tzes, A. (2014). On trajectory tracking model predictive control of an unmanned quadrotor helicopter subject to aerodynamic disturbances. *Asian Journal of Control*, *16*(1), 209-224.

[16] Zanelli, A., Horn, G., Frison, G., & Diehl, M. (2018, June). Nonlinear model predictive control of a human-sized quadrotor. In *2018 European Control Conference (ECC)* (pp. 1542-1547). IEEE.

[17] Dentler, J., Kannan, S., Mendez, M. A. O., & Voos, H. (2016, September). A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In *2016 IEEE conference on control applications (CCA)* (pp. 519-525). IEEE.

[18] Tajeddin, S. (2016). Automatic code generation of real-time nonlinear model predictive control for plug-in hybrid electric vehicle intelligent cruise controllers (Master's thesis, University of Waterloo).

[19] Ohtsuka, T. (2004). A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, *40*(4), 563-574.

[20] Kelley, C. T. (1995). Iterative methods for linear and nonlinear equations (Vol. 16). Siam.

[21] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics*. MIT press.

[22] Ioannou, P., & Fidan, B. (2006). *Adaptive control tutorial*. Society for Industrial and Applied Mathematics.

[23] Sanabria, D. E., & Mosterman, P. (2013). Ardrone simulink development kit v1. 1.