ADCONIP 2014 Hiroshima

Real-Time Swing-up of Double Inverted Pendulum by Nonlinear Model Predictive Control

Pathompong Jaiwat[1] and Toshiyuki Ohtsuka[2]

***Abstract*— In this study, the swing-up of a double inverted pendulum is controlled by nonlinear model predictive control (NMPC). The fast computation algorithm called C/GMRES (continuation/generalized minimal residual) is applied to solve a nonlinear two-point boundary value problem over a receding horizon in real time. The goal is to swing-up and stabilize two pendulums from the downward to upright position. To make the tuning process of the performance index simpler, the terminal cost in the performance index is given by a solution of the algebraic Riccati equation. The simulation results show that C/GMRES can solve the NMPC problem in real time and swing-up the double inverted pendulum with a significant reduction in the computational cost compared with Newton's method.**

I. Introduction

An inverted pendulum is a mechanical system consisting of a pendulum connected to a cart that move freely on the horizontal axis. The inverted pendulum is a classic problem in dynamics and control theory and is widely used for testing many types of control algorithm because of its highly nonlinear characteristic. Also, there are many practical problems for which the inverted pendulum is used as a representative model such as the stabilization of humanoid or robot motion [1], [2], the motion of a hexapod robot [3], the launching of a rocket, the movement of the human arm [4], and the behavior of a vehicle during rollover [5], [6]. Therefore, there have been numerous studies on inverted pendulums such as on their stabilization and swing-up.

A double inverted pendulum is an extension of a single pendulum, with one more pendulum added to a single inverted pendulum. Therefore, a double inverted pendulum has an extremely high nonlinearity property compared with a single inverted pendulum and it is very difficult to swing-up and stabilize a double inverted pendulum. Despite this, more sophisticated types of inverted pendulums such as triple and quintuple inverted pendulums can be stabilized by using a Linear-Quadratic Regulator (LQR) controller and a fuzzy control method [7], [8]. However, these control methods only deal with the stabilization and only allow the limited range of movement of the pendulum around the upright equilibrium point, where the system can be approximated as a linear system.

Many types of control algorithm have been applied to stabilize the double inverted pendulum about the upright equilibrium point, such as optimal control combined with neural network adaptive control [9] and LQR combined with Proportional-Derivative (PD) control [10]. One of the challenging problems of the double inverted pendulum is its swing-up from the downward position to the upright position. The swing-up of the double inverted pendulum can be performed by using an energy method and by separating the swing-up into three steps: first stabilizing the first pendulum in the upright position, then swinging-up the second pendulum while stabilizing the first, and finally stabilizing both pendulums in upright position [11]. However, this control method requires considerable time before both pendulums reach the upright position. Another method of swing-up is to using feedback control and bang-bang control [12]. However this method requires a torque exerted about the suspension point, which is not practical in some systems. The swing-up of double and triple inverted pendulums has been demonstrated experimentally in [13], [14]. However, the nominal trajectories in these experiments were calculated offline.

Nonlinear model predictive control (NMPC) has never been applied to the double inverted pendulum because of its high computational cost and the highly nonlinear properties of the double pendulum. That is, the swing-up double inverted pendulum by NMPC will have a high computational cost because of the need to solve the two-point boundary value problem (TP-BVP) in real time. This requirement is very difficult to achieve.

In this study, the double inverted pendulum is swung-up and stabilized by NMPC, where the fast calculation method called C/GMRES [15] is applied to solve the NMPC problem to achieve the real-time requirement. In this paper, we describe the dynamic model of the double inverted pendulum in Section II. The NMPC and C/GMRES algorithms are described in Section III. The simulation results are reported in Section IV. Finally, conclusions are given in Section V.

II. Double Inverted Pendulum

A. Dynamic model of double inverted pendulum

The double inverted pendulum is a mechanical system consisting of a series of two pendulums attached to a cart that moves freely on a horizontal surface. It has three degrees of freedom, which are the horizontal position x, the first pendulum angle θ_1, and the second pendulum angle θ_2. The free body diagram of the double inverted pendulum system is shown in Fig. 1. The cart has mass M, the first pendulum has mass m_1, and the second pendulum has mass m_2. The

[1]Pathompong Jaiwat is with the Graduate School of Engineering Science, Department of Systems Innovation, Osaka University, Osaka, Japan jaiwat@arl.sys.es.osaka-u.ac.jp

[2]Toshiyuki Ohtsuka is with the Graduate School of Informatics, Department of Systems Science, Kyoto University, Kyoto, Japan ohtsuka@i.kyoto-u.ac.jp

moments of inertia at the center of gravity (C.G.) of the first and second pendulums are I_1 and I_2, respectively.

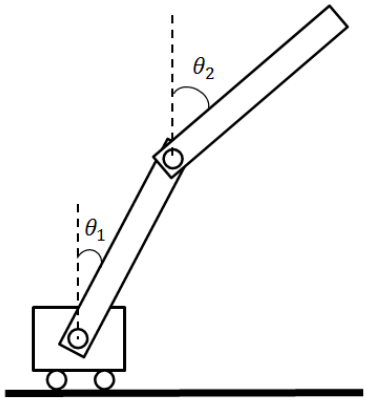

Fig. 1. Double inverted pendulum

To obtain the equations of motion of the double inverted pendulum, the coordinates relative to the C.G. must be defined. The coordinate of cart mass M is defined by (X_M, Y_M), and the coordinate of first and second pendulum are (X_{m_1}, Y_{m_1}) and (X_{m_2}, Y_{m_2}), respectively.

The coordinates (X_M, Y_M), (X_{m_1}, Y_{m_2}) and (X_{m_1}, Y_{m_2}) are given by

$$\begin{aligned} X_M &= x, \\ Y_M &= 0, \\ X_{m_1} &= x + l_1 \sin\theta_1, \\ Y_{m_1} &= l_1 \cos\theta_1, \\ X_{m_2} &= x + 2l_1 \sin\theta_1 + l_2 \sin\theta_2, \\ Y_{m_2} &= 2l_1 \cos\theta_1 + l_2 \cos\theta_2. \end{aligned}$$

The Lagrangian L is defined by $L = T - V$, where T is the kinetic energy and V is the potential energy of the system. The kinetic and potential energies of the double inverted pendulum can be obtained from the following equations:

$$\begin{aligned} T_M &= \frac{1}{2}M(\dot{X}_M^2 + \dot{Y}_M^2), \\ T_{m_1} &= \frac{1}{2}m_1(\dot{X}_{m_1}^2 + \dot{Y}_{m_1}^2) + \frac{1}{2}I_1\dot{\theta_1}^2, \\ T_{m_2} &= \frac{1}{2}m_2(\dot{X}_{m_2}^2 + \dot{Y}_{m_2}^2) + \frac{1}{2}I_2\dot{\theta_2}^2, \\ V_M &= 0, \\ V_{m_1} &= mgY_{m_1}, \\ V_{m_2} &= mgY_{m_2}. \end{aligned}$$

Thus, the Lagrangian can be expressed as $L = T_M + T_{m_1} + T_{m_2} - (V_M + V_{m_1} + V_{m_2})$. The equation of motion can be obtained from the following equations:

$$\begin{aligned} \frac{d}{dt}\frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} &= u, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta_1}} - \frac{\partial L}{\partial \theta_1} &= 0, \\ \frac{d}{dt}\frac{\partial L}{\partial \dot{\theta_2}} - \frac{\partial L}{\partial \theta_2} &= 0. \end{aligned} \tag{1}$$

Equation (1) can be rewritten in the following simpler matrix form:

$$C(\mathbf{q})\ddot{\mathbf{q}} + D(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + E(\mathbf{q}) = Gu, \tag{2}$$

where $\mathbf{q} = [x, \theta_1, \theta_2]^T$. Equation (2) can be rearranged in the form $\dot{\mathbf{x}} = f(\mathbf{x}, u)$, where $\mathbf{x} = [\mathbf{q}, \dot{\mathbf{q}}]^T = [x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2]^T$ is the state vector and, in particular, we have

$$f(\mathbf{x}, u) = \begin{bmatrix} \dot{\mathbf{q}} \\ C^{-1}(Gu - D\dot{\mathbf{q}} - E) \end{bmatrix}. \tag{3}$$

B. Linearization

As described in Subsection III-D, the terminal cost can be obtained by solving the algebraic Riccati equation. This equation is associated with the linear dynamic system, expressed by

$$\dot{\mathbf{x}} = A\mathbf{x} + Bu.$$

Equation (3) can be linearized to derive an approximate linear solution around the upward equilibrium point $\mathbf{x} = 0$; this yields

$$A = \begin{bmatrix} 0 & I \\ -C(0)^{-1}\frac{\partial E(0)}{\partial \mathbf{q}} & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ C(0)^{-1}G \end{bmatrix},$$

where I is the identity matrix.

III. Real-Time Nonlinear Model Predictive Control Problem

A. Problem formulation

MPC is a feedback control method that uses current state values and target values of a state variable to predict and optimize the system responses in advance. The goal is to make the system responses close to the target values. Normally, MPC optimizes a series of state trajectories on a receding horizon by providing an optimal input. Its uses the current state variables to find a series of optimal inputs, updates the state variables in the next sampling period, and repeats the calculation in the next time step.

To apply MPC to a nonlinear system, a nonlinear two point boundary value problem (NTP-BVP) must be solved. NMPC can be initially formulated by the nonlinear state equation

$$\dot{x}(t) = f(x(t), u(t), p(t)),$$

where $x(t) \in \mathbb{R}^n$ is the state vector, $u(t) \in \mathbb{R}^{m_u}$ is the input vector, and $p(t) \in \mathbb{R}^{m_p}$ is the vector of given time-dependent parameters. The control input $u(t)$ is determined at each time t to optimize a performance index J with a receding horizon from t to $t + T$. The NMPC problem is a

set of finite-horizon optimal control problems along with a fictitious horizon time τ as follows.

Minimize

$$J = \phi(x^*(t+T,t), p(t+T)) + \int_t^{t+T} L(x^*(\tau,t), u^*(\tau,t), p(\tau))d\tau,$$

subject to

$$\begin{cases} x^*_\tau(\tau,t) = f(x^*(\tau,t), u^*(\tau,t), p(\tau)), \\ x^*(t,t) = x(t), \\ C(x^*(\tau,t), u^*(\tau,t), p(\tau)) = 0, \end{cases}$$

where x_τ represents $\partial x/\partial\tau$, $x^*(\tau,t)$ is the trajectory along the fictitious time horizon τ, starting from $x(t)$ at $\tau = t$, and $u^*(\tau,t)$ is the control input, which is determined on the receding horizon as the solution of the finite-horizon optimal control problem. The actual input to the system is given by $u(t) = u^*(t,t)$ and $C(x,u)$ is an equality constraint of the system. An inequality constraint can be transformed into the equality constraint by introducing a dummy input as explained in [15].

B. Discretized problem

In this subsection, we describe the method of finding the optimal control on a discretized horizon. First, the horizon is divided into N steps and the optimal control problem is discretized on the τ-axis using the forward difference method as follows:

$$x^*_{i+1}(t) = x^*_i(t) + f(x^*_i(t), u^*_i(t), p^*_i(t))\Delta\tau, \tag{4a}$$
$$x^*_0(t) = x(t), \tag{4b}$$
$$C(x^*_i(t), u^*_i(t), p^*_i(t)) = 0, \tag{4c}$$
$$J = \phi(x^*_N(t), p^*_N(t)) + \sum_{i=0}^{N-1} L(x^*_i(t), u^*_i(t), p^*_i(t))\Delta\tau, \tag{4d}$$

where $\Delta\tau := T/N$, $x^*_i(t)$ corresponds to $x^*(i\Delta\tau, t)$, and $p^*_i(t)$ is given by $p(t + i\Delta\tau)$.

The discretized problem is solved at every sampling time t. Therefore, the input sequence on the horizon $\{u^*_i(t)\}_{i=0}^{N-1}$ is optimized at each sampling time t. To find the minimum of the performance index given by Eq. (4d), the control input should be chosen to minimize the Hamiltonian, which is a consequence of Pontryagin's minimum principle [16]. Let H denote the Hamiltonian defined by

$$H(x,\lambda,u,\mu,p) := L(x,u,p) + \lambda^T f(x,u,p) + \mu^T C(x,u,p),$$

where $\lambda \in \mathbb{R}^n$ is the costate and $\mu \in \mathbb{R}^{m_c}$ is the Lagrange multiplier associated with the equality constraint. The first-order necessary conditions for the sequences of the optimal control inputs $\{u^*_i(t)\}_{i=0}^{N-1}$, multipliers $\{\mu^*_i(t)\}_{i=0}^{N-1}$, and costates $\{\lambda^*_i(t)\}_{i=1}^{N}$ along the horizon can be found by using Pontryagin's minimum principle:

$$H_u(x^*_i(t), \lambda^*_{i+1}(t), u^*_i(t), \mu^*_i(t), p^*_i(t)) = 0, \tag{5a}$$
$$\lambda^*_i(t) = \lambda^*_{i+1}(t) + H^T_x(x^*_i(t), \lambda^*_{i+1}(t), u^*_i(t), \mu^*_i(t), p^*_i(t))\Delta\tau, \tag{5b}$$
$$\lambda^*_N(t) = \phi^T_x(x^*_N(t), p^*_N(t)), \tag{5c}$$

where H_u, H_x, and ϕ_x are $\partial H/\partial u$, $\partial H/\partial x$, and $\partial\phi/\partial x$, respectively. Note that the optimal input sequences $\{u^*_i(t)\}_{i=0}^{N-1}$ and $\{\mu^*_i(t)\}_{i=0}^{N-1}$ must satisfy Eqs. (5a)–(5c). Equations (5a)–(5c) are the necessary conditions for optimality. Since we wish to optimize the sequences of $\{u^*_i(t)\}_{i=0}^{N-1}$ and $\{\mu^*_i(t)\}_{i=0}^{N-1}$, we define a vector comprising the inputs and multipliers as

$$U(t) := [u^{*T}_0(t), \mu^{*T}_0(t), u^{*T}_1(t), \mu^{*T}_1(t), \ldots, u^{*T}_{N-1}(t), \mu^{*T}_{N-1}(t)]^T,$$

where $U(t) \in \mathbb{R}^{mN}$ and $m{:=}m_u + m_c$. From Eq. (5c), the costate λ_N depends on x_N. It can also be seen that, for the sequence $\{x^*_i(t)\}_{i=0}^{N}$, the future state depends on the current state. However, for the costate sequence $\{\lambda^*_i(t)\}_{i=0}^{N}$, the current costate depends on the future costate. That is, $\{x^*_i(t)\}_{i=0}^{N}$ is calculated recursively using Eqs. (4a) and (4b), and then $\{\lambda^*_i(t)\}_{i=0}^{N}$ is determined recursively from $i = N$ to $i = 1$ using Eqs. (5b) and (5c). Since $x^*_i(t)$ and $\lambda^*_{i+1}(t)$ in Eqs. (4c) and (5a) are determined by Eqs. (4a), (4b), (5b), and (5c), Eqs. (4c) and (5a) can be regarded as a single equation represented by a column vector F as follows:

$$F(U(t), x(t), t) := \begin{bmatrix} H^T_u(x^*_0(t), \lambda^*_1(t), u^*_0(t), \mu^*_0(t), p^*_0(t)) \\ C(x^*_1(t), u^*_1(t), p^*_1(t)) \\ \vdots \\ H^T_u(x^*_{N-1}(t), \lambda^*_N(t), u^*_{N-1}(t), \mu^*_{N-1}(t), p^*_{N-1}(t)) \\ C(x^*_{N-1}(t), u^*_{N-1}(t), p^*_{N-1}(t)) \end{bmatrix} = 0. \tag{6}$$

Solving Eq. (6) gives the optimal vector $U(t)$, which is a sequence of inputs and Lagrange multipliers.

C. Continuation/GMRES

From the previous subsection, to solve the optimal control problem, the equation $F(U,x,t) = 0$ (Eq. (6)) must be solved at each sampling time. However, iterative methods should be avoided because of their high computational cost. This subsection describes an alternative method called the C/GMRES method, which greatly reduces the computational cost. In this method, instead of solving Eq. (6) directly, we find the derivative of U with respect to time, $\dot{U}$, such that $F(U(t), x(t), t) = 0$ is satisfied by choosing $U(0)$ so that $F(U(0), x(0), 0) = 0$. Then we determine $\dot{U}$ so that

$$\dot{F}(U,x,t) = A_s F(U,x,t), \tag{7}$$

where A_s is a stable matrix used to stabilize $F = 0$. From the chain rule of differentiation, Eq. (7) becomes

$$F_U \dot{U} = A_s F - F_x \dot{x} - F_t.$$

If F_U is nonsingular, we obtain the following differential equation:

$$\dot{U} = F_U^{-1}(A_s F - F_x \dot{x} - F_t). \tag{8}$$

The vector $\dot{U}$ can be determined using Eq. (8) for the given variables $U, x, \dot{x}$, and t. The optimal solution $U(t)$ can be found without the use of an iterative optimization method by computing $U(t + \Delta t) = U(t) + \dot{U}(t)\Delta t$ in real time. However, finding $\dot{U}$ using Eq. (8) still has a high computational cost because of the need to find the Jacobians F_U, F_x, F_t, and F_U^{-1}. Moreover, from the definition of vector F in Eq. (6), the Jacobian F_U is dense. Therefore, to reduce the computational cost, we employ the following techniques.

1) Forward difference GMRES method: Equation (7) can be approximated as a linear equation by using forward difference GMRES (FDGMRES) to solve it for $\dot{U}$. For more information about FDGMRES, please refer to [15], [17]. FDGMRES is combined with the continuation method for real-time computation.

2) Continuation method: Sinve the vector $\dot{U}$ has been obtained by the FDGMRES method, $U(t)$ can be updated by integrating $\dot{U}$ in real time. For more information about this technique, please refer to [15].

The continuation method and FDGMRES are combined to form C/GMRES, which is a fast calculation algorithm. C/GMRES is used to find the solution of the nonlinear Eq. (6), which can be found without a line search from Newton's method and without directly solving nonlinear Eq. (6). Therefore, it requires much less computational time [15].

In spite of using C/GMRES to find the solution of Eq. (6), the weighting matrices in Eq. (4d) must be carefully selected to obtain good responses. A method for obtaining suitable weighting matrices is described in the next subsection.

D. Performance index

In this subsection, we discuss how to obtain appropriate weighting matrices for the performance index. The objective of the optimal control problem is to find the optimal solution of the input by minimizing the performance index. The control input at each time t is determined to minimize the performance index with horizon length T, as described in Subsection III-A. To control the position of the system, the functions in the performance index J given by Eq. (4d) are chosen as

$$\phi(x) = \frac{1}{2}(x - x_f)^T S_f (x - x_f), \tag{9a}$$

$$L(x, u) = \frac{1}{2}((x - x_f)^T Q(x - x_f) + (u - u_f)^T R(u - u_f)), \tag{9b}$$

where x_f denotes the objective state, u_f is the objective input and S_f, Q, and R are weighting matrices. Normally, matrices Q, R, and S_f are selected such that the response of the system is satisfactory. However, satisfactory responses are sometimes very difficult to achieve because it is very difficult to adjust these three weighting matrices (S_f, Q, and R) at the same time. To reduce the number of variables that need to be adjusted, the matrix S_f (terminal weighting matrix) can be mathematically found by solving the algebraic Riccati equation; a similar principle was used in [18], [19].

Let us consider an infinite-horizon optimal control problem with the quadratic cost function

$$J(u) = \int_0^\infty ((x - x_f)^T Q(x - x_f) + (u - u_f)^T R(u - u_f))dt$$

for the linear system

$$\dot{x} = Ax + Bu.$$

This type of problem is called a linear quadratic regulator (LQR) problem. The optimal control is obtained as the state-feedback $u = -Kx$ with the gain matrix given by $K = R^{-1}B^T S_f$. The matrix S_f can be found by solving the following algebraic Riccati equation:

$$A^T S_f + S_f A - S_f B R^{-1} B^T S_f + Q = 0. \tag{10}$$

It is well known that the minimum value of the performance index is expressed as

$$\min J = \int_0^\infty \frac{d}{dt}(x^T S_f x)dt = \frac{1}{2}x^T(0) S_f x(0). \tag{11}$$

Now, let us consider NMPC with an infinite horizon, i.e., $T = \infty$. Then, the performance index can be written as

$$\begin{aligned} J &= \int_t^\infty ((x - x_f)^T Q(x - x_f) + (u - u_f)^T R(u - u_f))d\tau, \\ &= \int_t^{t+T} ((x - x_f)^T Q(x - x_f) + \\ &\quad (u - u_f)^T R(u - u_f))d\tau + \\ &\quad \int_{t+T}^\infty ((x - x_f)^T Q(x - x_f) + \\ &\quad (u - u_f)^T R(u - u_f))d\tau. \end{aligned}$$

Applying the basic property of the LQR, Eq. (11), to the second term of the above equation and assuming that $x(t+T)$ is close to its original value, the performance index can be approximated as

$$\begin{aligned} J \approx &\frac{1}{2}(x(t + T) - x_f)^T S_f (x(t + T) - x_f) + \\ &\frac{1}{2}\int_t^{t+T} ((x - x_f)^T Q(x - x_f) + \\ &(u - u_f)^T R(u - u_f))d\tau \end{aligned}$$

It can be seen that the terminal-state penalty term $(x(t + T) - x_f)^T S_f (x(t + T) - x_f)$ is added to the finite-horizon performance index. For NMPC, the matrix S_f is found by solving Eq. (10) offline [20].

IV. Simulation Results

This section shows results obtained by simulation while adjusting the weighting matrices R and Q in the performance index Eq. (9b). The matrix S_f is obtained by solving Eq. (10). Table I shows the model parameters used in the simulation.

TABLE I
Model Parameters

Symbol	Quantity	Value
M	Cart mass	1 kg
m_1	First pendulum mass	0.8 kg
m_2	Second pendulum mass	0.5 kg
I_1	First pendulum moment of inertia	0.0126 kg m^2
I_2	Second pendulum moment of inertia	0.0185 kg m^2
l_1	First pendulum length	0.3 m
l_2	Second pendulum length	0.45 m

In the simulation, the sampling period is 0.005 s and the horizon length is divided into 5 steps ($N = 5$). The parameters associated with the horizon are chosen as $T = T_f(1 - e^{-\alpha t})$, $T_f = 1$ s, and $\alpha = 0.5$. Since the dynamical behaviors of the system to be observed are $\mathbf{x} = [x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2]^T$ and this simulation is focused on stabilizing θ_1 and θ_2, it is important to stabilize the angles of both pendulums to zero. Therefore, the angles are given the largest weight. The weighting matrices of the performance index are chosen as: $Q = \mathrm{diag}[5, 10, 10, 0, 0, 0]$ and $R = 1$. The objective state is defined as $\mathbf{x}_f = [0, 0, 0, 0, 0, 0]^T$. The initial state of the system is $\mathbf{x}_0 = [0, \pi, \pi, 0, 0, 0]^T$. The error in the optimality condition is represented by the norm of the function F in Eq. (6), $\|F\|$, upon substituting the input and the current state.

The simulation results were computed on a Windows 8 operating system with a 2.5 GHz Intel Dual-Core i5 CPU and 4 GB of RAM. The simulation code was written in MATLAB. For faster simulation, the nonlinear equation Eq. (6) was written in a MEX (MATLAB executable) file, which provides an interface between MATLAB and subroutines written in C/C++. The transformation between MATLAB's M-file to the MEX file was simply done by using the command `mex` in MATLAB.

Figure 2 shows the result obtained using Newton's method to solve the nonlinear equation (Eq. (6)) and C/GMRES at every sampling period. Newton's method was chosen to solve the NMPC problem because it is a fundamental method and widely used to solve nonlinear equations. The results show that the double inverted pendulum can be swung-up and stabilized in the upright position for both methods and that the time responses for both methods are similar except for the displacement x.

The difference in displacement between C/GMRES and Newton's method is caused by C/GMRES, which is a fast calculation method, combined with a lower weighting value for the displacement and zero weight for the velocity. To realize real-time calculation, the accuracy of C/GMRES must be reduced. Therefore, the slight difference in the cart velocity will affect the time history of the displacement as shown in Fig. 2. It can be seen from Fig. 2 that Newton's method has a significantly lower error of only about 1×10^{-7} compared with C/GMRES. The time responses and the error the of C/GMRES method can be made closer to those of Newton's method by reducing the sampling time of NMPC and the tolerance value in the GMRES method. However, there is a trade-off between accuracy and computational time.

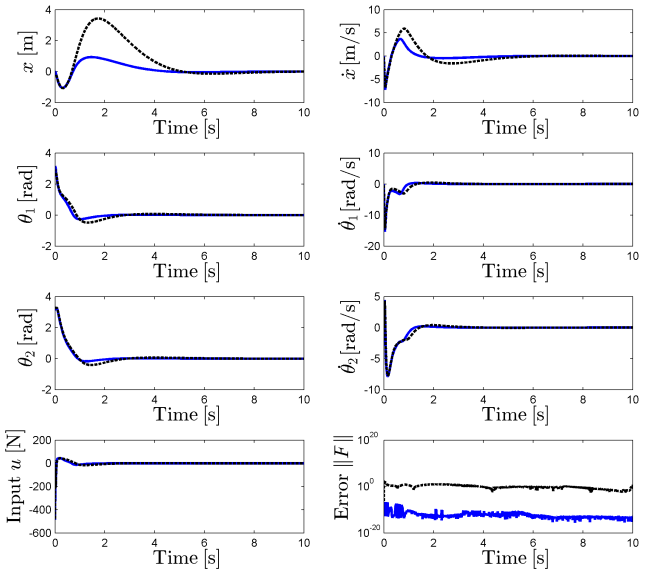

Fig. 2. Simulation results obtained using Newton's method (solid lines) and C/GMRES (dashed lines)

TABLE II
Average Computational Cost

Method	Computational Time per Update (s)	
	MATLAB	MATLAB & MEX
Newton's Method	0.1314	0.0207
C/GMRES	0.0669	0.0048

Table II shows the computational time per update for Newton's method and the C/GMRES method. Table II also shows the computational times for M and MEX file. The results show that the C/GMRES method has a lower computational cost than Newton's method, which is further reduced when a MEX file is used. It can be seen that the computational cost of C/GMRES with a MEX file is less than the sampling time of 0.005 s. Therefore, only the C/GMRES method can be used in real time.

V. Conclusions

The double inverted pendulum can be swung-up and stabilized by NMPC. Real-time NMPC can be performed by applying C/GMRES to solve the NMPC problem. In this study, the double inverted pendulum was initially at rest

in the downward position. Then, NMPC was used to find the optimal input by optimizing the performance index. The terminal weighting matrix was chosen by solving the algebraic Riccati equation. A continuous-time NMPC problem was first discretized over a receding horizon, and an NTP-BVP was formulated to find the sequence of optimal control inputs. To solve this nonlinear equation, Newton's method is conventionally used, which has a large computational cost. The results showed that the pendulum can be swung-up and stabilized in a short settling time using NMPC. To realize real-time computation, the fast computation method called C/GMRES was applied to solve the NMPC problem, which was coded in a MATLAB MEX file, and the results showed that the computational time was significantly reduced to less than the sampling time. Therefore, C/GMRES can be applied to solve the NMPC problem in real time.

REFERENCES

[1] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through ZMP manipulation based on inverted pendulum control," *Proceedings of 2002 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1404–1409, 2002.

[2] A. D. Kuo, "The six determinants of gait and the inverted pendulum analogy: A dynamic walking perspective," *Human Movement Science*, vol. 26, no. 4, pp. 617–656, 2007.

[3] R. Altendorfer, U. Saranli, H. Komsuoglu, D. Koditschek, H. B. Brown Jr, M. Buehler, N. Moore, D. McMordie, and R. Full, *Evidence for Spring Loaded Inverted Pendulum Running in a Hexapod Robot*. Springer, 2001.

[4] S. Jadlovská and J. Sarnovsky, "Classical double inverted pendulum–A complex overview of a system," *Proceedings of IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics*, pp. 103–108, 2012.

[5] P. Jaiwat and T. Ohtsuka, "Stabilization of vehicle rollover by nonlinear model predictive control," *Proceedings of SICE Annual Conference 2013*, pp. 1568–1573, Sept. 2013.

[6] S. C. Peters, J. E. Bobrow, and K. Iagnemma, "Stabilizing a vehicle near rollover: An analogy to cart-pole stabilization," *Proceedings of 2010 IEEE International Conference on Robotics and Automation*, pp. 5194–5200, May 2010.

[7] E. Lee and J. Perkins, "Comparison of techniques for stabilization of a triple inverted pendulum." [Online]. Available: http://www.cc.gatech.edu/~mstilman/class/RIP08/FINAL_PROJECTS/ErikJim.pdf

[8] C. Luo, D. Hu, Y. Pang, X. Zhu, and G. Dong, "Fuzzy control of a quintuple inverted pendulum with the LQR method and 2-ary fuzzy piecewise interpolation function," *Proceedings of 45th IEEE Conference on Decision and Control*, pp. 6307–6312, 2006.

[9] A. Bogdanov, "Optimal control of a double inverted pendulum on a cart," *OGI School of Science and Engineering, OHSU*, 2004.

[10] N. Singh and S. K. Yadav, "Comparison of LQR and PD controller for stabilizing double inverted pendulum system," *International Journal of Engineering*, vol. 1, no. 12, pp. 69–74, 2012.

[11] T. Henmi, M. Deng, and A. Inoue, "Swing-up control of a serial double inverted pendulum," *Proceedings of 2004 American Control Conference*, vol. 5, pp. 3992–3997, 2004.

[12] J. Awrejcewicz, G. Wasilewski, G. Kudra, and S. Reshmin, "An experiment with swinging up a double pendulum using feedback control," *Journal of Computer Systems Sciences International*, vol. 51, no. 2, pp. 176–182, 2012.

[13] K. Graichen, M. Treuer, and M. Zeitz, "Swing-up of the double pendulum on a cart by feedforward and feedback control with experimental validation," *Automatica*, vol. 43, no. 1, pp. 63–71, 2007.

[14] T. Glück, A. Eder, and A. Kugi, "Swing-up control of a triple pendulum on a cart with experimental validation," *Automatica*, 2013.

[15] T. Ohtsuka, "A continuation/GMRES method for fast computation of nonlinear receding horizon control," *Automatica*, vol. 40, pp. 563–574, 2004.

[16] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*. Taylor and Francis, 1975.

[17] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995.

[18] L. Magni, G. D. Nicolao, L. Magnani, and R. Scattolini, "A stabilizing model-based predictive control algorithm for nonlinear systems," *Automatica*, vol. 37, no. 9, pp. 1351–1362, 2001.

[19] L. Magni, R. Scattolini, and K. Aström, "Global stabilization of the inverted pendulum using model predictive control," in *Proceedings of the 15th IFAC World Congress*, 2002, p. 1554.

[20] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.