Stabilization of Suspension Vehicle Near Rollover by Nonlinear Model Predictive Control

Pathompong JAIWAT * and Toshiyuki OHTSUKA **

Abstract: A suspension vehicle near rollover is controlled by nonlinear model predictive control method (NMPC), in which the continuation/generalized minimal residual (C/GMRES) is used to solve an optimal control problem in real time. The suspension vehicle near rollover can be represented by a double inverted pendulum with suspension. This kind of double inverted pendulum consists of two pendulums connected together by a nonlinear spring, which represents the axle and the body of the suspension vehicle. The terminal cost is given by a solution to the algebraic Riccati equation to make the tuning process in performance index easier. The input force to make the vehicle tip up is determined based on the surface friction coefficient and the location of the vehicle’s center of gravity. The results obtained from simulation indicated that NMPC with C/GMRES could swing up and stabilize the system successfully in real time.

Key Words: nonlinear model predictive control, suspension vehicle, vehicle rollover, double inverted pendulum.

1. Introduction

Vehicles such as sport utility vehicles (SUVs) and trucks are normally susceptible to rollover accidents. In 2005, 21.1% of the deaths due to vehicle crashes in the United States were caused by rollover [1]. It can be seen that rollover accidents are one of major causes of human fatalities in land transportation. Studies have therefore been carried out on preventing rollover accidents, such as those employing automatic steering and braking systems to keep all of the vehicle’s wheels on the ground [2],[3]. However, most of the studies have only dealt with rollover detection and preventing the vehicle from tipping up only. These studies would surely make a significant reduction in rollover accidents. Rollover accidents in real situations may be caused by many reasons, such as when a vehicle is performing high-speed maneuvers, negotiating natural terrain that is usually deformable and uneven, or running over obstacles. Rollover accidents can occur immediately and unpredictably by running over obstacles. Therefore, a controller to prevent rollover accidents should not only control vehicles on the ground but also try to prevent accidents even after they tip up.

The stabilization of vehicles near rollover (stabilization of a vehicle with two wheels while the others are in the air can be considered near rollover) is also a serious problem in ensuring the safety of passengers and their payloads, and solving this will prevent rollover accidents by stabilizing and controlling the vehicle’s roll angle. This stabilization could further be applied to a rollover prevention system that returns all of the vehicle’s wheels back to the ground safely. The combination of stabilizing vehicles after they tip up and controlling them safely to return to a normal state should reduce the number of rollover accidents in land transportation.

This paper focuses on the stabilization of vehicles after they tip up. Studies have already been carried out on stabilizing vehicles with two wheels on the ground and the other two wheels in the air. These studies used various control methods. For example, a tire contact model that was comprised of a spring-damper system modified to be continuous and piecewise differentiable was used and controlled by feedback control to stabilize the roll angle [1]. Partial feedback linearization with a method of energy-shaping control was also employed [4]. Peters et al.[4] found that the dynamics of a double inverted pendulum and a vehicle during the tip-up process share some common characteristics, and therefore inverted pendulum dynamics can be used to describe rollover dynamics.

The stabilization of a double inverted pendulum is a fundamental problem with a clear control objective and many possible control methods such as fuzzy control [5], and the use of full-state feedback and a linear quadratic regulator [6] have already been applied to this kind of system. Most conventional controllers for double inverted pendulums, such as linear feedback control, fuzzy logic control and proportional-integral-derivative (PID) control, use linear models, which have a limited range of control around unstable equilibrium points. However, model predictive control (MPC) has been proved to be more flexible for nonlinear systems and can deal with many types of constraints. Therefore, it is logical to use MPC as a controller for real-time systems.

Nonlinear model predictive control (NMPC) has been applied to stabilize vehicles during rollover [7]. However, this research used a single inverted pendulum to represent the vehicle rollover dynamics. Although successful results with real-time computation have been reported, the single inverted pendulum can only represent the vehicle as a rigid body. A rigid body vehicle considers the vehicle to be one rigid mass, in which the wheel is rigidly attached to the vehicle’s body. However, the wheels on a normal passenger vehicle are connected to the vehicle’s body by springs, shock absorbers, and linkages to im-
prove driving comfort. This connection allows relative motion between the axle and the vehicle’s body that a rigid body vehicle cannot represent.

A double inverted pendulum with suspension was used in this study to represent a suspension vehicle during rollover. This kind of double inverted pendulum consists of two pendulums: the first pendulum represents the axle and the second represents the body of the suspension vehicle. Both of the pendulums are connected by a nonlinear spring, which represents suspension between the axle and body. NMPC has previously not been applied to suspension vehicle rollover dynamics because of its extremely high computational cost. Continuation/generalized minimal residual (C/GMRES) [8] was applied in this research to reduce the computational time. SUV specifications were used for the parameters of the inverted pendulum. The main objective of this study was to swing up and stabilize a vehicle with two wheels on the ground while the others were in the air, which would prove driving comfort. This connection allows relative motion between the axle and the vehicle’s body that a rigid body vehicle cannot represent.

The main reason we used NMPC in this study was to demonstrate that the NMPC of such a highly nonlinear system could be solved in real time. Another reason is MPC has many advantages that other controllers do not have, such as MPC allowing both equality and inequality constraints to be imposed on the system, and enabling operation closer to constraints. However, the main purpose of this paper is only to demonstrate the real-time NMPC, other properties of MPC are not discussed.

This paper describes NMPC and a real-time algorithm for NMPC (C/GMRES) in Section 2. The dynamic model of a vehicle during rollover is described in Section 3. The results from simulation are reported in Section 4 and we describe and compare the results from simulation using Newton’s method and C/GMRES. Conclusion are draw in Section 5.

2. Nonlinear Model Predictive Control Problem

2.1 Problem Formulation

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

To apply MPC to 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 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, $\tau$, 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

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

where $x^*$ represents $\partial J/\partial x$, $x^*(\tau, t)$ is the trajectory along the fictitious time horizon, $\tau$, 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 to the finite-horizon optimal control problem. The actual input to the system is given by $u(t) = u^*(t, t)$. $C(x, u)$ is the equality constraint of the system. An inequality constraint can be transformed into an equality constraint by introducing dummy input as has been explained by Ohtsuka [8].

2.2 Discretized Problem

This subsection describes a method of finding optimal control on a discretized horizon. First, the horizon is divided into $N$ steps and the optimal control problem is discretized on the $\tau$-axis using the forward difference method as follows:

$$\begin{align*}
x^*_p(t, \tau) &= x^*(t) + f(x^*(t), u^*(t), p^*_p(t))\Delta \tau, \\
x^*_q(t) &= x(t), \\
C(x^*_p(t), u^*_p(t), p^*_p(t)) &= 0, \\
J &= \phi(x^*_p(\tau), p^*_p(\tau)) + \sum_{i=0}^{N-1} L(x^*_p(t), u^*_p(t), p^*_p(t))\Delta \tau.
\end{align*}$$
where $\Delta t := T/N$, $x_i'(t)$ corresponds to $x'(t + i\Delta t, t)$, and $p_i'(t)$ is given by $p(t + i\Delta t)$.

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. (4), the control input should be chosen to minimize the Hamiltonian, which is a consequence of Pontryagin’s minimum principle [9]. 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 $\mu \in \mathbb{R}^{m_u}$ is the Lagrange multiplier associated with the equality constraint. The first-order necessary conditions for the sequences of optimal control input $[u_i'(t)]_{i=0}^{N-1}$, multiplier $[\mu_i'(t)]_{i=0}^{N-1}$, and costate $[\lambda_i'(t)]_{i=0}^{N}$ along the horizon can be found by using Pontryagin’s minimum principle:

$$H_u(x_i'(t), \lambda_i'(t), u_i'(t), \mu_i'(t), p_i'(t)) = 0,$$

$$\lambda_i'(t) = \lambda_i^{N-1}(t),$$

$$H_{u \lambda}(x_i'(t), \lambda_i'(t), u_i'(t), \mu_i'(t), p_i'(t)) \Delta t,$$

$$\lambda_{0}'(t) = \phi_f(x_0', p_0'(t), t).$$

where $H_u$, $H_\lambda$, and $\phi_f$ correspond to $\partial H/\partial u$, $\partial H/\partial \lambda$, and $\partial H/\partial x$. Note that the optimal input sequences, $[u_i'(t)]_{i=0}^{N-1}$ and $[\mu_i'(t)]_{i=0}^{N-1}$, must satisfy Eqs. (1)–(7). Equations (5)–(7) are the necessary conditions for optimality. Since we want 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 input and multipliers as

$$U(t) := [u_0^T(t), \mu_0^T(t), u_1^T(t), \mu_1^T(t), ..., u_{N-1}^T(t), \mu_{N-1}^T(t)]^T,$$

where $U(t) \in \mathbb{R}^{m+}$. From Eq. (7), costate $\lambda_{0}'(t)$ depends on $x_0$. It can also be seen that, for sequence $[x_i'(t)]_{i=0}^{N}$, the future state depends on the current state. However, for costate sequence $[\lambda_i'(t)]_{i=0}^{N}$, the current costate depends on the future costate. That is, $[\lambda_i'(t)]_{i=0}^{N}$ is calculated recursively using Eqs. (1) and (2), and then $[\lambda_i'(t)]_{i=0}^{N}$ is determined recursively from $i = N$ to $i = 1$ using Eqs. (6) and (7). Since $x_i'(t)$ and $\lambda_i'(t)$ in Eqs. (3) and (5) are determined with Eqs. (1), (2), (6), and (7), Eqs. (3) and (5) can be regarded as one equation represented by a column vector $F$ as shown below:

$$F(U(t), x(t), t) :=$$

$$H_u(x_0'(t), \lambda_0'(t), u_0'(t), \mu_0'(t), p_0'(t))$$

$$C(x_0'(t), u_0'(t), p_0'(t))$$

$$\vdots$$

$$H_u(x_{N-1}'(t), \lambda_{N-1}'(t), u_{N-1}'(t), \mu_{N-1}'(t), p_{N-1}'(t))$$

$$C(x_{N-1}'(t), u_{N-1}'(t), p_{N-1}'(t))$$

$$= 0. \tag{8}$$

Solving Eq. (8) gives the optimal vector, $U(t)$, which is a sequence of inputs and Lagrange multipliers.

### 2.3 Continuation/GMRES

The equation $F(U, x, t) = 0$ (Eq. (8)) from the previous subsection, must be solved at each sampling time to solve the real-time optimal control problem. However, iterative methods should be avoided because of the high computational cost. This subsection describes an alternative method called C/GMRES, which greatly reduces the computational cost. Instead of solving Eq. (8) directly, with this method 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

$$\dot{F}(U, x, t) = A_x F(U, x, t), \tag{9}$$

where $A_x$ is a stable matrix used to stabilize $F = 0$. From the chain rule of differentiation, Eq. (9) becomes

$$\dot{F}_U = A_x F - \dot{x} F_x - F_t F_t^T.$$

If $\dot{F}_U$ is nonsingular, we obtain the following differential equation:

$$\dot{U} = F_{UU}^{-1}(A_x F - \dot{x} F_x - F_t F_t^T). \tag{10}$$

Vector $\dot{U}$ can be determined using Eq. (10) for given variables $U, x, t$, and $t$. The optimal solution, $U(t)$, can be found without the use of an iterative method of optimization by computing $U(t + \Delta t) = U(t) + \dot{U}(t) \Delta t$ in real time. However, finding $\dot{U}$ using Eq. (10) still incurs a high computational cost because of the need to find Jacobians $F_{UU}$, $F_x$, $F_t$, and $F_{UU}^{-1}$. Moreover, from the definition of vector $F$ in Eq. (8), Jacobian $F_{UU}$ is dense. Therefore, to reduce the computational cost, we employ the following techniques.

#### 2.3.1 Forward difference GMRES method

Equation (9) can be approximated to the linear equation by using forward difference GMRES (FDGMRES) to solve nonlinear equation for $\dot{U}$. For more information about FDGMRES please refer to Ohtsuka [8], and Kelly [10]. FDGMRES is combined with the continuation method for real-time computation.

#### 2.3.2 Continuation method

Vector $\dot{U}$ has been obtained with FDGMRES. $U(t)$ can be updated by integrating $\dot{U}$ in real time. For more information about this technique please refer to Ohtsuka [8].

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 to nonlinear Eq. (8) and the solution can be found without any line search from Newton’s method and without directly solving Eq. (8). Therefore, it requires much less computational time [8].

Despite using C/GMRES to find the solution to Eq. (8), the weighting matrices in Eq. (4) must be carefully selected to obtain good responses. A method of obtaining the proper weighting matrices is described in the next subsection.

### 2.4 Performance Index

This subsection discusses obtaining appropriate weighting matrices for the performance index. The main objective of the optimal control problem is to find the optimal solution to input by minimizing the performance index. The control input at each time $t$ is determined to minimize a performance index with a horizon length, $T$, as described in Subsection 2.1. To control the position of the system, the functions in performance index $J$ given by Eq. (4) are chosen as

$$\phi(x) = \frac{1}{2} (x - x_f)^T S (x - x_f), \tag{11}$$

$$L(x, u) = \frac{1}{2} ((x - x_f)^T Q (x - x_f)$$

$$+(u - u_f)^T R (u - u_f)), \tag{12}$$
where $x_f$ denotes the objective state, $u_f$ is objective input and $S_f$, $Q$, and $R$ are weighting matrices. Matrices $Q$, $R$ and $S_f$ are normally selected such that the response of the system is satisfactory. However, satisfactory responses are sometimes very hard to achieve because it is very difficult to simultaneously adjust these three weighting matrices ($S_f$, $Q$, and $R$). To reduce the variables that need to be adjusted, matrix $S_f$ (terminal weighting matrix) can be mathematically found by solving the algebraic Riccati equation, and this kind of similar ideal was previously used [11],[12].

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

$$J_{LQ} = \frac{1}{2} \int_{0}^{\infty} ((x-x_f)^T Q (x-x_f) + (u-u_f)^T R (u-u_f)) dt$$

for the linear system

$$\frac{d}{dt} (x-x_f) = A (x-x_f) + B (u-u_f).$$

This kind of problem is called a linear quadratic regulator (LQR). Optimal control is obtained as a state-feedback, $u = u_f = -K (x-x_f)$, with the gain matrix given by $K = R^{-1} B^T S_f$. Matrix $S_f$ can be found by solving the following algebraic Riccati equation.

$$A^T S_f + S_f A - S_f B R^{-1} B^T S_f + Q = 0. \quad (13)$$

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

$$\min J_{LQ} = \frac{1}{2} \int_{0}^{\infty} (x(0) - x_f)^T S_f (x(0) - x_f). \quad (14)$$

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

$$J_{LQ} = \frac{1}{2} \int_{0}^{\infty} (x(\cdot) - x_f)^T S_f (x(\cdot) - x_f) d\tau,$$

$$+ \frac{1}{2} \int_{0}^{T} (x(\cdot) - x_f)^T S_f (x(\cdot) - x_f) d\tau + \frac{1}{2} \int_{T}^{\infty} (x(\cdot) - x_f)^T S_f (x(\cdot) - x_f) d\tau.$$

Applying the basic property of LQR, Eq. (14), to the second term of the above equation and assuming that $x(t + T)$ is close to the original, the performance index can be approximated as

$$J_{LQ} = \frac{1}{2} \int_{0}^{\infty} \left( (x(t + T) - x_f)^T S_f (x(t + T) - x_f) \right) dt,$$

$$+ \frac{1}{2} \int_{0}^{T} (x(\cdot) - x_f)^T S_f (x(\cdot) - x_f) d\tau.$$

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, matrix $S_f$ is found by solving Eq. (13) offline [13].

### 3. Vehicle Suspension Model and Simulation Method

#### 3.1 Vehicle Dynamic Model

The vehicle suspension model consists of the vehicle body connected to an axle according to Fig. 1. The vehicle suspension model in this study was taken from Peters et al.[4], and the specifications are listed in Table 1.

<table>
<thead>
<tr>
<th>Description</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axle mass</td>
<td>$m_1$</td>
<td>160 kg</td>
</tr>
<tr>
<td>Body mass</td>
<td>$m_2$</td>
<td>2030 kg</td>
</tr>
<tr>
<td>Axle moment of inertia</td>
<td>$I_1$</td>
<td>102 kg m²</td>
</tr>
<tr>
<td>Body moment of inertia</td>
<td>$I_2$</td>
<td>1240 kg m²</td>
</tr>
<tr>
<td>Axle angle offset</td>
<td>$b_0$</td>
<td>0.124 rad</td>
</tr>
<tr>
<td>Axle link length</td>
<td>$l_1$</td>
<td>0.806 m</td>
</tr>
<tr>
<td>Body link length</td>
<td>$l_2$</td>
<td>0.5 m</td>
</tr>
<tr>
<td>Linear stiffness</td>
<td>$k_1$</td>
<td>7.49 × 10⁷ Nm/rad</td>
</tr>
<tr>
<td>Linear stiffness</td>
<td>$k_2$</td>
<td>0 Nm/rad³</td>
</tr>
<tr>
<td>Linear stiffness</td>
<td>$k_3$</td>
<td>2.7 × 10⁷ Nm/rad³</td>
</tr>
<tr>
<td>Linear damping</td>
<td>$h_1$</td>
<td>3200 Nm/(rad/s)</td>
</tr>
</tbody>
</table>

Since we only considered the planar vehicle model in Figs. 1 (a)–(b) in this study, the specifications can be interpreted as the parameters of a double inverted pendulum with nonlinear spring suspension. The center of gravity (C.G.) and the length of the pendulum were assumed in accordance with Table 1. Note that this system is a double inverted pendulum with a massless cart, as is shown in Fig. 1 (c).

#### 3.2 Vehicle Rollover Dynamics

The vehicle rollover dynamics can be represented using the double inverted pendulum with suspension shown in Fig. 1 [4]. However, there are several characteristics that differ between the inverted pendulum and a vehicle as follows.

1. The cart and pendulum are connected by a pin joint, while the vehicle is held against the ground by a non-negative contact force, $N$, where the left wheel acts as a pivot point, given as

$$N_f = (m_1 + m_2) g - (m_1 + m_2) l_1 \cos(\theta_1 + \theta_1 \bar{\theta}_1)$$

$$- m_2 l_2 \sin(\theta_1 + \theta_2) \bar{\theta}_1 + \bar{\theta}_2$$

$$- (m_1 + m_2) l_1 \sin(\theta_1 + \theta_1) \bar{\theta}_1 \bar{\theta}_2^2,$$

$$- m_2 \cos(\theta_1 + \theta_2) (\bar{\theta}_1 + \bar{\theta}_2)^2, \quad (15)$$

where $N_f > 0$.

2. Vehicle rollover dynamics correspond to those of the massless cart system shown in Fig. 1 (c).

3. Tire input force is generated by friction between the tires and the ground. Therefore, the magnitude of tire friction force is limited by the following inequality constraint:

$$- \mu N_f \leq u \leq \mu N_f. \quad (16)$$

4. Note that there are no practical actuators for directly generating lateral tire force. It is thus modeled as a function of lateral and longitudinal slip, which may be controlled by vehicle steering and braking systems. The relationship between vehicle steering and lateral force can be found in Jazar [14], and Abe [15].

To obtain the equations of motion for the suspension vehicle, when the right wheels are lifted off and the left wheels act as a pivot point, the coordinates about C.G. must first be defined. The coordinates for the axle mass $m_1$ are defined by $(X_{m_1}, Y_{m_1})$, and the coordinates for the body vehicle are $(X_{m_2}, Y_{m_2})$. The
coordinates about C.G. of the axle \((X_{m1}, Y_{m1})\), and the body vehicle \((X_{m2}, Y_{m2})\) are defined by the following, where as \(x\) represents the horizontal distance.

\[
\begin{align*}
X_{m1} &= x + l_1 \cos(\theta_0 + \theta_1), \\
Y_{m1} &= l_1 \sin(\theta_0 + \theta_1), \\
X_{m2} &= X_{m1} - l_2 \sin(\theta_1 + \theta_2), \\
Y_{m2} &= Y_{m1} + l_2 \cos(\theta_1 + \theta_2).
\end{align*}
\]

The state vector where we particularly have

The Lagrangian function, \(L\), 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 energy of the double inverted pendulum can be derived with the equations below.

\[
\begin{align*}
T_{m1} &= \frac{1}{2} m_1 (\dot{X}_{m1}^2 + \dot{Y}_{m1}^2) + \frac{1}{2} I_1 \dot{\theta}_1^2, \\
T_{m2} &= \frac{1}{2} m_2 (\dot{X}_{m2}^2 + \dot{Y}_{m2}^2) + \frac{1}{2} I_2 \dot{\theta}_2^2, \\
V_{m1} &= mg Y_{m1}, \\
V_{m2} &= mg Y_{m2}.
\end{align*}
\]

Then, the Lagrangian function will be \(L = T_{m1} + T_{m2} - (V_{m1} + V_{m2})\). The equation of motion can be found with the following equations:

\[
\begin{align*}
\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} &= \tau, \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_1} - \frac{\partial L}{\partial \theta_1} &= 0, \\
\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}_2} - \frac{\partial L}{\partial \theta_2} &= 0.
\end{align*}
\]

The Lagrangian equation, Eq. (17), can be rewritten in simpler matrix form:

\[
C(q) \ddot{q} + D(q, \dot{q}) \dot{q} + E(q) = \tau = Gu,
\]

where \(q = [x, \theta_1, \theta_2]^T\) and \(\tau\) represents the effect of springs and dampers in the suspension vehicle given by \(\tau = [0, 0, -k_1 \dot{\theta}_1 - k_2 \dot{\theta}_2 - b_1 \theta_1, -k_2 \dot{\theta}_2 - b_2 \theta_2]^T\). Equation (18) can be rearranged in the form, \(\dot{\mathbf{x}} = f(\mathbf{x}, u)\), where \(\mathbf{x} = [q, \dot{q}]^T = [x, \theta_1, \theta_2, \dot{x}, \dot{\theta}_1, \dot{\theta}_2]^T\) is the state vector where we particularly have

\[
f(\mathbf{x}, u) = \begin{bmatrix} \dot{\mathbf{q}} \\ C^{-1}(Gu - D\dot{q} - E + \tau) \end{bmatrix}.
\]

The vehicle model dynamics in this study were only concerned with yaw and roll motions. As the pitch dynamics, caused by longitudinal weight transfer, were neglected, the vehicle will move at a constant speed and zero acceleration [14],[16]. This study took into consideration the vehicle at the beginning of the tip-up process, thus the vehicle will only use two of its wheels. For simplicity, the vehicle model in this study was assumed to be moving at high speed, and therefore there was no Ackerman effect [16]. Moreover, the tire slip angle [14],[16] was also neglected. The dynamics in the yaw direction are derived from the bicycle model as seen in Fig. 2 [17]. The radius of the vehicle’s turn, \(r\), in Fig. 2 is approximated by

\[
r = \frac{V_L}{\sin \delta},
\]

where \(V_L\) is the length of the vehicle’s wheelbase and \(\delta\) is the steering angle. The centripetal acceleration or lateral acceleration of the vehicle is given by

\[
a_c = \frac{v^2}{r} = \frac{v^2 \sin \delta}{V_L},
\]

where \(a_c\) is centripetal acceleration and \(v\) is the forward velocity or longitudinal velocity of the vehicle.

Equation (21) gives the basic ideal on how to control and stabilize the vehicle by using the steering angle [17]. NMPC was used in this study to find sufficient lateral input force for stabilization. Then, lateral acceleration \(a_c\) can be found by using the vehicle dynamics equation. It can be seen that by using Eq. (21), the control input determined by NMPC can be converted into the necessary steering angle to stabilize the vehicle with two of its wheels.

### 3.3 Linearization

As described in Subsection 2.4, the terminal cost, Eq. (11), can be obtained by solving the algebraic Riccati equation, Eq. (13). This equation is associated with linear system dynamics

\[
\frac{d}{dt}(\mathbf{x} - \mathbf{x}_f) = \mathbf{A}(\mathbf{x} - \mathbf{x}_f) + \mathbf{B}(u - u_f).
\]

The objective or final state, \(\mathbf{x}_f\), must first be defined before simulation. The input and the angular velocities of the vehicle equal zero, \(u = 0, i = 0, \dot{\theta}_1 = 0, \) and \(\dot{\theta}_2 = 0\) at the unstable equilibrium point. Therefore, the angle of the axle and the body of the rollover vehicle, \(\theta_1\) and \(\theta_2\), can be found at the final state or at the unstable equilibrium point by setting Eq. (19) to equal zero \((f(\mathbf{x}_f, 0) = 0)\). The results after solving equation \(f(\mathbf{x}_f, 0) = 0\) show that the equilibrium point of the system is at \(\theta_1 = 0.933\) rad and \(\theta_2 = 0.101\) rad.

Equation (19) can be linearized to derive an approximate linear solution around upward equilibrium point \(\mathbf{x}_f = [0, 0.933, 0.101, 0, 0, 0]^T\); this yields:

\[
A = \begin{bmatrix} 0 & 0 & 0 \\ -C(\mathbf{x}_f)^{-1} \frac{\partial E(\mathbf{x})}{\partial \mathbf{q}} & I & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ C(\mathbf{x}_f)^{-1}G \end{bmatrix}.
\]

where I is the identity matrix. After matrix A and B are obtained, the terminal weighting matrix, \(S_f\), in Eq. (11) can be determined by solving Eq. (13).

### 3.4 Modified C/GMRES

This subsection discusses the modifications to NMPC and C/GMRES. Input is determined to minimize a performance index with a receding horizon length, \(T\), as described in Subsection 2.1. The system must be discretized to solve the NMPC problem according to Subsection 2.2. The performance index
becomes Eq. (4) after discretization. Then, weighting matrix $S_f$ can be found according to Subsections 2.4 and 3.3. C/GMRES is applied to find the solution to Eq. (4) to achieve the real-time calculation.

The system begins its motion at $t = 0$ with the given initial state of the pendulum, and then the controller finds the optimal input sequence, $U$, by solving Eq. (8) directly or by applying C/GMRES. Since input to the system is constrained by the inequality constraint, $-\mu N_f \leq u \leq \mu N_f$, where $\mu$ is the friction coefficient and $N_f$ is the non-negative contact force according to Eq. (15), the magnitude of input is bounded by

$$ u_{\text{max}} = \mu N_f. $$

(23)

Variable $N$ in Eq. (23) depends on $\theta_1$ and $\tilde{\theta}_2$, and it can be seen that $\theta_{1,2}$ depends on $\theta_{1,2}$, $\theta_{1,2}$, and $u$. This means that the input constraint or the bound of input depends on $\theta_{1,2}$, $\theta_{1,2}$, and $u$ through $\theta_{1,2}$, as $u_{\text{max}}(x(t), u_0(t))$. Variable $u_0$ is the first component of vector $U$. Since the input bound depends on input itself, finding the optimal solution becomes more difficult, increasing the computational time and occasionally making the computation fail. Therefore, instead of solving the optimal control problem with the input inequality constraints, control input is optimized with no constraints and, before input is applied to the system, the input, $u$, is checked as to whether the condition, $|u| > |u_{\text{max}}|$, holds or not. If the input value is outside the constraint limit, input $u_0$ is set to equal $u_{\text{max}}$ or $-u_{\text{max}}$, and $u_0$ remains rechecked until $u_0$ is bounded by $u_{\text{max}}$. When the constraint is satisfied, input is used as actual input to the system positions, and then the system uses the new positions to find the optimal solution again as shown in Algorithm 1.

Algorithm 1 Finding input with constraint

1. At time $t$, measure state $x(t)$.
2. Find $\dot{U}(t)$ with GMRES.
3. Find $\dot{U}(t)$ by integrating $\dot{U}(t)$.
4. Calculate $u_{\text{max}}(x(t), u_0(t))$.
5. While $|u_0| > |u_{\text{max}}|$
   (a) $u_0 := u_{\text{max}}(x(t), u_0(t))$.
6. Apply $u_0$ to the system.
7. $t := t + \Delta t$ and go to step 1.

Note that the value of $u_0$ is rechecked by using while loop as described in Algorithm 1. Although there is no guarantee that $|u_0| \leq |u_{\text{max}}|$, the condition was never been violated in the simulation will be explained later. Moreover, when input becomes saturated the optimality of the system is lost. However, this simplification makes the algorithm executable in real time, as indicated by the simulation results presented in the next section.

4. Simulation Results

This section presents the results obtained from simulation while adjusting weighting matrices $R$ and $Q$ in the performance index, Eq. (12). Matrix $S_f$ is obtained by solving Eq. (13). Table 1 summarizes the model parameters we used in the simulation.

The sampling period in the simulation was 0.01 s and the horizon length was divided into 10, 5, and 2 steps ($N = 10, 5, 2$). The parameters associated with the horizon were chosen as $T = T_f(1 - e^{-\alpha})$, $T_f = 1$ s, and $\alpha = 0.5$. The state vector was defined as $x = [x, \dot{x}, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_1, \dot{\theta}_2]^T$, and this simulation was focused on stabilizing the vehicle’s axle roll angle, $\theta_1$, it was important to stabilize the vehicle’s axle angle to reach the unstable equilibrium point. Therefore, the angle was given the largest weight. However, to solve the algebraic Riccati equation for terminal weighting matrix $S_f$, the weighting value that corresponds to the horizontal direction cannot be zero. Therefore, a small value was set to the weighting for the horizontal direction. The weighting matrices of the performance index were chosen as: $Q = \text{diag}[1 \times 10^{-2}, 1 \times 10^{5}, 0, 0, 0, 0]$ and $R = 0.01$. The terminal weighting matrix $S_f$ was obtained by solving Eq. (13). The objective state was defined as $x_f = [0, 0.933, 0.101, 0, 0, 0]^T$. The initial state was chosen near the tip-up point; thus, the initial state of the system was $x_0 = [0, 0, 0, 0, 0, 0]^T$. The friction coefficient, $\mu$, was assumed to be 1.5. The error in the optimality condition is represented by the norm of the function $F$ in Eq. (8), $\|F\|$, upon substituting 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. The nonlinear equation Eq. (8) was written in MEX (MATLAB Executable) file to enable faster simulation. A MEX-file provides an interface between MATLAB and subroutines written in C/C++. The transformation between MATLAB’s M-file to MEX-file was simply done by using command mex in MATLAB.

Figure 3 compares between different terminal weighting matrices, $S_{f1}$, where the choices of $S_f$ were determined as follows. It was reasonable to set terminal weighting matrix $S_{f1} = Q$ because the terminal cost should focus on the same state variables as the weighting matrix, $Q$, in the performance index. It was also reasonable to make the system focus on the roll angle and roll rate of the vehicle ($\dot{\theta}_1$ and $\dot{\theta}_1$) and ignore the horizontal distance ($x$), and hence $S_{f2} = \text{diag}[0, 1 \times 10^{3}, 0, 0, 1 \times 10^{3}, 0]$. The effects of ignoring the terminal cost in the performance index is found by setting matrix $S_{f3} = 0$. Finally, $S_{f4}$ is obtained by solving Eq. (13), which results in

$$ S_{f4} = \begin{bmatrix} 0.876 & -1.198 \times 10^{2} & -9.092 \times 10^{3} \\ -1.20 \times 10^{2} & 4.443 \times 10^{6} & 3.293 \times 10^{6} \\ -9.092 \times 10^{4} & 3.293 \times 10^{6} & 3.010 \times 10^{10} \\ 3.837 \times 10^{1} & -1.042 \times 10^{3} & -7.923 \times 10^{3} \\ -7.529 \times 10^{4} & 7.170 \times 10^{3} & 5.779 \times 10^{3} \\ -4.403 \times 10^{4} & 5.922 \times 10^{3} & 4.944 \times 10^{3} \\ 3.837 \times 10^{1} & -7.529 \times 10^{4} & -4.403 \times 10^{4} \\ -1.042 \times 10^{4} & 7.170 \times 10^{4} & 5.922 \times 10^{4} \\ -7.923 \times 10^{3} & 5.779 \times 10^{3} & 4.944 \times 10^{5} \\ 3.349 \times 10^{1} & -6.576 \times 10^{3} & -3.847 \times 10^{3} \\ -6.576 \times 10^{1} & 1.7249 \times 10^{3} & 1.491 \times 10^{3} \\ -4.403 \times 10^{4} & 5.922 \times 10^{3} & 1.324 \times 10^{5} \end{bmatrix} $$

The results in Fig. 3 were obtained by using Newton’s method with discretization steps of $N = 10$. It can be seen that $S_{f4}$ yields results with the best time responses which have the fastest convergence and low error. It can be seen from the results in Fig. 3 that if matrix $S_f$ is freely adjusted, satisfactory
results are hard to achieve.

The black dark solid line in Fig. 4 plots the results obtained by directly solving Eq. (8) using Newton’s method. Newton’s method was computed in this simulation by using the trust-region dogleg algorithm [18]. The results revealed that NMPC successfully stabilized the vehicle to reach the equilibrium point, $\theta_1 = 0.933$ rad and $\theta_2 = 0.101$ rad. The error graph has a very small value except when input reaches the input constraint, which means that the algorithm has the ability to find a solution to NMPC and stabilize the vehicle near rollover.

Figure 4 also compares Newton’s method and C/GMRES with discretization steps of $N = 10$, $5$, and $2$. The results obtained with Newton’s method and C/GMRES at $N = 10$ are plotted as a black solid line and a pink solid line. It can be seen that the time responses with Newton’s method and C/GMRES at $N = 10$ are almost the same. Newton’s method was used as a reference to evaluate the time responses, because it provided the least error value when the constraint was not violated. The simulation results obtained with C/GMRES at $N = 5$ and $N = 2$

<table>
<thead>
<tr>
<th>Method</th>
<th>MATLAB</th>
<th>MATLAB &amp; MEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Newton’s method</td>
<td>0.2640</td>
<td>0.0570</td>
</tr>
<tr>
<td>C/GMRES</td>
<td>0.0311</td>
<td>0.0070</td>
</tr>
</tbody>
</table>

Table 2: Average computational time per update (s) for discretization steps of $N = 10$. are plotted as orange and blue dashed lines. It can be seen that C/GMRES with five and two discretization steps can also stabilize the system. Another trade-off between these two methods is the computational cost. The computational cost is presented in Table 2 and Fig. 5.

Table 2 and Fig. 5 indicate the computational times per update for Newton’s method and C/GMRES when the discretization steps of both methods were equal to 10 steps, $N = 10$. Table 2 and Fig. 5 also compare the computational times between M-file and MEX-file. The results indicate that C/GMRES incurred much lower computational cost than Newton’s method and was even lower when MEX-file was used. It can be seen that the computational time for Newton’s method gradually
reduced and became constant when the system reaches equilibrium, in contrast to C/GMRES in which the computational time was almost the same throughout the entire simulation time. Figure 5 also indicates that C/GMRES with MEX-file had a very short computational time and it can be seen from Table 2 that the average computational cost of C/GMRES with MEX-file is lower than 0.01 s of sampling time. Therefore, C/GMRES can be used in real-time situations.

Figure 6 plots the time history of input obtained with Newton’s method with the input constraint given by Eq. (23). If input is saturated, error will be extremely high, according to the error graph in Fig. 4 because the optimal input during the saturation period will be replaced by $u_{\text{max}}$. Therefore, optimality is lost during that period.

The required steering angle for the vehicle near rollover stabilization can be determined by using Eq. (21) and the simulation’s lateral acceleration as shown in Fig. 7, whereas the longitudinal velocity is equal to 80 km/h and the vehicle’s wheel base is equal to 2.9 m. It can be seen that the steering angle in Fig. 7 is within a realistic range and can be applied to a real system.

Figure 8 shows the regions of attraction of NMPC obtained by using C/GMRES. These regions were obtained by varying initial angle $\theta_1$ and initial angular velocity $\dot{\theta}_1$. We found from the simulation that the regions of attraction between Newton’s method and C/GMRES were approximately the same. It can be seen from Fig. 8 that there are unstable points inside the stable region at around $\theta_1 \in [0.1, 0.15]$ and $\dot{\theta}_1 \in [0.05, 0.35]$. This region of attraction reflects the nonlinearity of the system.

Figure 9 compares the multi-body vehicle and rigid-body ve-
Fig. 5 Time histories of computational time per update between the Newton’s method and C/GMRES for discretization steps of $N = 10$.

Fig. 6 Input and bound obtained with Newton’s method.

Fig. 7 Necessary steering angle to stabilize vehicle near rollover with constant longitudinal velocity of 80 km/h.

The dynamics of a suspension vehicle during rollover can be represented by a double inverted pendulum with suspension. An optimal control problem for the suspension vehicle was discretized over the horizon, and a nonlinear two point boundary value problem was formulated to find the sequence of optimal control inputs. Pontryagin’s minimum principle provided the necessary equations to optimize the performance index. The appropriate terminal weighting matrix in terminal cost could be chosen by solving algebraic Riccati equation to obtain the best time response. Optimal input was found by solving a nonlinear equation. Newton’s method is usually used to solve this equation, which is extremely time-consuming. C/GMRES was employed in this study, and the simulation results revealed that this method significantly reduced the computational time. They also indicated that C/GMRES yielded almost the same results as those in Newton’s method, which stabilized the vehicle near rollover; therefore, C/GMRES can be applied to real-time situations.

5. Conclusions

The dynamics of a suspension vehicle during rollover can be represented by a double inverted pendulum with suspension. An optimal control problem for the suspension vehicle was discretized over the horizon, and a nonlinear two point boundary value problem was formulated to find the sequence of optimal control inputs. Pontryagin’s minimum principle provided the necessary equations to optimize the performance index. The appropriate terminal weighting matrix in terminal cost could be chosen by solving algebraic Riccati equation to obtain the best time response. Optimal input was found by solving a nonlinear equation. Newton’s method is usually used to solve this equation, which is extremely time-consuming. C/GMRES was employed in this study, and the simulation results revealed that this method significantly reduced the computational time. They also indicated that C/GMRES yielded almost the same results as those in Newton’s method, which stabilized the vehicle near rollover; therefore, C/GMRES can be applied to real-time situations.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Scientific Research (B) (No. 24360165) from the Japan Society for the Promotion of Science.

References


---

**Pathompong Jaiwat**

He received his B.Eng. from the Sirindhorn International Institute of Technology, Thammasat University, Thailand, in 2009. He received his M.Eng. from Osaka University, Japan, in 2012. From 2008 to 2010, he worked as an teaching assistant at the Sirindhorn International Institute of Technology. He is currently a Ph.D student at the Graduate School of Engineering Science, Osaka University, Japan. His research interests include linear and nonlinear control systems, nonlinear model predictive control, and real-time optimization.

---

**Toshiyuki Ohtsuka (Member)**

He received his B.Eng., M.Eng. and D.Eng. from Tokyo Metropolitan Institute of Technology, Japan, in 1990, 1992 and 1995. From 1995 to 1999, he worked as an Assistant Professor at the University of Tsukuba. In 1999, he joined Osaka University, and he was a Professor at the Graduate School of Engineering Science from 2007 to 2013. In 2013, he joined Kyoto University as a Professor at the Graduate School of Informatics. His research interests include nonlinear control theory and real-time optimization with applications to aerospace engineering and mechanical engineering. He is a member of IEEE, ISCIE, JSME, JSASS, and AIAA.