Summary

This document is a hands-on of Maple worksheet for automatic code generation and simulation system for nonlinear model predictive control (NMPC). This method is well known as a fast algorithm and expected to real-time implementation in a system. This describes the components and usage of the automatic code generation system named "AutoGenU for Maple". This worksheet is required Maple 17 or later version and C compiler (Microsoft Visual C++ compiler or GNU C compiler).

Background

Nonlinear model predictive control (NMPC) approach has attracted attention in recent years. Continuation method combined with generalized minimal residual method (C/GMRES) [1] is well known as a fast algorithm and expected to real-time implementation. This worksheet is symbolic computation tool with automatic code generation of the easy-to-use nonlinear predictive control design environment based on C/GMRES.

Algorithm of NMPC

In this section, the nonlinear model predictive control problem is briefly summarized (See [1-7] in detail). NMPC, which is also known as receding horizon control, minimizes at each time a performance index with a moving horizon and results in a feedback control law (See Figure 1).

Let us consider a general nonlinear system governed by the state equation

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

where \( x(t) \in \mathbb{R}^n \) denote the state vector, \( u(t) \in \mathbb{R}^m \) denotes the control input vector, and
denote the state vector. The control input at each time \( t \) is determined so as to minimize a performance index with a receding horizon:

\[
J := \varphi(x(t + T), p(t + T)) + \int_t^{t+T} L(x(\tau), u(\tau), p(\tau)) \, d\tau.
\] (2)

At each time \( t \), the optimal control minimizing \( J \) is determined over the horizon \( [t, t + T] \), and its initial value at time \( t \) is used as the actual input to the controlled system. The length of the horizon \( T \) is given as a function of time \( T = T(t) \) in general. Equality constraints are also imposed over the horizon in general as

\[
C(x(t), u(t), p(t)) = 0,
\] (3)

where \( C \) is an \( m_c \)-dimensional vector-valued function. In the case of an inequality constraint, it is necessary to introduce a penalty function method or some heuristic modification of the problem. The optimal control \( u_{opt} \) is determined as a function over the horizon and also depends on \( t \) and \( x(t) \) as \( u_{opt}(\tau; t, x(t)) \), \( (\tau \in [t, t + T]) \). However, the actual input to the system is given only by the value of \( u_{opt} \) at time \( t \), that is, \( u(t) = u_{opt}(t; t, x(t)) \), which is a state feedback control law.

Let \( H \) denote the Hamiltonian defined by

\[
H(x, \lambda, u, \mu, p) := L(x, u, p) + \lambda^T f(x, u, p) + \mu^T C(x, u, p)
\] (4)

where \( \lambda \in \mathbb{R}^n \) denotes the costate, and \( \mu \in \mathbb{R}^n \) denotes the Lagrange multiplier associated with the equality constraint. Using Hamiltonian, it is replaced the following expanded performance index:

\[
J := \varphi(x(t + T), p(t + T)) + \int_t^{t+T} \left( H(x, \lambda, u, \mu, p) - \lambda^T \dot{x} \right) d\tau
\] (5)

Then, the following Euler-Lagrange equations are obtained as the necessary conditions for the optimal solution.

\[
\dot{x}(t) = f(x(t), u(t), p(t)),
\] (6)

\[
x(t) = x_0(t),
\] (7)

\[
\dot{\lambda}(t) = -H^T_x \left( x(t), \lambda(t), u(t), \mu(t), p(t) \right),
\] (8)

\[
\lambda(t + T) = \varphi_x^T (x(t + T), p(t + T)),
\] (9)

\[
H_u(x(t), \lambda(t), u(t), \mu(t), p(t)) = 0,
\] (10)

\[
C(x(t), u(t), p(t)) = 0,
\] (11)

where a suffix denotes a partial derivative. Dividing the horizon into \( N \) step, it is discretized the optimal control problem on the \( \tau \)-axis with the forward difference as follows:
\[ x_{i+1}^*(t) = x_i^*(t) + f(x_i^*(t), u_i^*(t), p_i^*(t))\Delta t, \quad (12) \]

\[ x_0^*(t) = x(t) \quad (13) \]

\[ \lambda_i^*(t) = \lambda_{i+1}^*(t) + H_x^u(x_i^*(t), \lambda_{i+1}^*(t), u_i^*(t), \mu_i^*(t), p_i^*(t))\Delta t, \quad (14) \]

\[ \lambda_N^*(t) = \varphi_T^X(x_N^*(t), p_N^*(t)) \quad (15) \]

\[ H_u^u(x_i^*(t), \lambda_{i+1}^*(t), u_i^*(t), \mu_i^*(t), p_i^*(t)) = 0, \quad (16) \]

\[ C(x_i^*(t), u_i^*(t), p_i^*(t)) = 0 \quad (17) \]

where \( \Delta t := \frac{T}{N} \), \( x_i^*(t) \) (\( i = 0, \ldots, N - 1 \)) corresponds to \( x^*(i\Delta t, t) \) starting from \( x(t) \) at \( i = 0 \), \( p_i^*(t) \) is given by \( p(t + i\Delta t), u_i^*(t) \) (\( i = 0, \ldots, N - 1 \)) is optimal control sequence, \( \mu_i^*(t) \) (\( i = 0, \ldots, N - 1 \)) is multiplier sequence, and \( \lambda_i^*(t) \) (\( i = 0, \ldots, N \)) is costate sequence. The sequences of the optimal control and the multiplier must satisfy equations (12)-(17), which define a two-point boundary value problem (TPBVP) for the discretized optimal control problem. It should be noted that the TPBVP for the discretized problems is identical to a finite difference approximation of the TPBVP for the original continuous-time problem. Therefore, the solution of the discretized problem converges to the solution of the continuous-time problem as \( N \to \infty \) under mild conditions.

Let us define a vector of the input and multipliers as

\[ U(t) := [u_0^T(t), \mu_0^T(t), \ldots, u_N^T(t), \mu_N^T(t)]^T, \]

\[ P_0(t) = u_0^*(t) \]

where \( m := m_u + m_\mu \) and also define a projection \( P_0 : \mathbb{R}^m \times N \to \mathbb{R}^m \) as \( P_0(U) := u_0^* \). Given \( U(t) \) and \( x(t) \), \( x_i^*(t) \) (\( i = 0, \ldots, N \)) is calculated recursively by (12) and (13), and then \( \lambda_i^*(t) \) (\( i = 0, \ldots, N \)) is also calculated recursively from \( i = N \) to \( i = 0 \) by (14) and (15). Since \( x_i^*(t) \) and \( \lambda_i^*(t) \) are determined by \( x(t) \) and \( U(t) \) through (12)--(15), equations (16) and (17) can be regarded as a nonlinear algebraic equation defined as

\[ (18) \]
The equation also depends on time $t$ through $T$ and $\Delta t$ in general. If the equation is solved with respect to $U(t)$ for the measured $x(t)$ at each time $t$, then the control input $u(t) = P_0(U(t))$ is determined. If the time-dependent variation of the optimal solution can be traced exactly, iterative methods for solving equation (18). It is employed another equivalent condition to trace the time-dependent solution as follows:

\[
F(U(t), x(t), t) := \begin{bmatrix}
H_u^T\left(x_0^*(t), \lambda_1^*(t), u_0^*(t), \mu_0^*(t), p_0^*(t)\right) \\
C\left(x_0^*(t), u_0^*(t), p_0^*(t)\right) \\
\vdots \\
H_u^T\left(x_{N-1}^*(t), \lambda_{N-1}^*(t), u_{N-1}^*(t), \mu_{N-1}^*(t), p_{N-1}^*(t)\right) \\
C\left(x_{N-1}^*(t), u_{N-1}^*(t), p_{N-1}^*(t)\right)
\end{bmatrix} = 0.
\]

The equation also depends on time $t$ through $T$ and $\Delta t$ in general. If the equation is solved with respect to $U(t)$ for the measured $x(t)$ at each time $t$, then the control input $u(t) = P_0(U(t))$ is determined. If the time-dependent variation of the optimal solution can be traced exactly, iterative methods for solving equation (18). It is employed another equivalent condition to trace the time-dependent solution as follows:

\[
\dot{F}(U, x, t) = -\zeta F(U, x, t) \quad (\zeta > 0) \tag{19}
\]

\[
F(U(0), x(0), 0) = 0 \tag{20}
\]

where the right-hand side in the first equation is added to stabilize $F = 0$. If Jacobian $F_U$ is nonsingular, it is obtained a differential equations of $U(t)$ as

\[
\dot{U} = F_U\left(-\zeta F - F_x \dot{x} - F_t\right) \tag{21}
\]

which can also be regarded as a linear algebraic equation with a coefficient matrix $F_U$ to determine $\dot{U}$ for given $U$, $x$, $\dot{x}$ and $t$. Then, $U(t)$ can be updated without any successive approximation by integrating equation (21) in real time, which is a kind of the continuation method [8]. Since equation (21) still involves numerically expensive operations to solve the linear equation associated with $F_U^{-1}$, it is employed generalized minimum residual method (GMRES) (See [9]), which is a kind of Krylov subspace methods and is efficient for large-scale linear equations. Moreover, the products of the Jacobians and some $W \in \mathbb{R}^{mN}$, $w \in \mathbb{R}^r$ and $\omega \in \mathbb{R}$ are approximated with the forward difference as follows:

\[
F_U(U, x, t) W + F_x(U, x, t) w + F_t(U, x, t) \omega \\
= D_h F\left(U, x, t : W, w, \omega\right) := \frac{F(U + hW, x + hw, t + h\omega) - F(U, x, t)}{h},
\]

where $h$ is a positive real number. There is no need to calculate the Jacobians and the linear equation iteratively in this method because of using a forward difference approximation for the product the Jacobians. Therefore, the method assures the real time optimal control ability because of small computational cost.

\section*{Automatic Code Generation System}

The automatic code generating system, named "AutoGenU for Maple", is developed to generate simulation programs using C/GMRES algorithm in C language. This system has not only code generation but also compile and simulation. This system consists of three files: "rhfuncu.c", "rhfuncu.h", and "rhfuncu.m".
"rhmainu.c", "AutoGenU.mw" ("AutoGenU_CMD.mw") and "NMPCPackage.mla" including
startup functions for the NMPC worksheet.

The C source file "rhfuncu.c" defines some fundamental functions for matrix operations and
numerical integration that are used commonly for any NMPC problems. The file "rhmainuc.c"
defines the main loop for simulation. Given parameters for defining the state equation, the
performance index and other conditions in the Maple worksheet, it is created a C source file, called a
C compiler command and showed graph of results in the same worksheet. The generated source file
includes "rhfuncu.c" and "rhmainu.c" when it is compiled by a compiler.

After setting a current directory in "Initialize" section, input the setting parameters for problem and
simulation in "Define Setting Parameters" and setting compiler in "Run Simulation" section in the
worksheet, it is automatically is starting to generate C code, run simulation and show graphs for the
result data when a button to execute all commands in the worksheet is clicked.

A list of setting parameters in "Define Setting Parameters" section is shown in Table 1.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimx</td>
<td>Dimension of the state vector</td>
</tr>
<tr>
<td>dimu</td>
<td>Dimension of the control input vector</td>
</tr>
<tr>
<td>dimc</td>
<td>Number of Constraints</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>dimp</td>
<td>Dimension of the vector of time-variant parameters</td>
</tr>
<tr>
<td>fxu</td>
<td>State equation: $\dot{x} = f(x, u, p)$</td>
</tr>
<tr>
<td>L</td>
<td>Integrand in the performance index: $L(x, u, p)$</td>
</tr>
<tr>
<td>Cxu</td>
<td>Constraints: $C(x, u, p) = 0$</td>
</tr>
<tr>
<td>pt</td>
<td>Terminal Penalty: $p(t)$</td>
</tr>
<tr>
<td>MyVarNames</td>
<td>List of variable names defined by user</td>
</tr>
<tr>
<td>MyVarValues</td>
<td>List of values for user's variables</td>
</tr>
<tr>
<td>MyArrNames</td>
<td>List of array names defined by user</td>
</tr>
<tr>
<td>MyArrDims</td>
<td>List of dimensions of user's arrays</td>
</tr>
<tr>
<td>MyArrValues</td>
<td>List of values for user's arrays</td>
</tr>
<tr>
<td>tsim0</td>
<td>Initial time of simulation</td>
</tr>
<tr>
<td>tsim</td>
<td>Final time of simulation</td>
</tr>
<tr>
<td>tf</td>
<td>Final horizon length: $T_f$</td>
</tr>
<tr>
<td>ht</td>
<td>Time step in simulation</td>
</tr>
<tr>
<td>alpha</td>
<td>Parameter for variable horizon: $T = T_f (1 - e^{-\alpha t})$</td>
</tr>
<tr>
<td>zeta</td>
<td>Parameter for stabilization of continuation method: $\dot{F}(U, x, t) = -\zeta F(U, x, t)$ ($\zeta &gt; 0$)</td>
</tr>
<tr>
<td>x0</td>
<td>Initial state: $x_0$</td>
</tr>
<tr>
<td>u0</td>
<td>Initial Guess for initial control input and multipliers: $u_0$</td>
</tr>
<tr>
<td>hdir</td>
<td>Step in the forward difference approximation for the products of the Jacobians and vectors: $h$ which is used $D_h F(U, x, t: W, w, \omega)$ $D_h F(U, x, t: W, w, \omega)_h := \frac{F(U + hW, x + hw, t + h\omega) - F(U, x, t)}{h}$</td>
</tr>
<tr>
<td>rtol</td>
<td>Tolerance of error in initial control input and multipliers, $u_0$ (rtol = $10^{-6}$)</td>
</tr>
<tr>
<td>kmax</td>
<td>Number of iteration in GMRES</td>
</tr>
<tr>
<td>dv</td>
<td>Number of grids on the horizon</td>
</tr>
<tr>
<td>dstep</td>
<td>Step for saving data</td>
</tr>
</tbody>
</table>
Table 1. Define setting parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>outfn</td>
<td>Filename of C source file</td>
</tr>
<tr>
<td>fndat</td>
<td>Header of data filenames</td>
</tr>
<tr>
<td>SimplifyLevel</td>
<td>Specification of simplify</td>
</tr>
<tr>
<td>Precondition</td>
<td>Specification of preconditioning</td>
</tr>
</tbody>
</table>

If "SimplifyLevel" > 0, Maple command: "simplify" is used and the command tries to generate the simplest expressions to apply simplification rules to an expression. "Precondition" is a parameter for preconditioning by the Hessian of the Hamiltonian. If "Precondition" > 0, it is used preconditioning by the Hessian of the Hamiltonian.

### Examples

We will introduce two examples for using the NMPC worksheet in this section.

#### Position control problem of hovercraft

Let us consider for a position control problem of the hovercraft as a numerical example using this worksheet (See [6, 7] in the detail).

![Figure 3. Hovercraft](image)

The equations of motion for hovercraft model is given by

\[
M \ddot{x} := u_1 \cos(\theta) + u_2 \cos(\theta),
\]

\[
M \ddot{y} := u_1 \sin(\theta) + u_2 \sin(\theta),
\]
where \((x, y)\) is center of mass, \(\theta\) is attitude angle, and \(u_i\) \((i = 1, 2)\) is thrusts with a constraint \(u_{\min} \leq u_i \leq u_{\max}\), \(r\) is the distance between a thruster and the center line of the hovercraft, and \(M\) and \(I\) are the mass and moment of inertia of the hovercraft, respectively (Figure 3).

A formulation for the problem can be given by the following equation:

\[
\ddot{\theta} := u_1 r - u_2 r,
\]

where \((x, y, \theta)\) is center of mass, \(\dot{x}, \dot{y}, \theta\) is attitude angle, and \(u_i\) \((i = 1, 2)\) is thrusts with constraints \(u_{\min} \leq u_i \leq u_{\max}\), \(r\) is the distance between a thruster and the center line of the hovercraft, and \(M\) and \(I\) are the mass and moment of inertia of the hovercraft, respectively.

A formulation for the problem can be given by the following equation:

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{x}_4 \\
\dot{x}_5 \\
\dot{x}_6 
\end{bmatrix} =
\begin{bmatrix}
\dot{x}_4 \\
\dot{x}_5 \\
\dot{x}_6 \\
\frac{u_1 \cos x_3 + u_2 \cos x_3}{M} \\
\frac{u_1 \sin x_3 + u_2 \sin x_3}{M} \\
\frac{u_1 r + u_2 r}{I}
\end{bmatrix}
\tag{22}
\]

where \(x = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}]^T \in \mathbb{R}^6\) is a status vector. Using dummy input values \(u_{di} \ (i = 1, 2)\), equality constraints are written by

\[
C_i = (u_i - \bar{u})^2 + u_{di}^2 - (u_{\max} - \bar{u})^2,
\tag{23}
\]

where \(\bar{u} = \frac{u_{\max} + u_{\min}}{2}\). Then, the extended performance index \(J\) including the dummy values is given by

\[
J = \frac{1}{2} \left( x(t + T) - x_j \right)^T S_f \left( x(t + T) - x_j \right)
+ \frac{1}{2} \int_t^{t+T} \left( (x - x_j)^T Q (x - x_j) + u^T R u - k_1 u_{d1} - k_2 u_{d2} \right) \, dt,
\tag{24}
\]

where \(x_j\) is a target position, \(S_f, Q\) and \(R\) are weighting matrices, \(k_i \ (i = 1, 2)\) are weighting coefficients.

The NMPC worksheet (AutoGenU for Maple) consists of the following section.

- **Introduction**
- **Initialize**
- **Define Setting Parameters**
- **Function for C Code Generation**
- **Generate Euler-Lagrange Equations**
- **Generate C Code**
The worksheet just needs to setting the following contents (See the next section).

- Setting current directory (in "Initialize" section)
- Definition of setting parameters in Table 1 (in "Define setting parameters" section)
- Selection of C compiler (in "Run Simulation" section)

Comparing with equations (1)--(3) and (22)--(24), main functions are given by

\[
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\dot{x}_3 \\
\dot{x}_4 \\
\dot{x}_5 \\
\dot{x}_6 \\
\end{bmatrix} =
\begin{bmatrix}
\dot{x}_4 \\
\dot{x}_5 \\
\dot{x}_6 \\
\frac{u_1 \cos x_3 + u_2 \cos x_3}{M} \\
\frac{u_1 \sin x_3 + u_2 \sin x_3}{M} \\
\frac{u_1 r + u_2 r}{I}
\end{bmatrix},
\]

(25)

\[
C = \begin{bmatrix}
(u_1 - \bar{u})^2 + u_3^2 - (u_{\text{max}} - \bar{u})^2 \\
(u_2 - \bar{u})^2 + u_4^2 - (u_{\text{max}} - \bar{u})^2
\end{bmatrix}
\]

(26)

\[
\varphi = \frac{1}{2} e^T S_f e,
\]

(27)

\[
L = \frac{1}{2} \left( e^T Q e + u^T R u - k_1 u_3 - k_2 u_4 \right).
\]

(28)

where \(e = x - x_f\) is a temporary vector, \(u = [u_1, \ldots, u_4] \in \mathbb{R}^4\) is an input vector, and \(k_i (i = 1, 2)\) are weighting parameters for dummy input values. The target position is chosen as \(x_f = [0, 0, 0, 0, 0, 0]^T\), these weighting matrices are chosen as \(S_f = \text{diag}[10, 15, 0.1, 1, 1, 0.01]\), \(Q = \text{diag}[10, 15, 0.1, 1, 1, 0.01]\) and \(R = \text{diag}[1, 1, 0.001, 0.001]\). The terminal penalty \(p(t)\) is given zero vector in this case. The horizon \(T\) is given by the following function of \(t:\)

\[T = T_f (1 - e^{-\alpha t})\]

so that the initial condition of \(\dot{x}(t)\) is given by trivial solution of (19) for \(T = 0\), \(T_f = 1.0\) [s] and \(\alpha = 1.0\). The parameter \(\zeta\) in (19) for stabilization \(F = 0\) is given by \(\zeta = 120.0\). The time step on the \(t\) axis is 0.01[s], and the horizon on the \(\tau\) axis is divided in 10 steps. The initial state is given in the simulation as \(x(0) = [0.25, 0.35, 0, 0, 0, 0]^T\). The initial \(\text{ Other parameters are given as}\)
This is a brief explanation of new maple worksheet: "AutoGenU_SAD.mw" as another example. This is the additional worksheet for an example of semi-active dumper system which is treated with in [3].

Let us consider for a semi-active dumper system as the following (See [3] in the detail).

The equations of motion for semi-active dumper is given by

\[ m \ddot{y} + u(t) \dot{y} + k y(t) = 0, \]

where \( y(t) \) is a displacement of mass point, \( m \) is a mass of the mass point, \( k \) is a spring constant and \( u(t) \) is a attenuation coefficient with a constraint \( 0 \leq u \leq u_{\text{max}} \) and \( u_{\text{max}} > 0 \) (See Figure 4). A formulation for the problem can be given by the following equation:

\[
\begin{bmatrix}
\dot{x}_1(t) \\
\dot{x}_2(t)
\end{bmatrix} = \begin{bmatrix}
x_2(t) \\
- \frac{k}{m} x_1(t) - \frac{1}{m} x_2(t) u(t)
\end{bmatrix}, \tag{29}
\]

where \( x = [x_1, x_2]^T = [y, \dot{y}]^T \in \mathbb{R}^2 \) is a status vector. The right side of (29) is corresponding to a function \( f \). Using dummy input values \( v(t) \), equality constrains are written by

\[
C(x(t), u(t), v(t)) = \left( u(t) - \frac{u_{\text{max}}}{2} \right)^2 + v^2(t) - \frac{u_{\text{max}}^2}{4} = 0, \tag{30}
\]

Then, the extended performance index \( J \) including the dummy value is given by

\[
J = \int \frac{1}{2} m \dot{y}^2 + \frac{1}{2} k y^2 + u(t)^2 + v^2(t) + \frac{u_{\text{max}}^2}{4} dt.
\]
where $S_f$ and $Q$ are weighting matrices, $r_i$ $(i=1, 2)$ are weighting coefficients. Therefore, comparing with equations (1)--(3) and (29)--(31), main functions are given by

$$J = \frac{1}{2} x^T(t + T) S_f x(t + T) + \int_t^{t+T} \left\{ \frac{1}{2} x^T(\tau) Q x(\tau) + r_1 u^2(\tau) \right\} d\tau,$$

where $S_f$ and $Q$ are weighting matrices, $r_i$ $(i=1, 2)$ are weighting coefficients. Therefore, comparing with equations (1)--(3) and (29)--(31), main functions are given by

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ a x_1 + b x_2 u_1 \end{bmatrix}$$

$$C = \left( u_1 - \frac{u_{\text{max}}}{2} \right)^2 + u_2^2 - \frac{u_{\text{max}}^2}{4}, \quad \text{(Number of Constraints: dimc = 1)}$$

$$\varphi = \frac{1}{2} x^T S_f x,$$

$$L = \frac{1}{2} \left( x^T Q x + r_1 u_1^2 \right) - r_2 u_2^2$$

where $a = \frac{k}{m}$ and $b = -\frac{1}{m}$ are constants, $u = [u(t), v(t)]^T = [u_1, u_2]^T \in \mathbb{R}^2$ is an input vector, and $r_i$ $(i=1, 2)$ are weighting parameters for dummy input values. In this case, the parameters $m$ and $k$ are chosen as $m = 1 \text{ [kg]}$ and $k = 1 \text{ [N/m]}$, namely, $a = -1$ and $b = -1$. These weighting matrices are chosen as $S_f = \text{diag}[1, 10]$, $Q = \text{diag}[1, 10]$ and $r = [r_1, r_2]^T = [1, 0.01]^T$. The terminal penalty $p(t)$ is given zero vector in this case. The horizon $T$ is given by the following function of $t$:

$$T = T_f \left( 1 - e^{-\alpha t} \right)$$

so that the initial condition of $\lambda(t)$ is given by trivial solution of (19) for $T=0$, $T_f = 1.0 \text{ [s]}$ and $\alpha = 0.5$. The parameter $\zeta$ in (19) for stabilization $F=0$ is given by $\zeta = 1000.0$. The time step on the $t$ axis is $0.001 \text{[s]}$, and the horizon on the $\tau$ axis is divided in 50 steps. The initial state is given in the simulation as $x(0) = [2, 0]^T$. The initial Other parameters are given $u_{\text{max}} = 1\text{[N s/m]}$. For details, see the additional NMPC Maple worksheet: "AutoGenU_SAD.mw" for the example.

How to use the Maple worksheet

This system consists of the following files.

- AutoGenU.mw (or AutGenU_CMD.mw)
- rhfunc.c
- rhmainu.c
- NMPCPackage.mla (which is including startup functions for NMPC)
"AutoGenU_CMD.mw" is command version of NMPC Maple worksheet. In this section, it is only introduced how to use the worksheet because it is easy to replace values of parameters in the command version worksheet. You can see some input examples in the "AutoGenU_CMD.mw" included in the products in details.

![Figure 5. Image of AutoGenU for Maple (GUI version)](image-url)
AutoGenU for Maple
(Command version)

Introduction

Initialize

![Initialize Maple for NMPC](image)

Current Directory: C:\Work\NMPC\var1

Example:
- C:\Work\test (for Windows) or /usr/Work/test (for Mac/Linux/Solaris)

Define Setting Parameters

User Settings

<table>
<thead>
<tr>
<th>Define Dimensions of $x, u, C(x,u,p)$ and $p(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimx</td>
</tr>
<tr>
<td>Dimu</td>
</tr>
<tr>
<td>Dimc</td>
</tr>
<tr>
<td>Dimp</td>
</tr>
</tbody>
</table>

Detail input examples for GUI version is as follows. First, set a current directory path in "Initialize" section and put two C files: "rhfuncu.c" and "rhmainu.c" for simulation program in the same directory. The package file: "NMPCPackage.mla" is also set in the same directory. Created C code and result data files are saved in the selected directory.
Next, set the parameters for simulation. In this hovercraft example, dimensions for each vector are as follows. You need only dimension data in each textbox.

Moreover, define the functions. In this case, input the functions as \( f, \ C, \ p, \ L \) and \( \varphi \) as follows. In this case, code editor for temporary parameters is used for the sake of simplicity of description in functions. If you want to define temporary parameters in the functions and the code editor region is not expanded, right-click the region (for Macintosh, Control-click) and select "Expand Code Edit Region", then code editor region is expanded.
Define $f(x,u,p)$, $C(x,u,p)$, $p(t)$, $L(x,u,p)$ and $\phi(x,p)$

**Figure 9.** Define of functions: $f(x,u,p)$, $C(x,u,p)$ and $p(t)$.
Figure 10. Define of functions: $L(x, u, p)$ and $\phi(x,p)$

Example of definition for user's variables and array is as follows. Input each element of "MyVarNames" or "MyArrNames" as string type. Input each element of "MyArrValues" as vector type which is corresponding a dimension of "MyArrDims", for example, "<...>“, respectively. In this case, elements of "MyArrNames": $q$, $r$, $sf$ and $xf$ are elements of weight matrices in the function $L$ and $\phi$ with a corresponding dimension of vectors.
In "Define Simulation Conditions" section, set the another simulation parameters. See Table 1 for details of the following parameters in Figure 11 and 12.

<table>
<thead>
<tr>
<th>MyVarNames</th>
<th>[&quot;H&quot;, &quot;a&quot;, &quot;c&quot;, &quot;unax&quot;, &quot;umin&quot;, &quot;ubar&quot;, &quot;g1&quot;, &quot;g2&quot;]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MyVarValues</td>
<td>[0.974, 0.0125, 0.0405, 0.342, -0.121, 0.1105, 0.28, 0.28]</td>
</tr>
<tr>
<td>MyArrNames</td>
<td>[&quot;a&quot;, &quot;i&quot;, &quot;sf&quot;, &quot;xi&quot;]</td>
</tr>
<tr>
<td>MyArrDims</td>
<td>[dimx, dimu, dimx, dimx]</td>
</tr>
<tr>
<td>MyArrValues</td>
<td>[&lt;10, 15, 0.1, 1, 1, 0.01&gt;, &lt;1, 1, 0.001, 0.001&gt;, &lt;10, 15, 0.1, 1, 1, 0.01&gt;, &lt;0, 0, 0, 0, 0, 0&gt;]</td>
</tr>
</tbody>
</table>

Figure 11. Setting for define setting parameters
The parameters: "outfn" and "fndat" are filename of C source file to be generated and header of data filenames, respectively. Data files are generated by executing the generated simulation program. Their filenames have a common header which is given by the user as value of parameter: "fndat" and end with "c.m", "e.m", "p.m", "x.m" or "u.m", respectively, according to saved data. A file whose name ends with "c.m" saves comments and conditions of simulation. A file with "e.m" saves the components of $F$, which represent errors in optimality, at each time. A file with "p.m" saves the time-variant parameters, $p(t)$. A file with "x.m" saves the state vector and the costate vector. A file with "u.m" saves the control input.
"SimplifyLevel" is a parameter for the simplest form of an expression by applying algebraic transformations. If "SimplifyLevel" > 0, simplify command in maple is used. "Precondition" is a parameter for preconditioning by the Hessian of the Hamiltonian. If "Precondition" > 0, precondition is used. In this case, the values of "SimplifyLevel" and "Precondition" are setting as 0 (See Figure 14).

Figure 15 shows an setting example of optional setting for using mediate variables in functions in code generation. Default settings are used optimization for all functions.
Figure 15. Setting for define setting parameters

Figure 16 shows an example setting flags in C code as optional settings. These flags are used for solver calculation. Default settings are not checked and the corresponding define settings are commented out in C code (See Figure 17). If you change the setting, click the corresponding checkbox.

### Generate C Code

<table>
<thead>
<tr>
<th>NMPC &quot;define&quot; Option</th>
<th>HDIR_EQ_HT</th>
<th>RESET_EQ</th>
<th>ADAMS</th>
<th>Restore option settings</th>
</tr>
</thead>
</table>

If each checkbox is selected, you can use the corresponding "#define" option in C Code.

- **HDIR_EQ_HT**: Flag for \( \Delta t \) where \( \Delta t \) is the time step size and \( \Delta t \) is the sampling period
- **RESET_EQ**: Flag for reset to \( U = 0 \) in GMRES method
- **ADAMS**: Flag for using the Adams Bashforth-Moulton method (If you do not check the checkbox, it is used the Euler method.)

Figure 16. Setting for define setting parameters
It is needed to select C compiler to compile C code on the Maple worksheet. It is needed to set a path for C compiler (cl.exe or gcc.exe) in PATH environment variable on Windows, Linux and Mac in advance. Default compiler is Visual C++ compiler. Selecting "cl.exe", it is needed to set for location of "vcvars32.bat", "vcvars64.bat" or "vcvarsamd64_x86.bat". If you use GNU gcc compiler, it is needed to change the compiler type. Characters in textbox of "Compile Command" is automatically changed. If you select "custom" mode, it can be used custom bat file (or shell script file ".sh file" with execute mode "+x") which is created by user. Figure 18 shows a setting example for "cl.exe". If textbox of "Setting for VC++" is blank in a case of "cl.exe", search of bat file path for Visual C++ is started at the execution time. If your PC operating system is Windsows 7 or later version and a login user does not have a permission as administrator, a process by batch file in "Setting for VC++" will ignored. If a permission error for "Compiler:-Setting" command occurs using VC++ on Windows 7 or later, open Maple as an administrator, type Compiler:-Setup() and set for location of "vcvars32.bat", "vcvars64.bat" or "vcvarsamd64_x86.bat" before using the NMPC worksheet (See Figure 18 and 19).
If you use watcom-1.3 which is installed with Maple, you can use a batch file "watcom_compile.bat" including this archive for the watcom compiler after selecting "custom" mode in Select compiler (See Figure 20 and 21). It is necessary to put a batch file in the same directory as a NMPC worksheet file which you use.
Figure 21. Setting example in a batch file for watcom compiler

Clicking "Execute the entire worksheet" button [!!!] in toolbar in the worksheet after the previous setting, execute of Maple worksheet is started (See Figure 22).

Figure 22. Execute the entire worksheet button in toolbar

If compile of created C code is successful and result data is output, graphs of result data is shown in the "Create and Show Graph" section, automatically. The example of graph for result data is as follows (See Figure 23).
**References**


Appendix A.

Main commands or symbols Maple Quick Reference for this worksheet are given the following table.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Example in this worksheet</th>
</tr>
</thead>
<tbody>
<tr>
<td>;</td>
<td>terminate command (semi column); display result</td>
<td>ev:=&lt;x[1]-p[1], x[2]&gt;;</td>
</tr>
<tr>
<td>[ ]</td>
<td>list delimiter (lists are ordered)</td>
<td>[dimx, dimu, dimx]</td>
</tr>
<tr>
<td>&quot; &quot;</td>
<td>string delimiter (double quote)</td>
<td>[&quot;a0&quot;, &quot;u1max&quot;, &quot;omg&quot;]</td>
</tr>
<tr>
<td>+, -, *, /, ^</td>
<td>add, subtract, multiply, divide, power</td>
<td>1.0 / 120.0</td>
</tr>
<tr>
<td>sin, cos, tan, cot, sec, csc</td>
<td>trigonometric functions</td>
<td>(uv[1] - u[2]) * s / a</td>
</tr>
<tr>
<td>Vector (with &quot;LinearAlgebra&quot; package)</td>
<td>(column) vector</td>
<td>Vector([x[4], x[5], x[6], cos(x[3]) * (u[1] + u[2]) / M, sin(x[3]) * (u[1] + u[2]) / M, (uv[1] - u[2]) * s / a])</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or as</td>
</tr>
<tr>
<td></td>
<td></td>
<td>\langle x[4], x[5], x[6], cos(x[3]) * (u[1] + u[2]) / M, sin(x[3]) * (u[1] + u[2]) / M, (uv[1] - u[2]) * s / a \rangle</td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>.</td>
<td>the period (dot) (for vector/matrix); the inner (or dot) product</td>
<td>((ev^%T.Sf.ev) / 2)</td>
</tr>
<tr>
<td>%T</td>
<td>transpose command</td>
<td>((ev^%T.Sf.ev) / 2)</td>
</tr>
<tr>
<td>seq</td>
<td>create a sequence</td>
<td>(ev := \text{Vector}([seq(xv[i] - xf[i], i = 1..\text{dimx} )]))</td>
</tr>
<tr>
<td>e</td>
<td>scientific notation</td>
<td>(1.0e-6)</td>
</tr>
</tbody>
</table>

[Note] The following is the same descriptions. \(1.0*10^{-6}\)