MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Preconditioned continuation model predictive control

Knyazev, A.; Fujii, Y.; Malyshev, A.

TR2015-084 July 2015

Abstract

Model predictive control (MPC) anticipates future events to take appropriate control actions. Nonlinear MPC (NMPC) describes systems with nonlinear models and/or constraints. A Continuation/GMRES Method for NMPC, suggested by T. Ohtsuka in 2004, uses the GMRES iterative algorithm to solve a forward difference approximation Ax = b of the Continuation NMPC (CNMPC) equations on every time step. The coefficient matrix A of the linear system is often illconditioned, resulting in poor GMRES convergence, slowing down the on-line computation of the control by CNMPC, and reducing control quality. We adopt CNMPC for challenging minimum-time problems, and improve performance by introducing efficient preconditioning, utilizing parallel computing, and substituting MINRES for GMRES.

2015 SIAM Conference on Control and Its Applications

Preconditioned Continuation Model Predictive Control

Andrew Knyazev* Yuta Fujii† Alexander Malyshev‡

Abstract

Model predictive control (MPC) anticipates future events to take appropriate control actions. Nonlinear MPC (NMPC) describes systems with nonlinear models and/or constraints. A Continuation/GMRES Method for NMPC, suggested by T. Ohtsuka in 2004, uses the GMRES iterative algorithm to solve a forward difference approximation $Ax = b$ of the Continuation NMPC (CNMPC) equations on every time step. The coefficient matrix A of the linear system is often ill-conditioned, resulting in poor GMRES convergence, slowing down the on-line computation of the control by CNMPC, and reducing control quality. We adopt CNMPC for challenging minimum-time problems, and improve performance by introducing efficient preconditioning, utilizing parallel computing, and substituting MINRES for GMRES.

1 Introduction

Model predictive control (MPC) is used in many applications to control complex dynamical systems. Examples of such systems include production lines, car engines, robots, other numerically controlled machining, and power generators. The MPC is based on optimization of the operation of the system over a future finite time-horizon, subject to constraints, and implementing the control only over the current time step.

Model predictive controllers rely on dynamic models of the process, most often linear empirical models, in which case the MPC is linear. Nonlinear MPC (NMPC), which describes systems with nonlinear models and constraints, is often more realistic, compared to the linear MPC, but computationally more difficult. Similar to the linear MPC, the NMPC requires solving optimal control problems on a finite prediction horizon, generally not convex, which poses computational challenges.

Numerical solution of the NMPC optimal control problems may be based on Newton-type optimization schemes. Exact Newton-type optimization schemes require an analytic expression of a corresponding Jacobian matrix, which is rarely available in practice and is commonly replaced with a forward difference (FD) approximation; see, e.g., [5]. Such approximate Newton-type optimization schemes utilize the FD approximation of the original nonlinear equation during every time step. An efficient variant of the approximate Newton-type optimization can be performed by a Continuation NMPC (CNMPC) numerical method proposed by T. Ohtsuka in [8], where each step of the algorithm requires solving a system of linear equations performed by the GMRES iterative method [10].

Our contributions presented below are two-fold. We describe an extension of CNMPC with a terminal constraint, suitable to solve minimum-time optimal control problems, and with an optimization parameter. We investigate preconditioning for GMRES in the context of the NMPC problems and using the MINRES iteration [9] instead of GMRES. MINRES provides overall faster implementation, compared to GMRES without restarts, of our approach in cases, where many iterations are required. Our numerical simulations show that the preconditioning can considerably improve the quality of controllers with marginal extra computational time, which can be reduced or eliminated by employing a parallel processing for the preconditioner setup.

The rest of the paper is organized as follows. In Section 2, we formulate CNMPC of Ohtsuka, extended to having a terminal constraint and a parameter. Section 3 describes the original algorithm of Ohtsuka, where the FD linear system is solved using GMRES, and then introduces MINRES as an alternative to GMRES, discusses preconditioning for GMRES and MINRES, and suggests specific algorithms of constructing the preconditioner and using it to accelerate convergence of iterations. In Section 4, we give a detailed description of a test minimum-time optimal control problem, defining a quickest arrival of the system to a given destination, with inequality constraints on the system control, and its CNMPC formulation. Section 5 presents our results of numerical experiments solving the test problem, demonstrating advantages of the proposed approaches.

*Mitsubishi Electric Research Labs (MERL) 201 Broadway, 8th floor, Cambridge, MA 02139, USA, knyazev@merl.com, http://www.merl.com/people/knyazev.

†This work was performed while the author was affiliated with Advanced Technology R&D Center, Mitsubishi Electric Corporation, 8-1-1, Tsukaguchi-honmachi, Amagasaki City, 661-8661, Japan, http://www.mitsubishielectric.com/company/rd/research/labs/advanced_technology/.

‡Mitsubishi Electric Research Labs (MERL) 201 Broadway, 8th floor, Cambridge, MA 02139, USA, malyshev@merl.com, http://www.merl.com/people/malyshev.

2 Finite horizon optimization by CNMPC

As a specific example of a mathematical formalism of NMPC, we consider an extended version of the control problem considered by T. Ohtsuka [8] as follows,

$$\min_{u,p} J,$$

$$J = \phi(t+T, x(t+T), p) + \int_t^{t+T} L(t', x(t'), u(t'), p)dt'$$

subject to

$$\dot{x} = \frac{dx}{dt'} = f(t', x(t'), u(t'), p), \tag{2.1}$$

$$C(t', x(t'), u(t'), p) = 0, \tag{2.2}$$

$$\psi(t+T, x(t+T), p) = 0. \tag{2.3}$$

Here, $x = x(t)$ denotes the vector of the state of the dynamic system, also serving as an initial state for the optimal control problem over the horizon. The vector $u = u(t)$ is the control vector, serving as an input to control the system. The scalar function J describes a performance cost to be minimized, which includes a terminal cost (the first term in the sum) and a cost over the finite horizon (the second term in the sum). Equation (2.1) is the system dynamic model that may be nonlinear in x and/or u. Equation (2.2) describes the equality constraints for the state x and the control u. The horizon time length T may in principle also depend on t, e.g., for time-optimal control problems. In this case, the original problem can be converted into a fixed horizon problem by letting $T(t) = 1 \cdot t_f$, where t_f is an additional parameter to be included in p and determined in MPC. Substituting $t + \tau t_f$ for the time t', we arrive at a problem with the normalized time scale τ and fixed horizon $[t, t+1]$. Such a conversion is applied to the test problem in Section 4.

Compared to [8], one extra constraint (2.3), described by the terminal constraint function ψ, and an extra parameter vector p are being added to the problem formulation, allowing one to extend CNMPC to a wide range of optimal control and design problems.

The NMPC optimal control problem is solved by a variational approach. Its discrete counterpart is solved by the traditional Lagrange method of undetermined multipliers. We denote the costate vector by λ and the Lagrange multiplier vector associated with the equality constraint (2.2) by μ. The terminal constraint (2.3) is relaxed by introducing the Lagrange multiplier ν. The so-called Hamiltonian function, as defined in control theory, is

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

To discretize the continuous formulation of the optimal control problem stated above, we introduce a uniform horizon time grid by dividing the horizon $[t, t+T]$ into N time steps of size $\Delta\tau$ and replace the time-continuous vector functions $x(\tau)$ and $u(\tau)$ by their indexed values x_i and u_i at the grid points. Thus, N is a number of artificial time steps for the optimal control problem over the horizon. The integral in the performance cost J over the time horizon is approximated by a simple quadrature rule. The time derivative of the state vector is approximated by the forward difference formula. Then the discretized optimal control problem appears as follows,

$$\min_{u_i,p} J,$$

$$J = \phi(\tau_N, x_N, p) + \sum_{i=0}^{N-1} L(\tau_i, x_i, u_i, p)\Delta\tau,$$

subject to

$$x_{i+1} = x_i + f(\tau_i, x_i, u_i, p)\Delta\tau, \quad i = 0, 1, \ldots, N-1,$$

$$C(\tau_i, x_i, u_i, p) = 0, \quad i = 0, 1, \ldots, N-1,$$

$$\psi(\tau_N, x_N, p) = 0.$$

We note that we have so far discretized the NMPC optimal control problem only in the horizon time. We will discretize the system time t later using the uniform time step size Δt, i.e. discretization in the horizon time may be different from the time discretization of the system.

The necessary optimality conditions for the discretized horizon problem are obtained using the discrete Lagrangian function

$$\begin{aligned}\mathscr{L}(X, U) = {} & \phi(\tau_N, x_N, p) + \sum_{i=0}^{N-1} L(\tau_i, x_i, u_i, p)\Delta\tau \\ & + \lambda_0^T[x(t) - x_0] \\ & + \sum_{i=0}^{N-1} \lambda_{i+1}^T[x_i - x_{i+1} + f(\tau_i, x_i, u_i, p)\Delta\tau] \\ & + \sum_{i=0}^{N-1} \mu_i^T C(\tau_i, x_i, u_i, p)\Delta\tau + \nu^T\psi(\tau_N, x_N, p),\end{aligned}$$

where $X = [x_i \ \lambda_i]^T$ and $U = [u_i \ \mu_i \ \nu \ p]^T$. Namely, the necessary optimality conditions coincide with the stationarity conditions

$$\frac{\partial \mathscr{L}^T}{\partial X}(X, U) = 0 \text{ and } \frac{\partial \mathscr{L}^T}{\partial U}(X, U) = 0.$$

For example, the derivative with respect to u_i, which is $\partial \mathscr{L}^T/\partial u_i = 0$, yields the following equation:

$$\frac{\partial L}{\partial u_i}(\tau_i, x_i, u_i, p)\Delta\tau + \lambda_{i+1}^T \frac{\partial f}{\partial u_i}(\tau_i, x_i, u_i, p)\Delta\tau + \mu_i^T \frac{\partial C}{\partial u_i}(\tau_i, x_i, u_i, p)\Delta\tau = 0.$$

Using the Hamiltonian function, it can be shortened to

$$\frac{\partial H}{\partial u_i}(\tau_i, x_i, \lambda_{i+1}, u_i, \mu_i, p)\Delta\tau = 0.$$

Taking the derivative with respect to μ_i, which is $\partial \mathscr{L}^T/\partial \mu_i = 0$, we obtain the following equation, which also involves the factor $\Delta\tau$,

$$C(\tau_i, x_i, u_i, p)\Delta\tau = 0.$$

Now we proceed to the construction of a vector function $F(U, x, t)$, which is used to formulate the full set of necessary optimality conditions. The vector function $U = U(t)$ combines the control input u, the Lagrange multiplier μ, the Lagrange multiplier ν, and the parameter p, all in one vector, as follows,

$$U(t) = [u_0^T, \dots, u_{N-1}^T, \mu_0^T, \dots, \mu_{N-1}^T, \nu^T, p^T]^T.$$

The vector argument x in the function $F(U, x, t)$ denotes the current measured state vector, which serves as the initial vector x_0 in the following algorithm, defining an evaluation of $F(U, x, t)$.

1. Starting with the current measured state x_0, compute x_i, $i = 1, 2 \dots, N$, by the forward recursion

   $$x_{i+1} = x_i + f(\tau_i, x_i, u_i, p)\Delta\tau,\ i = 0, \dots, N-1.$$

   Then starting with the value

   $$\lambda_N = \frac{\partial \phi^T}{\partial x}(\tau_N, x_N, p) + \frac{\partial \psi^T}{\partial x}(\tau_N, x_N, p)\nu$$

   compute the costate λ_i, $i = N-1, \dots, 0$, by the backward recursion

   $$\lambda_i = \lambda_{i+1} + \frac{\partial H^T}{\partial x}(\tau_i, x_i, \lambda_{i+1}, u_i, \mu_i, p)\Delta\tau.$$

2. Calculate the vector function $F[U, x, t]$, using the just obtained x_i and λ_i, $i = 0, 1 \dots, N$, as follows,

$$F[U, x, t] = \begin{bmatrix} \frac{\partial H^T}{\partial u}(\tau_0, x_0, \lambda_1, u_0, \mu_0, p)\Delta\tau \\ \vdots \\ \frac{\partial H^T}{\partial u}(\tau_i, x_i, \lambda_{i+1}, u_i, \mu_i, p)\Delta\tau \\ \vdots \\ \frac{\partial H^T}{\partial u}(\tau_{N-1}, x_{N-1}, \lambda_N, u_{N-1}, \mu_{N-1}, p)\Delta\tau \\ C(\tau_0, x_0, u_0, p)\Delta\tau \\ \vdots \\ C(\tau_i, x_i, u_i, p)\Delta\tau \\ \vdots \\ C(\tau_{N-1}, x_{N-1}, u_{N-1}, p)\Delta\tau \\ \psi(\tau_N, x_N, p) \\ \frac{\partial \phi^T}{\partial p}(\tau_N, x_N, p) + \frac{\partial \psi^T}{\partial p}(\tau_N, x_N, p)\nu \\ + \sum_{i=0}^{N-1} \frac{\partial H^T}{\partial p}(\tau_i, x_i, \lambda_{i+1}, u_i, \mu_i, p)\Delta\tau \end{bmatrix}.$$

The optimality condition is the nonlinear equation

$$F[U(t), x(t), t] = 0 \tag{2.4}$$

with respect to the unknown $U(t)$, which needs to be solved numerically by a computer processor at each time step of NMPC in real time on the controller board. This is the most difficult and challenging part of implementation of NMPC. At the initial time $t = t_0$, we need to approximately solve (2.4) directly.

Let us denote the step size of the system time discretization by Δt, assume that $U(t - \Delta t)$ is already available at the time t, and set $\Delta U = U(t) - U(t - \Delta t)$. For a small scalar $h > 0$, which may be different from the system time step Δt and from the horizon time step $\Delta\tau$, we introduce the operator

$$a(V) = (F[U(t - \Delta t) + hV, x(t), t] - F[U(t - \Delta t), x(t), t])/h. \tag{2.5}$$

Then equation (2.4) is equivalent to the equation

$$ha(\Delta U/h) = b, \text{ where } b = -F[U(t - \Delta t), x(t), t].$$

Let us denote the j-th column of the $m \times m$ identity matrix by e_j, where m is the dimension of the vector U, and construct an $m \times m$ matrix A with the columns Ae_j, $j = 1, \dots, m$, defined by the formula

$$Ae_j = a(e_j). \tag{2.6}$$

The matrix A approximates the symmetric Jacobian matrix $F_U[U(t - \Delta t), x(t), t]$ so that $a(V) = AV + O(h)$.

It is important to realize that the operator $a(\cdot)$ in (2.6) may be nonlinear. In particular, this explains why our algorithms of explicitly computing A for the purpose of a preconditioner setup may result in a non-symmetric matrix A. Numerical stability of computations may be improved by enforcing the symmetry, by substituting $(A + A^T)/2$ for A. The deviation from the symmetry gets smaller with a sampling period h, which we are free to choose independently of Δt and $\Delta\tau$.

A key limitation in the choice of h comes from the fact that the cancellation error starts picking up in the finite difference evaluation in the operator $a(V)$ due to inexact arithmetic of the controller processor. This is an unavoidable side effect of using the finite difference approximation of the derivative. A recommended lower bound for the value of h can for example be 10^{-8} in the double precision arithmetic, but the optimal value also depends on the function $F[U, x, t]$.

Given the formulas for computing the vector function $F[U, x, t]$, nonlinear equation (2.4) must be solved at the points of the grid $t_i = t_0 + i\Delta t$, $i = 0, 1, \ldots$.

At the initial state $x_0 = x(t_0)$, we find an approximate solution U_0 to the equation $F[U_0, x_0, t_0] = 0$ by a suitable optimization procedure. The dimension of the vector $u(t)$ is denoted by n_u. Since

$$U(t) = [u_0^T, \ldots, u_{N-1}^T, \mu_0^T, \ldots, \mu_{N-1}^T, \nu^T, p^T]^T,$$

the first block entry of U_0, formed from the first n_u elements of U_0, is taken as the control u_0 at the state x_0. The next state $x_1 = x(t_1)$ is either measured by a sensor or computed by the formula $x_1 = x_0 + \Delta t f(t_0, x_0, u_0)$; cf. (2.1). Now we start the recursion as follows.

At the time t_i, where $i > 0$, we arrive with the state x_i and the vector U_{i-1}. The operator

$$a_i(V) = \left(F[U_{i-1} + hV, x_i, t_i] - F[U_{i-1}, x_i, t_i]\right)/h,$$

defined by (2.5), determines an $m \times m$ matrix A_i with the columns

$$A_i e_j = a_i(e_j),\ j = 1, \ldots, m,$$

as in (2.6). At the current time t_i, our goal is to solve the following equation

$$ha_i(\Delta U_i/h) = b_i, \text{where } b_i = -F[U_{i-1}, x_i, t_i]. \tag{2.7}$$

Then we set $U_i = U_{i-1} + \Delta U_i$ and choose the first n_u components of U_i as the control u_i. The next state $x_{i+1} = x(t_{i+1})$ either comes from a sensor, estimated, or computed by the formula $x_{i+1} = x_i + \Delta t f(t_i, x_i, u_i)$.

Having the basic setup of CNMPC now described, leading to equation (2.7), next we discuss numerical solution of (2.7). Let us highlight that equation (2.7) is never solved exactly in practice, thus, a choice of an algorithm may greatly affect not only the performance of the controller, but also the computed control as well.

3 Algorithms

A direct way to solve (2.7) approximately is generating the matrix A_i and then solving the system of linear equations $A_i\Delta U_i = b_i$ by, e.g., the Gaussian elimination.

Another way is solving (2.7) by a suitable Krylov subspace iteration, e.g., by GMRES [10] or MINRES [9] methods, where we do not need to generate the matrix A_i explicitly. Namely, we simply use the operator $a_i(V)$ instead of computing the matrix-vector product A_iV, for arbitrary vectors V; cf., [5, 6]. In his seminal paper [8], T. Ohtsuka uses the GMRES iteration.

A typical implementation of the preconditioned GMRES without restarts is given by Algorithm 1, where Tr denotes an action of a precontioner T on a vector r, as explained below. The unpreconditioned GMRES, as in [8], simply uses $z = r$. We denote by $H_{i_1:i_2,j_1:j_2}$ the submatrix of H with the entries H_{ij} such that $i_1 \le i \le i_2$ and $j_1 \le j \le j_2$.

Algorithm 1 Preconditioned GMRES without restarts

Input: $a(v)$, b, x_0, $k_{\max}$, T
Output: Solution x of $a(x) = b$

$r = b - a(x_0)$, $z = Tr$, $\beta = \|z\|_2$, $v_1 = z/\beta$
for $k = 1, \ldots, k_{\max}$ **do**
  $r = a(v_k)$, $z = Tr$
  $H_{1:k,k} = [v_1, \ldots, v_k]^T z$
  $z = z - [v_1, \ldots, v_k]H_{1:k,k}$
  $H_{k+1,k} = \|z\|_2$
  $v_{k+1} = z/\|z\|_2$
end for
$y = \arg\min_y \|H_{1:k_{\max}+1,1:k_{\max}} y - [\beta, 0, \ldots, 0]^T\|_2$
$x = x_0 + [v_1, \ldots, v_{k_{\max}}]y$

We emphasize that the operator $a_i(\cdot)$ may be nonlinear, but approximates the symmetric Jacobian matrix $F_U[U_{i-1}, x_i, t_i]$. This implies a slight deviation from the symmetry property $V_2^T a_i(V_1) = (a_i(V_2))^T V_1$ for arbitrary vectors V_1 and V_2. We assume that the deviation is small and propose applying the MINRES iteration to solve equation (2.7).

When the operator $a_i(\cdot)$ is linear and symmetric, the projected $(k_{\max} + 1) \times k_{\max}$ matrix H, constructed by GMRES without preconditioning, is tridiagonal. The MINRES method is then a special variant of GMRES, which makes use of the tridiagonal structure. The table below, adopted from [3], gives a comparison of computational complexities of MINRES and GMRES without preconditioning for solution of a linear system $Ax = b$ with a symmetric $m \times m$ matrix A in terms of memory storage required by working vectors in the solvers and the number of floating-point operations. By t_P we denote the work needed for evaluating $a_i(V)$.

Solver	Storage	Work per iteration
MINRES	$7m$	$t_P + 9m$
GMRES	$(k_{\max}+2)m$	$t_P + (k_{\max}+3)m + \frac{m}{k_{\max}}$

If the matrix A_i gets ill-conditioned, the convergence of GMRES or MINRES may stagnate. The convergence can be improved by preconditioning. A matrix T_i that approximates the matrix A_i^{-1} and such that computing the product $T_i r$ for an arbitrary vector r is relatively easy, is referred to as a preconditioner. The preconditioning for the system of linear equations $Ax = b$ with the preconditioner T formally replaces the original system $Ax = b$ with the equivalent preconditioned linear system $TAx = Tb$. If the condition number $\kappa(TA) = \|TA\|\|A^{-1}T^{-1}\|$ of the matrix TA is small, convergence of iterative solvers for the preconditioned system can be fast. However, the convergence of the preconditioned GMRES, in contrast to that of the preconditioned MINRES with a symmetric positive definite preconditioner, is not necessarily determined by the condition number $\kappa(TA)$. Results on convergence of GMRES in a nonlinear case can be found in [1].

When the approximate solution $x_{k_{\max}}$ computed by GMRES after $k_{\max}$ iterations is not accurate enough, it is very common to restart GMRES with x_0 equal to $x_{k_{\max}}$ instead of increasing the maximum number of iterations $k_{\max}$. Practical implementations of GMRES perform restarts. Restarts allow to cap the GMRES memory use to $k_{\max}+2$ vectors, but may significantly slow down the convergence. In our tests, we apply GMRES without restarts for simplicity of presentation.

To setup the preconditioner, the matrix A_i is computed at some time t_i and then its LU factorization $A_i = LU$ is computed, where L is a lower- and U is an upper-triangular matrix. The product Tr is mathematically given by $Tr = U^{-1}(L^{-1}r)$, but is computed by back-substitution, which is much cheaper than the computation of the inverses of L and U. The same preconditioner T is used in a number of subsequent grid points starting from t_i. The computation of the matrix A_i requires m evaluations $a_i(e_j)$, see (2.6), that can be efficiently implemented in parallel.

The symmetry of the preconditioner T can be used to reduce the memory storage and processor work; see, e.g., [2]. For example, the factorization $T = LDL^T$, see e.g. [4], instead of the LU factorization allows us using only half of memory. The anti-triangular factorization from [7] may also reduce both the memory requirements and work in preconditioning.

MINRES requires symmetric positive definite preconditioners such as in [12]. In our MINRES simulations, although not reported in Section 5 in details, we use the preconditioned MINRES-QLP method from [3].

4 Test problem

In this section, we formulate a test nonlinear problem called TfC below for brevity, which describes the minimum-time motion from a state (x_0, y_0) to a state (x_f, y_f) with an inequality constrained control.

The problem TfC has the following components:

- State vector: $\vec{x} = \begin{bmatrix} x \\ y \end{bmatrix}$. Input: $\vec{u} = \begin{bmatrix} u \\ u_d \end{bmatrix}$.
- Parameter variables: $\vec{p} = [t_f]$, where t_f denotes the length of the evaluation horizon.
- Dynamics: $\dot{\vec{x}} = f(\vec{x}, \vec{u}, \vec{p}) = \begin{bmatrix} (Ax+B)\cos u \\ (Ax+B)\sin u \end{bmatrix}$.
- Constraints: $C(\vec{x}, \vec{u}, \vec{p}) = [(u - c_u)^2 + u_d^2 - r_u^2] = 0$, i.e., the control u always stays within the band $c_u - r_u \le u \le c_u + r_u$).
- Terminal constraints: $\psi(\vec{x}, \vec{p}) = \begin{bmatrix} x - x_f \\ y - y_f \end{bmatrix} = 0$ (the state should pass through the point (x_f, y_f) at $t = t_f$)
- Objective function to minimize:

$$J = \phi(\vec{x}, \vec{p}) + \int_t^{t+t_f} L(\vec{x}, \vec{u}, \vec{p}) dt',$$

where

$$\phi(\vec{x}, \vec{p}) = t_f, \quad L(\vec{x}, \vec{u}, \vec{p}) = -w_d u_d$$

(the state should arrive at (x_f, y_f) in the shortest time; the function L serves to stabilize the slack variable u_d)

- Constants: $A = B = 1$, $x_0 = y_0 = 0$, $t_0 = 0$, $x_f = y_f = 1$, $c_u = 0.8$, $r_u = 0.2$, $w_d = 0.005$.

The components of the corresponding discretized problem on the horizon are given below:

- the scaled horizon time $(\tau - \tau_0)/t_f \in [0, 1]$ substitutes the original horizon time $\tau \in [\tau_0, \tau_0 + t_f]$;
- the discretized scaled horizon time is thus $\tau_i = i\Delta\tau$, where $i = 0, 1, \ldots, N$, and $\Delta\tau = 1/N$;
- the participating variables are the state $\begin{bmatrix} x_i \\ y_i \end{bmatrix}$, the costate $\begin{bmatrix} \lambda_{1,i} \\ \lambda_{2,i} \end{bmatrix}$, the control $\begin{bmatrix} u_i \\ u_{di} \end{bmatrix}$, the Lagrange multipliers μ_i and $\begin{bmatrix} \nu_1 \\ \nu_2 \end{bmatrix}$;

- the state is governed by the model equation

$$\begin{cases} x_{i+1} = x_i + \Delta\tau\left[p\left(Ax_i + B\right)\cos u_i\right], \\ y_{i+1} = y_i + \Delta\tau\left[p\left(Ax_i + B\right)\sin u_i\right], \end{cases}$$

where $i = 0, 1, \ldots, N-1$;

- the costate is determined by the backward recursion ($\lambda_{1,N} = \nu_1$, $\lambda_{2,N} = \nu_2$)

$$\begin{cases} \lambda_{1,i} = \lambda_{1,i+1} \\ \qquad + \Delta\tau\left[pA(\cos u_i\lambda_{1,i+1} + \sin u_i\lambda_{2,i+1})\right], \\ \lambda_{2,i} = \lambda_{2,i+1}, \end{cases}$$

where $i = N-1, N-2, \ldots, 0$;

- the equation $F(U, x_0, t_0) = 0$, where

$$U = [u_0, u_{d,0}, \ldots, u_{N-1}, u_{d,N-1}, \\ \mu_0, \ldots, \mu_{N-1}, \nu_1, \nu_2, p],$$

has the following rows from the top to the bottom:

$$\begin{cases} \Delta\tau p\left[\left(Ax_i + B\right)\left(-\sin u_i\lambda_{1,i+1} + \cos u_i\lambda_{2,i+1}\right)\right. \\ \qquad \left. + 2\left(u_i - c_u\right)\mu_i\right] = 0 \\ \Delta\tau p\left[2\mu_i u_{di} - w_d\right] = 0 \end{cases}$$

$$\left\{\ \Delta\tau p\left[(u_i - c_u)^2 + u_{di}^2 - r_u^2\right] = 0\right.$$

$$\begin{cases} x_N - x_r = 0 \\ y_N - y_r = 0 \end{cases}$$

$$\begin{cases} \Delta\tau\{\sum\limits_{i=0}^{N-1}\left(Ax_i + B\right)(\cos u_i\lambda_{1,i+1} + \sin u_i\lambda_{2,i+1}) \\ + \mu_i[(u_i - c_u)^2 + u_{di}^2 - r_u^2] - w_d u_{di}\} + 1 = 0. \end{cases}$$

Substituting $p\mu_i$ for μ_i, prior to differentiating the Lagrangian, leads to alternative simpler and more numerically stable, as observed in our tests, formulas, as follows

$$\begin{cases} \Delta\tau\left[p(Ax_i + B)\left(-\sin u_i\lambda_{1,i+1} + \cos u_i\lambda_{2,i+1}\right)\right. \\ \qquad \left. + 2\left(u_i - c_u\right)\mu_i\right] = 0 \\ \Delta\tau\left[2\mu_i u_{di} - w_d p\right] = 0 \end{cases}$$

$$\left\{\ \Delta\tau\left[(u_i - c_u)^2 + u_{di}^2 - r_u^2\right] = 0\right.$$

$$\begin{cases} x_N - x_r = 0 \\ y_N - y_r = 0 \end{cases}$$

$$\begin{cases} \Delta\tau[\sum\limits_{i=0}^{N-1}\left(Ax_i + B\right)(\cos u_i\lambda_{1,i+1} + \sin u_i\lambda_{2,i+1}) \\ \qquad - w_d u_{di}] + 1 = 0. \end{cases}$$

We use the latter formulas in our numerical experiments described in the next section.

5 Numerical results

In our numerical experiments with the TfC problem the system of linear equations (2.7) is solved by the GMRES method. We have also tested MINRES, obtaining the controls similar to those with GMRES, reported here. The number of evaluations of $a(V)$ in GMRES does not exceed an a priori chosen parameter denoted by $k_{\max}$, the error tolerance is $tol = 10^{-5}$. The sampling time in the evaluation horizon is $\Delta\tau = 0.1$, the sampling time of the simulation is $\Delta t = 0.02$, and $h = 10^{-5}$.

The preconditioners are constructed as follows. At the time instances $t = jt_p$, $j = 0, 1, \ldots$, with an a priori chosen time increment t_p we calculate all entries of the matrix A by (2.6) and its LU factorization $A = LU$ by Gaussian elimination with partial pivoting. The computed factors L and U are then used in the preconditioner as follows $Tr = U^{-1}(L^{-1}r)$ for all sampling points $t = i\Delta t$ in the interval $[jt_p, (j+1)t_p)$.

The whole set of simulations reported here consists of the following four cases:

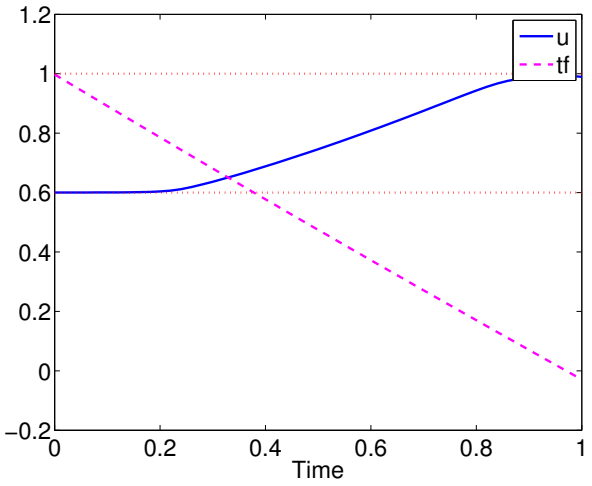
1. no preconditioning, $k_{\max} = 10$;
2. preconditioning with $t_p = 0.2$ sec, $k_{\max} = 1$;
3. preconditioning with $t_p = 0.4$ sec, $k_{\max} = 2$;
4. preconditioning with $t_p = 0.4$ sec, $k_{\max} = 10$.

The computed results are similar in all reported cases. Figure 1 displays the typical CNMPC control u, within the constant constraints, and the time to destination t_f, both as functions of the system time in seconds, shown at the horizontal axis. Figure 2 shows a typical system trajectory in the x-y plane.

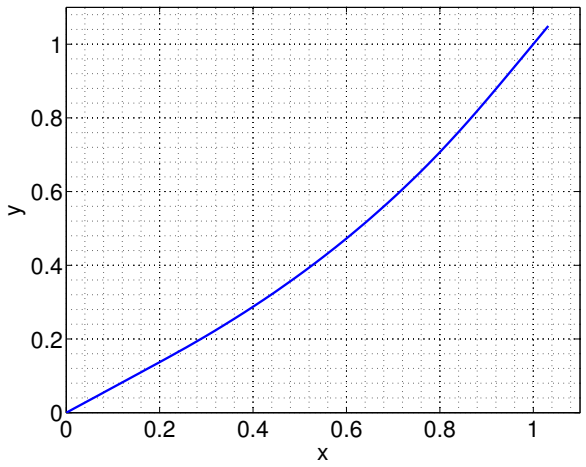
Figures 3–6 show the value of $\|F\|$, which we want to be vanished, and the GMRES residual (the left vertical axis) and the number of the actually performed GMRES iterations (the right vertical axis) at every system time step for all four cases, where the horizontal axis represents the system time in seconds. Figure 3 corresponds to the GMRES iterations without preconditioning. Figures 4-6 involve the preconditioner, recalculated with various frequencies, determined by the time increment t_p, and for different $k_{\max}$ ranging from 1 to 10.

In Figure 3, the number of the actually performed GMRES iterations without preconditioning is always the maximum allowed in this test $k_{\max} = 10$. We use this test as a baseline for comparisons.

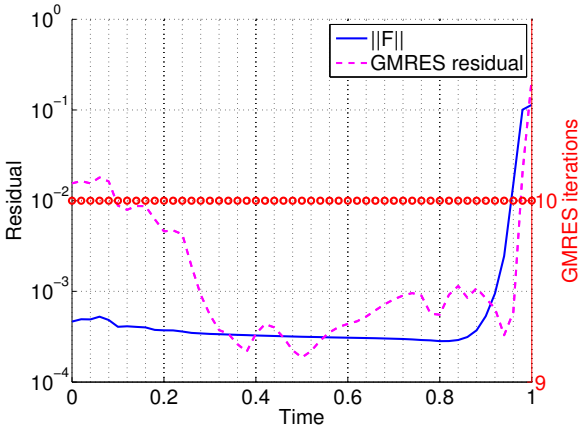
We first point out a good behavior of the preconditioned GMRES even with $k_{\max} = 1$ and where the preconditioner is reconstructed once each $t_p = 0.2$ sec, see Figure 4. This clearly demonstrates the fact that preconditioning reduces the number of evaluations of the vector function $F(U, x, t)$.



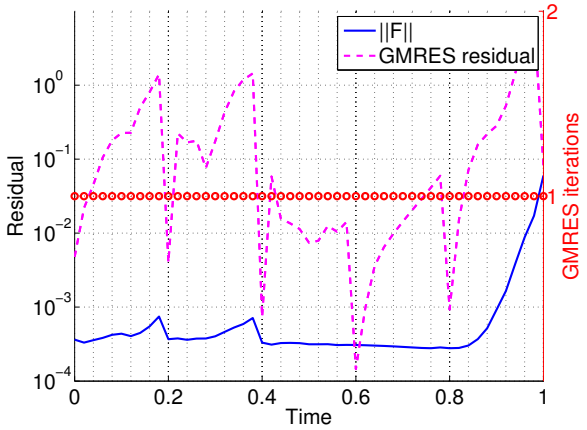
(1) NMPC control u and time to destination t_f for TfC (reaches the target at $t = 0.96$)



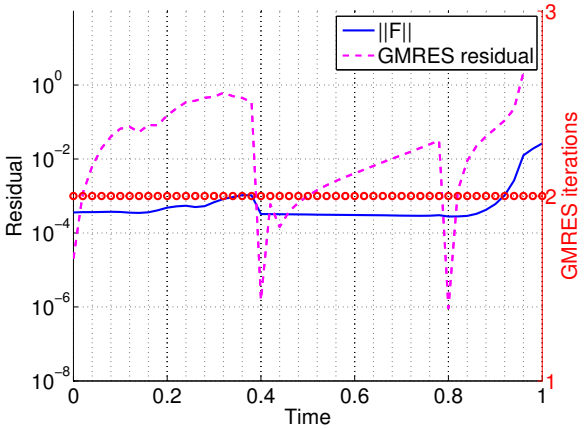
(2) TfC trajectory by NPMC



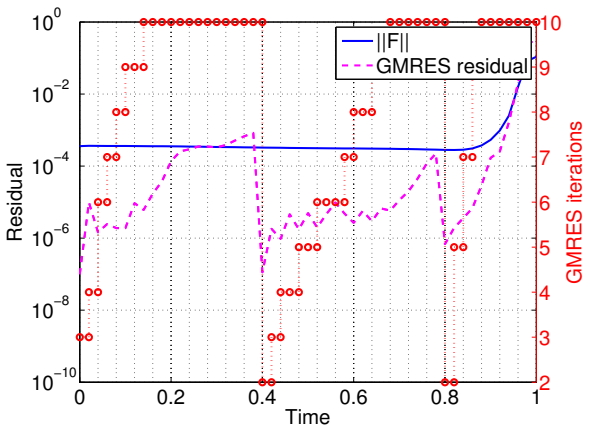
(3) GMRES without preconditioning, $k_{\max} = 10$



(4) GMRES with preconditionining, $t_p = 0.2$ sec, $k_{\max} = 1$



(5) GMRES with preconditionining, $t_p = 0.4$ sec, $k_{\max} = 2$



(6) GMRES with preconditioning, $t_p = 0.4$ sec, $k_{\max} = 10$

The effect of increasing the maximum number $k_{\max}$ of GMRES steps is seen by comparing Figures 4-6. Specifically, in Figure 4, $t_p = 0.2$ sec and $k_{\max} = 1$, compared to $t_p = 0.4$ sec and $k_{\max} = 2$ in Figure 5, i.e., we can recompute the preconditioner twice less frequently at the cost of increasing $k_{\max}$ from 1 to 2, and we observe a slightly better quality of the solution, as measured by the generally smaller values of $\|F\|$ and the GMRES residual (the left vertical axis).

In Figure 6, the preconditioner is recomputed as frequent as in Figure 5, but the largest allowed number of GMRES iterations is increased from $k_{\max} = 2$ to $k_{\max} = 10$. We observe in Figure 6 that GMRES often activates the default tolerance stopping criteria for the residual norm smaller than 10^{-5}, before maxing out the allowed number of iterations $k_{\max}$. Overall, this leads to a generally much smaller residual in Figure 6 compared to that in Figure 5. However, the most decisive quantity $\|F\|$ behaves similar both in Figures 5 and 6, and the computed controls are so similar that the increase of $k_{\max}$ from 2 to 10 may be unnecessary.

Efficiency of preconditioning is illustrated by comparing Figures 3 and 5, where the number of iterations is reduced five times giving similar/smaller values of $\|F\|$.

In minimum-time optimal control problems, the length of the evaluation horizon gets smaller as the state (x, y) approaches the goal position. Near the goal position $(1, 1)$ the control has less capability (controllability) to direct the state towards the goal because of short time for control. This makes the equation $F(U) = 0$ more difficult for numerical solution, thus, $\|F\|$ increases near the goal position, as seen in Figures 3–6.

Conclusions

Time-optimal problems are practically important, giving optimal solutions for guidance, navigation and control, which can be used for vehicles, trains, etc. Due to heavily nonlinear equations and highly coupled variables, the time-optimal problems are difficult to solve numerically. We present an apparently first successful extension of CNMPC for real-time control of such problems. Our numerical experiments demonstrate dramatic acceleration of convergence of iterations without sacrificing control quality, if proper preconditioning is used. The proposed concurrent construction of the preconditioner can be trivially efficiently implemented in parallel on controllers having multiple processing units, such as multi-core, graphics processing units, and modern field-programmable gate arrays. Replacing GMRES with the MINRES iterative solver may help reducing controller memory requirements and increasing the speed of convergence. Our algorithm, including the preconditioner setup implemented in parallel and the iterative solver, can significantly speed up the calculation of the control, compared to traditional sequential CNMPC algorithms, thus allowing to control system with faster dynamics. Our future work concerns analyzing MINRES, as a possible replacement of GMRES, and developing efficient preconditioners, with faster on-line setup and application, within the framework of CNMPC.

References

[1] S. Bellavia and B. Morini, *A globally convergent Newton-GMRES subspace method for systems of nonlinear equations*, SIAM J. Sci. Comput., 23(3) (2001), pp. 940–960, doi:10.1137/S1064827599363976.

[2] M. Benzi, G. H. Golub and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, (2005), pp. 1–137, doi:10.1017/S0962492904000212.

[3] S.-C. T. Choi and M. A. Saunders, *Algorithm 937: MINRES-QLP for symmetric and Hermitian linear equations and least-squares problems*, ACM Trans. Math. Softw., 40(2) (2014), pp. 16:1–16:12, doi:10.1145/2527267.

[4] G. H. Golub and C. F. Van Loan, *Matrix computations, 4th ed.*, The John Hopkins U. Press, Baltimore, MR, 2013. ISBN: 978-1-4214-0794-4.

[5] C. T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM, Philadelphia, 1995, doi:10.1137/1.9781611970944.

[6] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: a survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397, doi:10.1016/j.jcp.2003.08.010.

[7] N. Mastronardi and P. Van Dooren, *The antitriangular factorization of of symmetric matrices*, SIAM J. Matrix Anal. Appl., 34(1) (2013), pp. 173–196, doi:10.1137/110858860.

[8] T. Ohtsuka, *A Continuation/GMRES method for fast computation of nonlinear receding horizon control*, Automatica, 40(4) (2004), pp. 563–574, doi:10.1016/j.automatica.2003.11.005.

[9] C. C. Page and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629, doi:10.1137/0712047.

[10] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm fr solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869, doi:10.1137/0907058.

[11] T. Tanida and T. Ohtsuka, *Preconditioned C/GMRES algorithm for nonlinear receding horizon control of hovercrafts connected by a string*, Proc. IEEE Int. Conf. Control Applic., Taipei, Taiwan, September 2-4, (2004), pp. 1609–1614, doi:10.1109/CCA.2004.1387606.

[12] E. Vecharynski and A. Knyazev, *Absolute value preconditioning for symmetric indefinite linear systems*, SIAM J. Sci. Comput., 35(2) (2013), pp. A696–A718, doi:10.1137/120886686.