2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)
Sheraton Arabella Park Hotel, Munich, Germany, July 3-7, 2017

Successive Linearization Based Model Predictive Control of Variable Stiffness Actuated Robots

Altay Zhakatayev, *Member, IEEE*, Bexultan Rakhim, Olzhas Adiyatov, Almaskhan Baimyshev, Huseyin Atakan Varol, *Senior Member, IEEE*

Abstract— **Variable stiffness actuation is a new design paradigm for high performance and energy efficient robots with inherent safety features. Nonlinear model predictive control (NMPC) was employed to control these robots due to its ability to cope with constrained and nonlinear systems. Even though the results for NMPC are promising, one major weakness is the computational cost of this algorithm. It restricts the use of NMPC to low degree of freedom robots with relatively slow dynamics. This problem can be alleviated by finding an approximate linear representation of the system and using less computation hungry traditional model predictive control (MPC). In this work, we present our successive linearization based MPC (SLMPC) framework for variable stiffness actuated (VSA) robots. The system is linearized and discretized at every sampling instant and a quadratic problem is formulated using this discrete-time linear model. Solution of this quadratic problem provides the control inputs for the control horizon. In order to compare our scheme to NMPC, we conducted experiments with a reaction wheel augmented VSA system. For a 16 s trajectory tracking experiment, the root-mean-square errors were 0.54 and 0.64 degrees for NMPC and SLMPC methods, respectively, whereas the average computation time of the control rule was 2.17 ms for NMPC and 1.25 ms for SLMPC. Halving the computation time without compromising tracking performance shows the potential of our approach as a viable control alternative for VSA robots.**

I. Introduction

Design and control of variable stiffness actuated (VSA) systems has recently attracted a major attention within the robotics community [1]. VSA systems [2]–[7] have the potential to address fundamental issues of robotics such as safety, energy-efficiency and adaptability. Their complex mechanical structure brings forth formidable challenges in design and control. After initial focus on mechanical design, researchers started tackling the control problem of these systems. An optimal control based framework was presented for task planning of VSA robots in [8]. Later, nonlinear model predictive control (NMPC) was used to track trajectories generated by optimal control problem (OCP) based task planners [9], [10]. Originally used for chemical plants with slow dynamics, main limitation of the NMPC approach is the computational cost. This prohibits the use of short sampling times for VSA robots with fast dynamics and multiple links. This problem might be solved by linearizing the robot dynamics at each sampling time and then using the linearized MPC (LMPC) method in closed-loop scheme. One might argue that LMPC will perform worse than NMPC due to a less accurate model employed. However, we hypothesize that shorter sampling time due to lower computational cost could counteract this disadvantage by responding faster to deviations from the desired trajectory.

MPC with linearization was applied to multiple problems in various domains. MPC using successive linearization was employed for the control of a chemical reactor in [11]. In their receding horizon control scheme for nonlinear stochastic systems, researchers used Jacobian linearization to derive a sequence of convex optimization problems [12]. Recently, linearized MPC was used for anti-surge control of a centrifugal compressor using recycle valve opening and driver torque as inputs [13]. The nonlinear model of the compressor was linearized and discretized at every sampling time of the controller. Stochastic MPC with successive linearization and probabilistic safety constraints was used to prevent unintended roadway departures of a vehicle in [14]. A MPC scheme for the control of a wind turbine employing linearized parameter-varying models was presented in [15]. Instead of linearizing the system successively at each sampling time, the authors used a family of pre-obtained linearized models of a wind turbine at different operating points. Model reduction based linear MPC was presented as a controller for nonlinear distributed parameter systems [16].

NMPC and LMPC (obtained from successive linearization of a nonlinear model) were benchmarked for the horizontal trajectory tracking control of autonomous surface vehicles in [17]. NMPC tracked the planned trajectory with less error and required more computation time. MPC, successive linearization based MPC and Wiener NMPC were applied to the control of a three-tank-system [18]. The authors concluded that successive linearization based NMPC is superior to Wiener NMPC, and these two overperform regular MPC. It is also noteworthy that the sampling times for real-world MPC experiments were 1 min in [11], 50 ms in [13] and 20 ms in [9].

In this paper, we apply successive linearization MPC (SLMPC) and NMPC to control a VSA robot with fast dynamics and compare their performances. To the best of our knowledge, SLMPC was not used before for the control of VSA robots. The rest of the paper is organized as follows. In Section II SLMPC method is explained in depth. We delibaretely followed a step-by-step presentation to allow other users of VSA robots to easily implement this scheme for their systems. Section III presents our experimental study. After the description of the reaction wheel augmented VSA system, we explain our experiments for benchmarking. Results of these experiments are provided and discussed in Section IV, while Section V concludes the paper.

II. Successive Linearization Based MPC

Traditional MPC employs discrete-time linear models. However, the majority of dynamical systems are nonlinear and continuous. One way to apply MPC to nonlinear systems is successive linearization. According to this scheme, the continuous nonlinear system is linearized at every sampling time around the current system state. Once the linearized continuous system is obtained, it is discretized for the given sampling time of the controller. The discretized linear model is then used to generate matrices which will be used to compute the prediction horizon values based on the current system state and the control inputs in the horizon. These prediction matrices are then used to form a quadratic programming (QP) problem which aims to find a control sequence that minimizes a given cost function. The block diagram for this successive linearization based MPC scheme is illustrated in Fig. 1. In the remainder of this section, we will provide a detailed description of this scheme.

A. Linearization of a Continuous Nonlinear System

For a system with n_x states, n_u inputs, and n_o outputs, we can define the corresponding state $\mathbf{x} \in \mathbb{R}^{n_x}$, input $\mathbf{u} \in \mathbb{R}^{n_u}$ and output $\mathbf{y} \in \mathbb{R}^{n_o}$ vectors. Each output y_i and dynamics of each state x_i can be written as nonlinear functions $y_i = z_i(\mathbf{x}, \mathbf{u}),\ i = 1,..,n_o$ and $\dot{x_i} = f_i(\mathbf{x}, \mathbf{u}),\ i = 1,..,n_x$, respectively. Using state-space representation, the overall system dynamics can be represented as

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}), \quad \mathbf{y} = Z(\mathbf{x}, \mathbf{u}). \tag{1}$$

The system in (1) might have constraints specified by $\mathbf{x_l} \le \mathbf{x} \le \mathbf{x_u}$ and $\mathbf{u_l} \le \mathbf{u} \le \mathbf{u_u}$, where $\mathbf{x_l}$, $\mathbf{x_u}$, $\mathbf{u_l}$, $\mathbf{u_u}$ are, correspondingly, lower and upper limit vectors for state and control inputs. The system described in (1) can be linearized by finding its gradient matrices at a desired operational point. Jacobian sub-matrix of the function F in derivatives of state vector components can be written as

$$A_c = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_{n_x}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{n_x}}{\partial x_1} & \frac{\partial f_{n_x}}{\partial x_2} & \cdots & \frac{\partial f_{n_x}}{\partial x_{n_x}} \end{bmatrix} \tag{2}$$

or $A_c(i,j) = \frac{\partial f_i}{\partial x_j}$ in short. Using the same notation, we can write $B_c(i,j) = \frac{\partial f_i}{\partial u_j}$, $C_c(i,j) = \frac{\partial z_i}{\partial x_j}$ and $D_c(i,j) = \frac{\partial z_i}{\partial u_j}$. The desired operational point is specified by the state vector $\mathbf{x_{op}}$, its time derivative $\dot{\mathbf{x}}_{\mathbf{op}}$, and the control vector $\mathbf{u_{op}}$. The partial derivatives in the above matrices need to be evaluated at $\mathbf{x} = \mathbf{x_{op}}$, $\dot{\mathbf{x}} = \dot{\mathbf{x}}_{\mathbf{op}}$ and $\mathbf{u} = \mathbf{u_{op}}$. Defining the linearized system state $\mathbf{x_L}$, control $\mathbf{u_L}$ and output $\mathbf{y_L}$ vectors, the system evolution can be described by

$$\dot{\mathbf{x}}_{\mathbf{L}} = \dot{\mathbf{x}}_{\mathbf{op}} + A_c(\mathbf{x_L} - \mathbf{x_{op}}) + B_c(\mathbf{u_L} - \mathbf{u_{op}}). \tag{3}$$

Collecting the constant terms of (3) into one term $\mathcal{K} = \dot{\mathbf{x}}_{\mathbf{op}} - A_c\mathbf{x_{op}} - B_c\mathbf{u_{op}} \in \mathbb{R}^{n_x}$, the state-space representation of linearized system becomes

$$\dot{\mathbf{x}}_{\mathbf{L}} = A_c\mathbf{x_L} + B_c\mathbf{u_L} + \mathcal{K}, \quad \mathbf{y_L} = C_c\mathbf{x_L} + D_c\mathbf{u_L}. \tag{4}$$

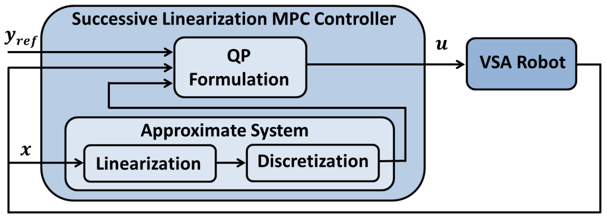

Fig. 1. The schematic diagram of the successive linearization based model-predictive control for VSA systems.

B. Discretization of a Continuous Linearized System

Next, we discretize the continuous linearized system. First, we multiply both sides of (4) by matrix exponential $e^{-A_c t}$

$$e^{-A_c t}\dot{\mathbf{x}}_{\mathbf{L}}(t) = e^{-A_c t}A_c\mathbf{x_L}(t) + e^{-A_c t}(B_c\mathbf{u_L}(t) + \mathcal{K}). \tag{5}$$

Equation (5) can be reorganized as

$$\frac{d}{dt}(e^{-A_c t}\mathbf{x_L}(t)) = e^{-A_c t}(B_c\mathbf{u_L}(t) + \mathcal{K}). \tag{6}$$

Integrating and then multiplying both sides of (6) by $e^{A_c t}$, we obtain the analytical solution of the continuous linear dynamic model as

$$\mathbf{x_L}(t) = e^{A_c t}\mathbf{x_L}(0) + \int_0^t e^{A_c(t-\tau)}(B_c\mathbf{u_L}(\tau) + \mathcal{K})d\tau. \tag{7}$$

Denoting the discrete time indices as $\mathbf{x_L}[k] = \mathbf{x_L}(kT_s)$, where T_s is the sampling time, and replacing t in (7) with kT_s, we can write

$$\mathbf{x_L}[k+1] = e^{A_c T_s}\mathbf{x_L}[k] + \int_{kT_s}^{kT_s+T_s} e^{A_c(kT_s+T_s-\tau)}(B_c\mathbf{u_L}(\tau) + \mathcal{K})d\tau. \tag{8}$$

Assuming the input $\mathbf{u_L} = \mathbf{u_L}[k]$ is constant during each time step and A_c is invertible, (8) can be reduced to

$$\mathbf{x_L}[k+1] = e^{A_c T_s}\mathbf{x_L}[k] + A_c^{-1}(e^{A_c T_s} - I)(B_c\mathbf{u_L}[k] + \mathcal{K}) \tag{9}$$

Equation (9) is the exact solution of the discretized linear system state evolution. The output becomes

$$\mathbf{y_L}[k] = C_c\mathbf{x_L}[k] + D_c\mathbf{u_L}[k]. \tag{10}$$

In order to write (9) and (10) in a compact form, we can define discrete-time state-space matrices as $A_d = e^{A_c T_s}$, $B_d = A_c^{-1}(e^{A_c T_s} - I)B_c$, $\mathbb{K} = A_c^{-1}(e^{A_c T_s} - I)\mathcal{K}$, $C_d = C_c$, and $D_d = D_c$. Since from now on we will operate in discrete-time domain, with a slight abuse of the notation we can denote $\mathbf{x_L}[k]$ as $\mathbf{x_D^k}$ and $\mathbf{u_L}[k]$ as $\mathbf{u_D^k}$. Then the discrete-time linearized state-space equations can be written concisely as

$$\mathbf{x_D^{k+1}} = A_d\mathbf{x_D^k} + B_d\mathbf{u_D^k} + \mathbb{K}, \quad \mathbf{y_D^k} = C_d\mathbf{x_D^k} + D_d\mathbf{u_D^k}. \tag{11}$$

C. Derivation of the Quadratic Programming Problem

MPC formalism requires the computation of the system evolution for a finite future horizon, also termed the *Prediction Horizon*. Predictions for a given set of control inputs are accomplished using the discrete-time linear system model. The objective is to track a reference trajectory by minimizing a cost function and simultaneously satisfying the constraints. For the current time step $\mathbf{k}$, the cost function usually depends on the tracking error $\mathbf{e^k} = \mathbf{y_D^k} - \mathbf{r^k}$ (the deviation of the output $\mathbf{y_D^k}$ from the desired reference trajectory $\mathbf{r^k}$) and control inputs $\mathbf{u_D^k}$. We can describe the evolution of the error over the prediction horizon starting from current time step k as

$$\mathbf{e^k} = C_d\mathbf{x_D^k} + D_d\mathbf{u_D^k} - \mathbf{r^k} \tag{12a}$$

$$\mathbf{e^{k+1}} = C_d\mathbf{x_D^{k+1}} + D_d\mathbf{u_D^{k+1}} - \mathbf{r^{k+1}} = C_dA_d\mathbf{x_D^k} + C_dB_d\mathbf{u_D^k} + C_d\mathbb{K} + D_d\mathbf{u_D^{k+1}} - \mathbf{r^{k+1}} \tag{12b}$$

$$\mathbf{e^{k+2}} = C_d\mathbf{x_D^{k+2}} + D_d\mathbf{u_D^{k+2}} - \mathbf{r^{k+2}} = C_dA_d^2\mathbf{x_D^k} + C_dA_dB_d\mathbf{u_D^k} + C_dB_d\mathbf{u_D^{k+1}} + C_dA_d\mathbb{K} + C_d\mathbb{K} + D_d\mathbf{u_D^{k+2}} - \mathbf{r^{k+2}} \tag{12c}$$

and so on. We can represent the error vectors over the horizon with n_p time steps as a concatenated vector $\overrightarrow{\mathbf{e}} \in \mathbb{R}^{n_p \cdot n_o}$. Similarly, we can express the control signals over the whole horizon as $\overrightarrow{\mathbf{u}} \in \mathbb{R}^{n_p \cdot n_u}$. Using this notation, the system behavior over the horizon can be summarized by

$$\underbrace{\begin{bmatrix} \mathbf{e^k} \\ \mathbf{e^{k+1}} \\ \mathbf{e^{k+2}} \\ \vdots \\ \mathbf{e^{k+n_p-1}} \end{bmatrix}}_{\overrightarrow{\mathbf{e_k}}} = \underbrace{\begin{bmatrix} C_d \\ C_dA_d \\ C_dA_d^2 \\ \vdots \\ C_dA_d^{n_p-1} \end{bmatrix}}_{P\in\mathbb{R}^{n_pn_o\times n_x}} \mathbf{x_D^k} + \underbrace{\begin{bmatrix} D_d & 0 & 0 & \cdots \\ C_dB_d & D_d & 0 & \cdots \\ C_dA_dB_d & C_dB_d & D_d & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ C_dA_d^{n_p-2}B_d & C_dA_d^{n_p-3}B_d & C_dA_d^{n_p-4}B_d & \cdots \end{bmatrix}}_{H\in\mathbb{R}^{n_pn_o\times n_pn_u}} \cdot \underbrace{\begin{bmatrix} \mathbf{u_D^k} \\ \mathbf{u_D^{k+1}} \\ \mathbf{u_D^{k+2}} \\ \vdots \\ \mathbf{u_D^{k+n_p-1}} \end{bmatrix}}_{\overrightarrow{\mathbf{u_k}}} + \underbrace{\begin{bmatrix} 0 \\ C_d \\ C_d(I+A_d) \\ \vdots \\ C_d(I+\sum_{i=1}^{n_p-2}A_d^i) \end{bmatrix}}_{E\in\mathbb{R}^{n_pn_o\times n_x}} \mathbb{K} - \underbrace{\begin{bmatrix} \mathbf{r^k} \\ \mathbf{r^{k+1}} \\ \mathbf{r^{k+2}} \\ \vdots \\ \mathbf{r^{n_p-1}} \end{bmatrix}}_{\overrightarrow{\mathbf{r_k}}} \tag{13}$$

where P, H and E are the *output prediction matrices*, $\mathbb{K}$ is the constant term from (11). $\overrightarrow{\mathbf{e_k}}$, $\overrightarrow{\mathbf{u_k}}$, and $\overrightarrow{\mathbf{r_k}}$ are the combined error, control input and reference vectors for the horizon starting at instance k. The compact form of (13) is

$$\overrightarrow{\mathbf{e_k}} = P\mathbf{x_D^k} + H\overrightarrow{\mathbf{u_k}} + K \tag{14}$$

where $K = E\mathbb{K} - \overrightarrow{\mathbf{r_k}} \in \mathbb{R}^{n_p \cdot n_o}$ combines all constant terms.

The cost function, quantitative indicator of the controller performance, is defined as

$$J = \frac{1}{2}(\overrightarrow{\mathbf{e_k}}^T Q \overrightarrow{\mathbf{e_k}} + \overrightarrow{\mathbf{u_k}}^T R \overrightarrow{\mathbf{u_k}}) \tag{15}$$

where $Q \in \mathbb{R}^{n_pn_o \times n_pn_o}$ and $R \in \mathbb{R}^{n_pn_u \times n_pn_u}$ are positive definite and commonly diagonal matrices of weights for error and input vectors, respectively. By substituting the right-hand-side of (14) into the cost function (15) we get

$$J = \frac{1}{2}((P\mathbf{x_D^k} + H\overrightarrow{\mathbf{u_k}} + K)^T Q (P\mathbf{x_D^k} + H\overrightarrow{\mathbf{u_k}} + K) + \overrightarrow{\mathbf{u_k}}^T R \overrightarrow{\mathbf{u_k}}).$$

By removing the terms not depending on the control input $\overrightarrow{\mathbf{u_k}}$ and grouping the quadratic and linear terms, we obtain

$$J = \frac{1}{2}\overrightarrow{\mathbf{u_k}}^T \underbrace{(H^TQH + R)}_{G} \overrightarrow{\mathbf{u_k}} + \underbrace{(\mathbf{x_D^k}^T P^T + K^T)QH}_{W^T} \overrightarrow{\mathbf{u_k}} \tag{16}$$

Using the matrix G and the vector W, the cost function J becomes

$$J = \frac{1}{2}\overrightarrow{\mathbf{u_k}}^T G \overrightarrow{\mathbf{u_k}} + W^T \overrightarrow{\mathbf{u_k}}. \tag{17}$$

However, since the control inputs are included in the cost function with a non-zero weight, there may occur a steady-state error of the output, such that having this error in the cost function is "cheaper" than reducing it by increasing the control inputs. This may be solved by defining the cost function through the increments of the control inputs rather than the inputs themselves. This way, the influence of the values of the control inputs on the cost function are minimal. The components of the control vector over the horizon can be transformed to $\mathbf{u_D^k} = \mathbf{u_D^{k-1}} + \delta\mathbf{u_1}$, $\mathbf{u_D^{k+1}} = \mathbf{u_D^k} + \delta\mathbf{u_2} = \mathbf{u_D^{k-1}} + \delta\mathbf{u_1} + \delta\mathbf{u_2}$, ..., $\mathbf{u_D^{k+n_p-1}} = \mathbf{u_D^{k+n_p-2}} + \delta\mathbf{u_{n_p}} = \mathbf{u_D^{k-1}} + \delta\mathbf{u_1} + ... + \delta\mathbf{u_{n_p}}$ or, concisely as

$$\underbrace{\begin{bmatrix} \mathbf{u_D^k} \\ \mathbf{u_D^{k+1}} \\ \vdots \\ \mathbf{u_D^{k+n_p-1}} \end{bmatrix}}_{\overrightarrow{\mathbf{u_k}}} = \underbrace{\begin{bmatrix} \mathbf{u_D^{k-1}} \\ \mathbf{u_D^{k-1}} \\ \vdots \\ \mathbf{u_D^{k-1}} \end{bmatrix}}_{\overrightarrow{\mathbf{u}}_{\mathbf{k-1}}} + \underbrace{\begin{bmatrix} I & 0 & \cdots \\ I & I & \cdots \\ \vdots & \vdots & \ddots \\ I & I & \cdots \end{bmatrix}}_{\Delta\in\mathbb{R}^{n_pn_u\times n_pn_u}} \underbrace{\begin{bmatrix} \delta\mathbf{u_1} \\ \delta\mathbf{u_2} \\ \vdots \\ \delta\mathbf{u_{n_p}} \end{bmatrix}}_{\overrightarrow{\delta\mathbf{u_k}}}$$

where Δ is a lower diagonal matrix, each element of which is an identity matrix $I \in \mathbb{R}^{n_u \times n_u}$ or a zero matrix 0 of the same size. Then (14) can be rewritten as

$$\overrightarrow{\mathbf{e_k}} = P\mathbf{x_D^k} + H(\overrightarrow{\mathbf{u}}_{\mathbf{k-1}} + \Delta\overrightarrow{\delta\mathbf{u_k}}) + K. \tag{18}$$

Since the $\overrightarrow{\mathbf{u}}_{\mathbf{k-1}}$ is constant during the whole horizon, we can include it into the constant vector K. The cost function J_Δ in $\delta\mathbf{u}$ notation can be described by

$$J_\Delta = \frac{1}{2}\overrightarrow{\delta\mathbf{u_k}}^T G_\Delta \overrightarrow{\delta\mathbf{u_k}} + W_\Delta^T \overrightarrow{\delta\mathbf{u_k}} \tag{19}$$

where $G_\Delta = \Delta^T G \Delta$ and $W_\Delta^T = \overrightarrow{\mathbf{u}}_{\mathbf{k-1}}^T G\Delta + W^T\Delta$.

An important feature of MPC is that it can work with an imposed set of constraints (limits on any linear function of the states) and bounds (limits on control inputs). In order to impose constraint on states, evolution of the state vector for the horizon should be determined

$$\underbrace{\begin{bmatrix} \mathbf{x_D^k} \\ \mathbf{x_D^{k+1}} \\ \mathbf{x_D^{k+2}} \\ \vdots \\ \mathbf{x_D^{k+n_p}} \end{bmatrix}}_{\overrightarrow{\mathbf{x_k}} \in \mathbb{R}^{(n_p+1)n_x}} = \underbrace{\begin{bmatrix} I \\ A_d \\ A_d^2 \\ \vdots \\ A_d^{n_p} \end{bmatrix}}_{P_x \in \mathbb{R}^{(n_p+1)n_x \times n_x}} \mathbf{x_D^k} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots \\ B_d & 0 & 0 & \cdots \\ A_d B_d & B_d & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ A_d^{n_p-1} B_d & A_d^{n_p-2} B_d & A_d^{n_p-3} B_d & \cdots \end{bmatrix}}_{H_x \in \mathbb{R}^{(n_p+1)n_x \times n_p n_u}} \cdot \begin{bmatrix} \mathbf{u_D^k} \\ \mathbf{u_D^{k+1}} \\ \mathbf{u_D^{k+2}} \\ \vdots \\ \mathbf{u_D^{k+n_p-1}} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ I \\ I + A_d \\ \vdots \\ I + \sum_{i=1}^{n_p-1} A_d^i \end{bmatrix}}_{E_x \in \mathbb{R}^{(n_p+1)n_x \times n_x}} \mathbb{K} \tag{20}$$

where $\overrightarrow{\mathbf{x_k}}$ is the vector of n_p+1 state vectors for the horizon time duration, P_x, H_x and E_x are *state prediction matrices*. In short (20) can be written as

$$\overrightarrow{\mathbf{x_k}} = P_x \mathbf{x_D^k} + H_x \overrightarrow{\mathbf{u_k}} + E_x \mathbb{K} \tag{21}$$

Constraint on the state, defined as $\mathbf{x_l} \leq V\overrightarrow{\mathbf{x_k}} \leq \mathbf{x_u}$, can be written in terms of input vector $\mathbf{x_l} - VP_x\mathbf{x_D^k} - VE_x\mathbb{K} \leq VH_x\overrightarrow{\mathbf{u_k}} \leq \mathbf{x_u} - VP_x\mathbf{x_D^k} - VE_x\mathbb{K}$. Here the matrix $V \in \mathbb{R}^{(n_p+1)n_c \times (n_p+1)n_x}$ selects n_c state vector components, for which there are active constraints. If constraints are defined for all states individually, then V is simply the identity matrix. Finally, by using the $\overrightarrow{\delta\mathbf{u_k}}$ notation and denoting $U = VH_x\Delta$, the QP problem can be defined as

$$\arg\min_{\overrightarrow{\delta\mathbf{u_k}}} \; J = \frac{1}{2}\overrightarrow{\delta\mathbf{u_k}}^T G_\Delta \overrightarrow{\delta\mathbf{u_k}} + W_\Delta^T \overrightarrow{\delta\mathbf{u_k}} \tag{22a}$$

$$\text{subject to } \; \mathbf{U_l} \leq U\overrightarrow{\delta\mathbf{u_k}} \leq \mathbf{U_u} \tag{22b}$$

$$u_\mathbf{l} \leq \overrightarrow{\delta\mathbf{u_k}} \leq u_\mathbf{u} \tag{22c}$$

where $\mathbf{U_l} = \mathbf{x_l} - \mathbf{U^k}$ and $\mathbf{U_u} = \mathbf{x_u} - \mathbf{U^k}$, $\mathbf{U^k} = VP_x\mathbf{x_D^k} + VE_x\mathbb{K} + VH_x\overrightarrow{\mathbf{u}}_{\mathbf{k-1}}$, while $u_\mathbf{l} = \Delta^{-1}(\mathbf{u_l} - \overrightarrow{\mathbf{u}}_{\mathbf{k-1}})$ and $u_\mathbf{u} = \Delta^{-1}(\mathbf{u_u} - \overrightarrow{\mathbf{u}}_{\mathbf{k-1}})$. We note that constraints and bounds in (22) might be fixed or varied at each sampling time.

A regular QP solver uses matrices G and W along with information about constraints and bounds, to generate a minimizer. In the closed-loop MPC scheme, the first control vector in the horizon is applied to the system and the rest are dismissed. After the current time step passes, new measurements of the system are obtained. The described process of linearization, discretization, generation of prediction matrices, and applying only the first control is repeated each time step in a closed-loop fashion.

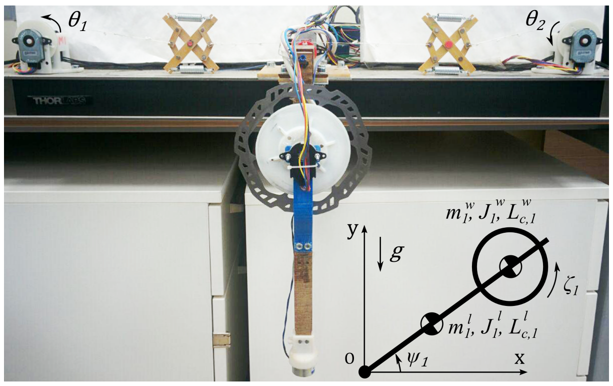

Fig. 2. VSA robot with a reaction wheel and its kinematic model.

III. Experiments

A. Experimental Setup

In order to assess the performance of the SLMPC, we conducted experiments with a single-link reaction wheel integrated VSA robot. An earlier version of this setup was controlled in [19] using NMPC. The picture of the experimental setup and its kinematic model are shown in Fig. 2. The link joint is connected to two antagonistically placed brushless motors (BG40x25, Dunkermotoren) with planetary gearboxes (PLG32 (20.25:1), Dunkermotoren) via nonlinear elastic elements (NEEs). The system is described by the model

$$\mathcal{M}\ddot{\psi} + \mathcal{B}\dot{\psi} + \mathcal{N} = \mathcal{T} \tag{23}$$

where $\psi = [\psi_1, \zeta_1]'$ is the vector of generalized coordinates. ψ_1 and ζ_1 are the link and the reaction wheel angles measured with incremental encoders. $\mathcal{M} = [\mathcal{M}_{11} \; J_1^w; \; J_1^w \; J_1^w]$ is the inertia matrix with $\mathcal{M}_{11} = m_1^l(L_{c,1}^l)^2 + m_1^w(L_{c,1}^w)^2 + J_1^l + J_1^w$. Here $m_1^l, J_1^l, L_{c,1}^l$ are the mass, the inertia about the center of mass, and the distance from the pivot to the center of mass of the link and $m_1^w, J_1^w, L_{c,1}^w$ are the same parameters but of the reaction wheel. The matrix of viscous frictional forces is the diagonal matrix $\mathcal{B} = diag([b_1^l, b_1^w])$, where b_1^l and b_1^w are the viscous frictional coefficients of the link and of the wheel joint. The gravity vector is $\mathcal{N} = [g(m_1^l L_{c,1}^l + m_1^w L_{c,1}^w)\cos(\psi_1), 0]'$. Finally the external torque vector can be written as $\mathcal{T} = [\tau_{E,1}(\psi,\theta), \tau_{C,1}(I_w)]'$, where $\tau_{E,1}(\psi,\theta)$ is the elastic torque acting on the link from the nonlinear elastic elements and can be written as $\tau_{E,1}(\psi,\theta) = \rho(T_1 - T_2)$. ρ is the radius of the pulley, attached to the joint, while T_1 and T_2 are tendon tensions. Tendon tensions are functions of NEE compressions $T_i = \alpha_i\delta_i^2 + \beta_i\delta_i$, $i = 1, 2$, where α_i and β_i are constant parameters of NEEs, δ_i denotes the compressions of NEEs. NEE displacements are functions of link and actuator angular positions $\delta_1 = \delta_0 - \rho(\psi_1 + \frac{\pi}{2}) + \rho_p(\theta_1 - \theta_1^0)$, $\delta_2 = \delta_0 + \rho(\psi_1 + \frac{\pi}{2}) - \rho_p(\theta_2 - \theta_2^0)$. δ_0 denotes the initial compression of the NEEs, ρ_p is the radius of the actuator pulley, θ_i, $i = 1, 2$ are actuator angular positions, and θ_i^0, $i = 1, 2$ are initial actuator positions corresponding to initial compression of NEEs. Torque acting on the reaction wheel $\tau_{C,1}(I_w) = k_w I_w$ depends on the input current I_w

and torque constant k_w of the brushless DC motor (EC40, Maxon) turning the wheel. Brushless motor dynamics can be written as

$$\mathcal{J}_m\ddot{\theta} + \mathcal{B}_m\dot{\theta} + \mathcal{V}_m = \mathcal{S}_m I \tag{24}$$

where $\theta = [\theta_1, \theta_2]'$, θ_1 and θ_2 are angular coordinates of the actuators and $I = [I_1, I_2]'$, I_1 and I_2 are input currents. $\mathcal{J}_m = diag([J_m, J_m])$, $\mathcal{B}_m = diag([b_m, b_m])$ and $\mathcal{S}_m = diag([K_m, K_m])$ are diagonal matrices, where J_m, b_m, K_m are, correspondingly, inertia of the rotor, damping coefficient and overall torque constant of an actuator. The reduction ratio of the gearbox should be taken into account for these parameters. $\mathcal{V}_m = [\tau_{m,1}, -\tau_{m,2}]'$, where $\tau_{m,i} = \rho_p T_i$, $i = 1, 2$ is the external torque applied to each actuator.

The physical parameters of the system used for experiments are $m_1^l = 0.559$ kg, $m_1^w = 0.277$ kg, $J_1^l = 9.64 \cdot 10^{-3}$ kgm^2, $J_1^w = 0.35 \cdot 10^{-3}$ kgm^2, $L_{c,1}^l = 0.166$ m, $L_{c,1}^w = 0.100$ m, $b_1^l = 6.2 \cdot 10^{-3}$ Nms, $b_1^w = 1.5 \cdot 10^{-4}$ Nms, $\rho = 0.013$ m, $\rho_p = 0.022$ m, $\alpha_1 = 12400$ N/m^2, $\beta_1 = 1360$ N/m, $\alpha_2 = 13600$ N/m^2, $\beta_2 = 1350$ N/m, $k_w = 0.131$ Nm/A, $J_m = 1.39 \cdot 10^{-3}$ kgm^2, $b_m = 0.02$ Nms, $K_m = 0.972$ Nm/A, $g = 9.81 m/s^2$.

B. SLMPC Implementation

The state, control and output vectors for our experimental setup are defined as $\mathbf{x} = [\psi_1, \dot{\psi}_1, \dot{\zeta}_1, \theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2]'$, $\mathbf{u} = [I_1, I_2, I_w]'$ and $\mathbf{y} = [\psi_1, \dot{\psi}_1]'$. The state vector $\mathbf{x}$ does not contain the angular position of the reaction wheel ζ_1 because this variable does not influence the system dynamics. Overall, the system has seven states, two outputs and three control inputs. Using these vectors, the nonlinear state-space representation of the system as in (1) can be written as

$$\dot{\mathbf{x}} = F(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \dot{\psi}_1 \\ -\mathcal{M}^{-1}\left(\mathcal{B}\dot{\psi} + \mathcal{N} - \mathcal{T}\right) \\ \dot{\theta}_1 \\ \dot{\theta}_2 \\ -\mathcal{J}_m^{-1}\left(\mathcal{B}_m\dot{\theta} + \mathcal{V}_m - \mathcal{S}_m I\right) \end{bmatrix}. \tag{25}$$

After the linearization and discretization of (25) for the sampling time $T_s = 10$ ms, the approximate system model and the associated QP problem was formulated as described in Section II and illustrated in Fig. 1. The horizon length was set to $n_p = 20$ steps, which resulted in a control horizon of 200 ms.

A 16 s trajectory consisting of large and small step changes, ramps and sinusoidals was designed for the link position reference $\psi_{1,ref}$. The reference for the link velocity was obtained by numerical differentiation of the position reference. The first n_o diagonal elements of the Q weight matrix were set as $Q_{LMPC} = diag([100, 0.1])$. In general, we put the emphasis on the link position tracking and supplement this with a small weight on the link angular velocity to prevent abrupt spikes. The corresponding first n_u elements of R matrix were tuned to $R_{LMPC} = diag([10^{-3}, 10^{-3}, 10^{-1}])$. These weights were used for the other steps in the control horizon as well. Based on the physical limitations of the experimental setup, the constraints on the states $\mathcal{X}$ and bounds on control inputs $\mathcal{U}$ of the system were set as

$$\mathcal{X} = \left\{ \begin{array}{l} -260 \leq \dot{\zeta}_1 \leq 260, \quad -5 \leq \dot{\theta}_{1,2} \leq 5 \ [rad/s]; \\ 0.001 \leq \delta_{1,2} \leq 0.035 \ [m] \end{array} \right\},$$

$$\mathcal{U} = \left\{ \quad -3.1 \leq I_{1,2} \leq 3.1, \quad -1.2 \leq I_w \leq 1.2 \ [A] \ \right\}.$$

For the real-time solution of the QP problem in the form of (22), qpOASES was used at every time step. qpOASES is an open-source QP solver which utilizes an online active set strategy [20], [21]. The code to control the VSA robot was written in C language and run on a desktop computer (Intel i5-4690 CPU and 4 GB memory) with Linux operating system. The formation of the QP problem requires extensive matrix operations. For matrix manipulations, we used Eigen Library [22] due to its simplicity and computational efficiency. Two data acquisition cards (PCI-6259, National Instruments) were used to interface the computer to the physical setup.

C. NMPC Implementation

In order to compare the performance of SLMPC to NMPC, we also implemented NMPC controller following the procedure presented in [19]. ACADO Toolkit was used to generate the NMPC controller [23]. Q and R matrices for the NMPC were set to $Q_{NMPC} = diag([20, 0.01])$ and $R_{NMPC} = diag([10^{-3}, 10^{-3}, 10^{-4}])$ after tuning for best performance. These matrices have different values in the SLMPC and NMPC implementations because of different problem formulations. We could have formulated the NMPC problem in the same way as the SLMPC problem by defining additional states as increments of the control inputs. However, this would lead to undesirable increase of computational time of the controller due to increased number of state variables. Therefore, we tuned both controllers manually for their best performance. Other parameters of the NMPC controller such as the sampling time and horizon length were set to the values used for the SLMPC controller.

IV. Results

Reference tracking of SLMPC and NMPC controllers along with the reaction wheel velocity and input currents are shown in Fig. 3. Both controllers follow the trajectory with small error and the bounds are satisfied except the reaction wheel current exceeding the maximum allowed current instantaneously two times for the NMPC. The root mean square error (RMSE) is defined as $Er = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\psi_1^i - \psi_{1,ref}^i)^2}$, where $n = 1601$ is the number of time steps in the whole experiment. RMSE values were $Er_{NMPC} = 0.54$ and $Er_{SLMPC} = 0.64$ degrees for NMPC and SLMPC experiments, respectively.

The computation time of the control inputs for both controllers were recorded during the experiments. Average computation time for NMPC controller is 2.17 ms and for SLMPC controller 1.25 ms. Qualitative observation of the states and control inputs show that the signals of SLMPC

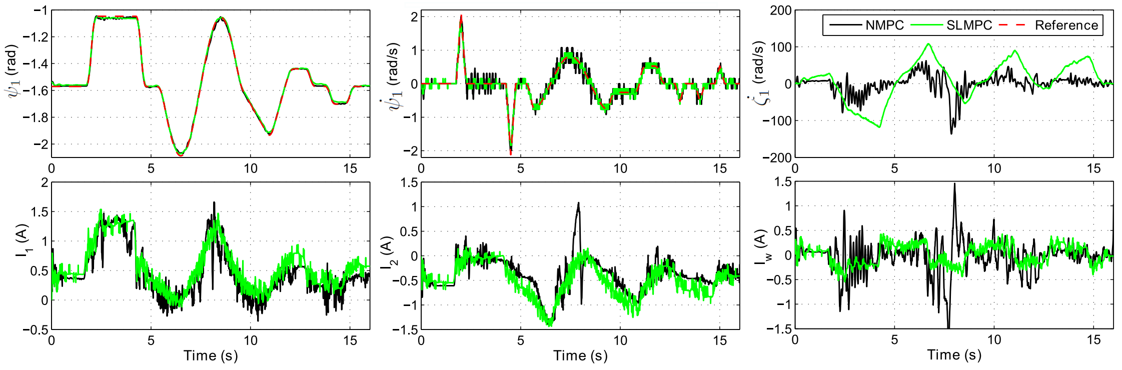

Fig. 3. Experimental values of link position, link and reaction wheel velocities, three input currents for the NMPC and SLMPC cases.

are more smooth. This might be attributed to the $\overrightarrow{\delta \mathbf{u_k}}$ notation which penalizes the changes in the control inputs. Experimental values of two actuator positions and velocities are shown in Fig. 4.

V. Conclusion

In this work, we presented our successive linearization based MPC framework for the control of VSA robots. Experiments conducted with a VSA setup showed that SLMPC provides similar performance to NMPC while requiring less computation time.

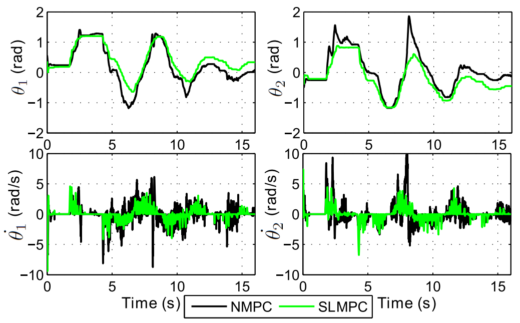

Fig. 4. Experimental actuator positions and velocities for NMPC and SLMPC cases.

References

[1] W. Sebastian, G. Giorgio, E. Oliver *et al.*, "Variable stiffness actuators: Review on design and components," *IEEE/ASME T. Mech.*, vol. 21, no. 5, pp. 2418–2430, 2016.

[2] M. Laffranchi, N. Tsagarakis, and D. G. Caldwell, "A compact compliant actuator (CompAct) with variable physical damping," in *Proc. IEEE Conf. on Robotics and Automation*, 2011, pp. 4644–4650.

[3] A. Jafari, N. G. Tsagarakis, I. Sardellitti *et al.*, "A new actuator with adjustable stiffness based on a variable ratio lever mechanism," *IEEE/ASME T. Mech.*, vol. 19, no. 1, pp. 55–63, 2014.

[4] B. Vanderborght, N. G. Tsagarakis, C. Semini *et al.*, "MACCEPA 2.0: Adjustable compliant actuator with stiffening characteristic for energy efficient hopping," in *Proc. IEEE International Conference on Robotics and Automation*, 2009, pp. 544–549.

[5] M. G. Catalano, G. Grioli, M. Garabini *et al.*, "VSA-CubeBot: a modular variable stiffness platform for multiple degrees of freedom robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2011, pp. 5090–5095.

[6] J. Shintake, B. Schubert, S. Rosset *et al.*, "Variable stiffness actuator for soft robotics using dielectric elastomer and low-melting-point alloy," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 1097–1102.

[7] S. S. Groothuis, G. Rusticelli, A. Zucchelli *et al.*, "The variable stiffness actuator vsaUT-II: Mechanical design, modeling, and identification," *IEEE/ASME T. Mech.*, vol. 19, no. 2, pp. 589–597, 2014.

[8] D. J. Braun, F. Petit, F. Huber *et al.*, "Robots driven by compliant actuators: optimal control under actuation constraints," *IEEE T. Robotics*, vol. 29, no. 5, pp. 1085–1101, 2013.

[9] A. Zhakatayev, M. Rubagotti, and H. A. Varol, "Closed-loop control of variable stiffness actuated robots via nonlinear model predictive control," *IEEE Access*, vol. 3, pp. 235–248, 2015.

[10] ——, "Time optimal control of variable stiffness actuated systems," *IEEE/ASME Transactions on Mechatronics*, 2017.

[11] H. Seki, S. Ooyama, and M. Ogawa, "Nonlinear model predictive control using successive linearization - application to chemical reactors," in *Trans. of the Soc. of Instrument and Control Engineers*, vol. E-3, No. 1, 2004, pp. 66–72.

[12] M. Cannon, D. Ng, and B. Kouvaritakis, "Successive linearization NMPC for a class of stochastic nonlinear systems," in *Nonlinear Model Predictive Control*, L. Magni, D. M. Raimondo, and F. Allgöwer, Eds. Springer Berlin Heidelberg, 2009, pp. 249–262.

[13] A. Cortinovis, H. J. Ferreau, D. Lewandowski *et al.*, "Safe and efficient operation of centrifugal compressors using linearized MPC," in *Proc. IEEE Conference on Decision and Control*, 2014, pp. 3982–3987.

[14] C. Liu, A. Carvalho, G. Schildbach *et al.*, "Stochastic predictive control for lane keeping assistance systems using a linear time-varying model," in *Proc. American Control Conference*, 2015, pp. 3355–3360.

[15] A. Morsi, H. S. Abbas, and A. M. Mohamed, "Model predictive control of a wind turbine based on linear parameter-varying models," in *Proc. IEEE Conf. on Control Applications*, 2015, pp. 318–323.

[16] I. Bonis, W. Xie, and C. Theodoropoulos, "A linear model predictive control algorithm for nonlinear large-scale distributed parameter systems," *AlChE Journal*, vol. 58, no. 3, pp. 801–811, 2011.

[17] H. Zheng, R. R. Negenborn, and G. Lodewijks, "Trajectory tracking of autonomous vessels using model predictive control," in *Proc. World Congress of the Int. Fed. of Automatic Control*, 2014, pp. 8812–8818.

[18] A. Bamimore, O. Taiwo, and R. King, "Comparison of two nonlinear model predictive control methods and implementation on a laboratory three tank system," in *Proc. IEEE Conference on Decision and Control and ECC*, 2011, pp. 5242–5247.

[19] A. Baimyshev, A. Zhakatayev, and H. A. Varol, "Augmenting variable stiffness actuation using reaction wheels," *IEEE Access*, vol. 4, pp. 4618–4628, 2016.

[20] H. J. Ferreau, C. Kirches, A. Potschka *et al.*, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, pp. 1–37, 2013.

[21] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, vol. 18, no. 8, pp. 816–830, 2008.

[22] G. Guennebaud, B. Jacob *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010.

[23] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit - An open-source framework for automatic control and dynamic optimization," *Opt. Contr. Appl. Methods*, vol. 32, no. 3, pp. 298–312, 2011.