# GPU-Based Parameterized NMPC Scheme for Control of Half Car Vehicle With Semi-Active Suspension System

Karthik Murali Madhavan Rathai, Olivier Sename, and Mazen Alamir

***Abstract*—In this letter, we propose a black-box compatible simulation-based approach for solving nonlinear model predictive control (NMPC) problem via a parameterized technique to control the vertical dynamics of a half car vehicle equipped with semi-active (SA) suspension system. The method taps the potential of the graphic processing unit (GPU) to simulate the system parallelly for several combinations of control inputs and the optimal input is elicited which minimizes the objective function and satisfies the constraints. The method was tested in MATLAB/Simulink environment by means of simulations and a comparative study was conducted with ACADO-qpOASES NMPC framework. The simulation results display better performance of the proposed approach in terms of computation time, closed loop objective, and constraint satisfaction when juxtaposed to the ACADO-qpOASES NMPC controller.**

***Index Terms*—Automotive control, predictive control for nonlinear systems, control applications.**

## I. Introduction

OVER the last decade, with the induction of autonomous vehicles on public roads, a tremendous amount of emphasis has been placed on automotive research and development both in industry as well as academia. In this regard, in recent years advanced control methods has garnered immense attention due to the requirements on quality, safety and performance. Given this prelude, in this letter we address vertical dynamics control problem and more specifically on NMPC of a SA suspension system for half car vehicle. The prime functionality of the suspension controller is to guarantee safety and comfort for the onboard passengers. However this design task poses several control and modeling challenges. On one end there are inherent system nonlinearities, state/input constraints, partially modeled dynamics etc. and on the other end there are performance requirements (comfort and safety), real-time (RT) adaptability and operability etc. and the control engineer is coerced to walk along a tight rope between the two. Thus, a meticulous care ought to be undertaken in the design process to balance this tradeoff. In this regard, MPC provides a systematic framework to deal with system constraints, nonlinearities, objective/performance requirements and also, with increased computational power these days, MPC proves to be a viable option for RT constrained control.

The underlying idea (receding horizon control) behind the MPC scheme is to find the optimal input sequence from an optimal control problem (OCP) over a look ahead period given the current state of the system. From the obtained solution, the first optimal input injected into the system and this procedure is repeated at the next sampling period with the receipt of the propagated state. The two rudimental approaches to deal with this optimization problem involves a) derivative based methods and b) derivative free methods [1]. The former method finds the optimal solution by solving via iterative methods while the latter searches for the optimal solution by means of simulation. In this letter, the latter approach is adopted and the simulations are computed in parallel by virtue of Graphic Processing Unit (GPU).

Along the line of research for control of suspension systems, several contributions has been proposed such as (to name a few) Skyhook controller, Mixed Sky-Hook and ADD controller, $H_\infty$ and LPV methods [2], MPC approaches [3]–[5] etc. See [6] for a detailed exposition on different control methods. In regards with parallely solving the MPC problem, there has been several research conducted in the past which harnesses the potential of GPU. A brief tutorial on different parallel architectures for MPC is proposed in [7]. In [8], scenario approach based stochastic MPC was implemented for drinking water network system by means of accelerated proximal gradient method. In [9], a path integral based MPC method was proposed and experimentally validated for a RC car. In [10], several GPU based interior point methods were developed for linear MPC framework. In [11], a sampling based parallelized nonlinear MPC (NMPC) scheme was proposed and the method was experimentally validated for an inverted pendulum system.

Manuscript received February 26, 2019; revised April 23, 2019; accepted April 25, 2019. Date of publication May 6, 2019; date of current version May 20, 2019. This work was supported by the ITEA3 European Project through EMPHYSIS (Embedded Systems With Physical Models in the Production Code Software) under Grant 15016. Recommended by Senior Editor R. S. Smith. *(Corresponding author: Karthik Murali Madhavan Rathai.)*

The authors are with the CNRS, GIPSA-lab, Grenoble INP, Université Grenoble Alpes, 38000 Grenoble, France (e-mail: karthik.murali-madhavan-rathai@grenoble-inp.fr; olivier.sename@grenoble-inp.fr; mazen.alamir@grenoble-inp.fr).

Digital Object Identifier 10.1109/LCSYS.2019.2915002

In this letter we propose a parallelized parameterized NMPC (pNMPC) scheme for a real half car vehicle equipped with SA suspension system. The method taps the power of GPU computing for solving the automotive suspension control problem. The key aspects of the method are:

a) **Black-box model compatibility** - It is not uncommon in automotive industry that only a black-box model (simulation code) is provided in the process of model exchange between the OEMs and supply/service provider due to protection of intellectual property (IP) rights. In the interest of implementation of NMPC, the proposed simulation based approach obviates any need to stipulate the structure of dynamics, objective or constraint functions, whereas the existing derivative based methods either with GPU or otherwise, typically enforces and exploits the structure of dynamics, objective and constraint functions to accelerate the controller.

b) **Efficient RT operability** - The fast computation of the control input by means of GPU renders the method to be RT operable, especially for fast sampled automotive systems.

c) **Automotive standard** - The method is amenable with the existing automotive standards as only simple math operations are required and also, independent of any optimization solver.

This letter is organized as follows. Section II details the half car mathematical model and the quasi-nonlinear SA suspension model. Section III entails the objective and constraints requirements for the MPC scheme. Section IV briefly discusses the parallelized pNMPC method and its implementation in NVIDIA CUDA based GPUs. In Section V, the analysis and simulation results are presented, where comparative study between ACADO-qpOASES NMPC controller and the proposed parallelized pNMPC method are conducted. This letter is finally ended with conclusions and future works in Section VI.

## II. Half Car Model With Semi-Active Suspension System

### A. Half Car Mathematical Model

The half car vertical dynamics model is a 4 degrees of freedom (DOF) model [2], which involves chassis dynamics (heave motion), roll dynamics and dynamics of the two unsprung masses (wheels). The model ought to be viewed as the vehicle being scrunched from the front and rear ends into a single block. Let the left and right corner of the vehicle be indexed with $i \in \{l, r\}$ respectively. The 4 DOF mathematical model is expressed with the following equations

$$\begin{cases} m_s \ddot{z}_s = -\sum_{i\in\{l,r\}} F_{s,i} \\ I_x \ddot{\theta} = (l_l F_{s,l} - l_r F_{s,r}) \\ m_{us,l} \ddot{z}_{us,l} = (-F_{s,l} + F_{t,l}) \\ m_{us,r} \ddot{z}_{us,r} = (-F_{s,r} + F_{t,r}) \end{cases} \tag{1}$$

where, $z_s$ and $\theta$ represents the chassis mass position with respect to centre of gravity (COG) and roll angle respectively. $z_{us,i}$ $\forall i \in \{l, r\}$ represents the unsprung mass position for the left and right corners respectively. $m_s$, $m_{us,l}$, $m_{us,r}$ represent the chassis mass, unsprung masses for the left and right corners. $I_x$ represents the moment of inertia along the roll axis. $l_l$ and $l_r$ represents the length of the chassis from the left and right corners with respect to COG. $F_{s,i}$ represents the chassis forces and $F_{t,i}$ represents the wheel forces $\forall i \in \{l, r\}$ which are expressed with

$$\begin{aligned} F_{s,i} &= -k_{s,i}(z_{s,i} - z_{us,i}) + u_i \\ F_{t,i} &= -k_{t,i}(z_{us,i} - z_{r,i}) \end{aligned} \tag{2}$$

where, $k_{s,i}$ and $k_{t,i}$ represents the stiffness coefficent of the SA suspension system and wheel respectively. $z_{r,i}$ and $z_{us,i}$ represents the vertical road displacement and unsprung mass position $\forall i \in \{l, r\}$. $u_i$ represents the actuation force obtained from the nonlinear SA damper model (see Section II-B). $z_{s,i}$ represents the sprung mass displacement at each corner which are obtained from the following equations

$$\begin{aligned} z_{s,l} &= z_s + l_l \sin\theta \\ z_{s,r} &= z_s - l_r \sin\theta. \end{aligned} \tag{3}$$

### B. Nonlinear Quasi-Static SA Damper Model

The SA damper force $u_i$ (2) is expressed by utilizing the Guo's damper force model [2] $\forall i \in \{l, r\}$ with

$$u_i = k_0 z_{d,i} + c_0 \dot{z}_{d,i} + f_c \phi_i \tanh(a_1 \dot{z}_{d,i} + a_2 z_{d,i}) \tag{4}$$

where $k_0$, $c_0$, $f_c$, $a_1$ and $a_2$ represent the damper stiffness coefficient, viscous damping coefficient, dynamic yield force of the fluid, hysteresis coefficient due to velocity and position respectively. $\phi_i$, $\forall i \in \{l, r\}$ represents the duty cycle (PWM-DC) input signal which manipulates the damper characteristics online by changing the input voltage. $z_{d,i} = z_{s,i} - z_{us,i}$ and $\dot{z}_{d,i} = \dot{z}_{s,i} - \dot{z}_{us,i}$ represents the deflection position and velocity $\forall i \in \{l, r\}$ between the chassis and wheel respectively.

Let $\mathbf{X} = [z_s, \theta, z_{us,l}, z_{us,r}, \dot{z}_s, \dot{\theta}, \dot{z}_{us,l}, \dot{z}_{us,r}]$ denote the state vector, $\mathbf{U} = [\phi_l, \phi_r]$ denote the input vector and $\mathbf{D} = [z_{r,l}, z_{r,r}]$ denote the disturbance vector, then the half car model (1) can be compactly expressed with

$$\dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{U}(t), \mathbf{D}(t)) \tag{5}$$

where $\mathbf{X} \in \mathbb{R}^8$, $\mathbf{U} \in \mathbb{R}^2$ and $\mathbf{D} \in \mathbb{R}^2$. The parameters are obtained from the INOVE test platform at GIPSA lab, Grenoble. The INOVE test platform discussed is a 1:5-scaled baja style racing car which consists of 4 controllable Electro-Rheological (ER) SA dampers (ER-SA) and 4 DC motors to generate different road profiles for each wheel corner [12]. The sampling period ($T_s$) of the INOVE platform is $5ms$. The model parameters are listed in [5].

## III. Model Predictive Control Design

### A. Objective Requirements

The objective design can be briefly classified as a) comfort and b) ride handling objective [2].

1) *Comfort objective:* The goal of the comfort based objective is to minimize the vertical acceleration of the chassis ($\ddot{z}_s$), obtained from (1). The comfort objective for a given look ahead period $T_l$ is expressed with

$$J^{acc}_{T_l}(\mathbf{X}(.), \mathbf{U}(.), \mathbf{D}(.)) = \int_0^{T_l} (\ddot{z}_s(t))^2 dt \tag{6}$$

2) *Ride handling objective:* The goal of the ride handling objective is to minimize the roll angle ($\theta$) of the vehicle.

The ride handling objective for a given look ahead period $T_l$ is expressed with

$$J^{roll}_{T_l}(\mathbf{X}(.), \mathbf{U}(.), \mathbf{D}(.)) = \int_0^{T_l} (\theta(t))^2 dt. \tag{7}$$

### B. Constraint Requirements

The constraints for the SA suspension system primarily arise from the physical limitations and secondarily from performance requirements [13]. For the MPC design, the included constraints are

1) *ER-SA damper input constraints:*
   a) *Damper force constraint (Physical):* The ER-SA damper force is bounded, i.e., $u_i \in [u_{min,i}, u_{max,i}]$, $\forall i \in \{l, r\}$.
   b) *PWM-DC input constraints:* The operating DC for the PWM signal is constrained to $\phi_i \in [\phi_{min,i}, \phi_{max,i}]$, $\forall i \in \{l, r\}$.
2) *State constraints:*
   a) *Stroke deflection constraint (Physical):* This forms a linear state constraint, i.e., $z_{d,i} \in [z_{dmin,i}, z_{dmax,i}]$, $\forall i \in \{l, r\}$.
   b) *Peak acceleration constraint (Performance):* This bounds the peak acceleration of the chassis mass, i.e., $\ddot{z}_s \in [\ddot{z}_{min,s}, \ddot{z}_{max,s}]$.
   c) *Wheel rebound constraint (Performance):* This bounds the deflection position between the wheel and road. This is to ensure the tyre deflection forces are bounded, i.e., $z_{us,i} - z_{r,i} \in [z_{rebmin,i}, z_{rebmax,i}]$, $\forall i \in \{l, r\}$.
   d) *Unsprung mass displacement constraint (Performance):* This bounds the displacement of the unsprung mass. This reinforces the road holding condition for the vehicle, i.e., $z_{us,i} \in [z_{usmin,i}, z_{usmax,i}]$, $\forall i \in \{l, r\}$.
3) *Road disturbance assumption:* The road disturbance is assumed to be constant over the prediction horizon ($T_l$) with $\mathbf{D}(0) = \mathbf{D}_0$, i.e., the road profile measured at the current time instant by means of observers [14]. There is no loss of generality with this formulation as it can be easily extendable with any road models such as ISO road profiles or from road preview sensors.

The mixed input-state constraint set is compactly expressed with $(\mathbf{X}, \mathbf{U}) \in \Omega_{(\mathbf{X},\mathbf{U})} \subset \mathbb{R}^8 \times \mathbb{R}^2$, the input constraint set is compactly expressed with $\mathbf{U} \in \Omega_{\mathbf{U}} \subset \mathbb{R}^2$ and the state constraint set is compactly expressed with $\mathbf{X} \in \Omega_{\mathbf{X}} \subset \mathbb{R}^8$.

### C. MPC Problem Formulation

In summary, with the proposed objective and constraints function, the NMPC OCP is casted as

$$\begin{aligned} J_{obj}(\mathbf{X}_0, \Gamma_0, \mathbf{D}_0, \mathbf{U}(.)) = \min_{\mathbf{U}(.)} \ & \Gamma^1_0 J^{acc}_{T_l} + \Gamma^2_0 J^{roll}_{T_l} \\ \text{subject to } & \dot{\mathbf{X}}(t) = f(\mathbf{X}(t), \mathbf{U}(t), \mathbf{D}(t)) \\ & \mathbf{X}_0 = \mathbf{X}(0), \mathbf{D}(t) = \mathbf{D}_0 \\ & \mathbf{X}(t) \in \Omega_{\mathbf{X}}, \mathbf{U}(t) \in \Omega_{\mathbf{U}} \\ & (\mathbf{X}(t), \mathbf{U}(t)) \in \Omega_{(\mathbf{X},\mathbf{U})} \end{aligned} \tag{8}$$

where $\Gamma_0 = [\Gamma^1_0, \Gamma^2_0]$ with $\Gamma^1_0$ and $\Gamma^2_0$ being the convex combination weights between the two objectives. Once the optimal input trajectory is computed, the first control action in $\mathbf{U}^*$ is injected into the system and this procedure is repeated in receding horizon fashion. In this letter, a constant input profile is assumed over the control horizon.

## IV. Parallelized pNMPC Method

The crux of the method is to finitely parameterize the input constraint set $\Omega_{\mathbf{U}}$ and by virtue of parallel computing methods, the system (5) is simulated for every parameterized input given the current state, disturbance and objective specification information. The optimal input is elicited which minimizes the objective and satisfies the constraint in (8) (see [5] for more details). The pseudo-code for the implementation of parallelized pNMPC is shown in Algorithm 1.

The pseudo-code deserves some explanation to elucidate its working principle. The details enclosed in this section provides the reader with the need to know basics to handle parallel computing using the NVIDIA CUDA GPUs for implementation of the proposed parallelized pNMPC method. The pseudo-code is written in a manner to benefit the reader to easily connect with CUDA-C implementation. For the respective application programming interfaces (APIs) and syntax details refer [15].

*Explanation of Implementation:*

1) *Initialization, I/Os and Syntax declaration:*
   a) The first step is the data initialization where the model/constraint parameters and GPU parameters are initialized. $b_s$, $g_s$ represents the size of the grid and blocks respectively [15]. $n_{\phi_l}$, $n_{\phi_r}$ represents the number of quantized levels of the PWM-DC input signal for the left/right corner of the vehicle respectively.
   b) The input variables for the method are $\mathbf{X}_0, \mathbf{D}_0, \Gamma_0$ and the output variable is $\mathbf{U}^*$.
   c) The decorators __HOST__, __GLOBAL__, __DEVICE__ denotes the function call made from host (CPU) to host, host to device (GPU) and device to device respectively. The first function invoked is the **MAIN**.
2) **MAIN** function:
   a) The lines 2–5 dynamically allocates memory (DMA) in the host and the device for the objective and inputs for every input combination.
   b) In line 6, the **KER** function is launched with appropriate launch parameters $g_s$ and $b_s$.
   c) The lines 7–8 transfers the computed objective and input data from the device to the host.
   d) The lines 9–11 returns the optimal input $\mathbf{U}^*$ by finding the index of the minimum objective or constraint violation.
3) **KER** function:
   a) The lines 14 and 15 sets the thread indices for each PWM-DC input combination. From this point onward, each thread parallely computes the solution for each input combination.
   b) The line 16 serializes the 2D grid into a single vector for objective and input vector designation.

c) The line 17 is a check condition to make sure the thread access is not exceeded.
d) The lines 20 and 21 obtains the input combination and objective value from the functions **GRID2D** and **ODESIM2OBJ**.

4) **GRID2D** function:
a) The line 24–25 sets the PWM-DC input values for left and right corner of the vehicle.
b) The function assigns the input values for every thread call, i.e., for all input combinations.

5) **ODESIM2OBJ** function:
a) The line 29 initializes the objective and constraint violation variable to zero and line 30 invokes the **GRID2D** function for specific thread indices which corresponds to an input combination.
b) The lines 31–40 runs the Euler integration scheme for the system (5) until the look ahead period $T_l$ with an integration step of $h$. In case the constraints are violated, the objective is set to a very high value `MAX` and the constraint violation is quantified with **N** function, which computes the 2-norm for the constraints. Otherwise, the objective in (8) is computed numerically.
c) The line 41 returns the sum of objective and constraint violation for the given thread indices.

## V. Analysis and Simulation Results

The conducted simulation study can be broadly classified into three parts which are A) Computational time (CT) analysis b/w CPU (serial) and GPU (parallel) for the proposed pNMPC method, B) Comparative analysis b/w ACADO-qpOASES NMPC controller [16] and the proposed parallelized pNMPC method and C) Performance analysis with a road profile test. Both methods were implemented in MATLAB/Simulink on a Intel Core i7 PC and NVIDIA GTX 1050Ti with 768 CUDA cores. Code generation option was utilized for ACADO-qpOASES NMPC controller and the proposed parallelized pNMPC was programmed in CUDA C and patched into Simulink with S-function.

### A. Computational Time Analysis b/w CPU and GPU

The raison d'être for conducting this analysis is to emphasize the significance of GPUs for solving huge simulations and viability of the approach for control of real SA suspension system. From Fig. 1, it is evident that the pNMPC method fares well in GPU compared to CPU in terms of mean CT per sampling period. The abscissa indicates the number of discretized inputs for the PWM-DC signal for both left ($n_{\phi_l}$) and right ($n_{\phi_r}$) corner of the vehicle (i.e., $n_{\phi_l} \times n_{\phi_r}$ input combinations). The road profile involved a chirp signal (same profile for both corners) with an amplitude of 2.5$mm$ and frequency sweep from $1-14Hz$ for a duration of 10 $s$. The objective was comfort ($\Gamma_0^1 = 1, \Gamma_0^2 = 0$).

### B. Comparative Analysis

A comparative analysis was conducted b/w ACADO-qpOASES NMPC controller and the proposed parallelized pNMPC method. The basis for conducting this letter was to analyze computational time (CT), normalized closed-loop objective (NCLO), i.e., with respect to nominal damping (see Section V-C) and feasibility analysis (FA), i.e., constraint satisfaction. The study involved a gradual increase in the complexity parameter and the aforementioned criteria were

**Algorithm 1** Parallelized pNMPC Implementation

**Data Initialization**: Model/Constraint parameters, $b_s$, $g_s$ $n_{\phi_l}$, $n_{\phi_r}$
**Input**: $\mathbf{X}_0, \mathbf{D}_0, \Gamma_0$
**Output**: $\mathbf{U}^*$

1: **function** __HOST__**MAIN**($\mathbf{X}_0, \mathbf{D}_0, \Gamma_0$)
2: **HostDMAObj**($h_{obj}$)
3: **HostDMAInp**($h_{inp}$)
4: **DeviceDMAObj**($d_{obj}$)
5: **DeviceDMAInp**($d_{inp}$)
6: **KER**$_{\{g_s,b_s\}}$($d_{obj}, d_{inp}, \mathbf{X}_0, \mathbf{D}_0, \Gamma_0$)
7: **DevicetoHostCpy**($h_{obj}, d_{obj}$)
8: **DevicetoHostCpy**($h_{inp}, d_{inp}$)
9: $k_{opt} \leftarrow$ **indexmin**($h_{obj}$)
10: $\mathbf{U}^* \leftarrow h_{inp}(k_{opt})$
11: **return** $\mathbf{U}^*$
12: **end function**

13: **function** __GLOBAL__**KER**$_{\{b_s,g_s\}}$($d_{obj}, d_{inp}, \mathbf{X}_0, \mathbf{D}_0, \Gamma_0$)

14: $i \leftarrow$ `bIdx.x*bDim.x+tIdx.x`
15: $j \leftarrow$ `bIdx.y*bDim.y+tIdx.y`
16: $r \leftarrow n_{\phi_l} i + j$
17: **if** $i \geq n_{\phi_l} \vee j \geq n_{\phi_r}$ **then**
18: **return**
19: **end if**
20: $d_{inp}[r] \leftarrow$ **GRID2D**($i, j$)
21: $d_{obj}[r] \leftarrow$ **ODESIM2OBJ**($\mathbf{X}_0, \mathbf{D}_0, \Gamma_0, i, j$)
22: **end function**

23: **function** __DEVICE__**GRID2D**($i, j$)
24: $\phi_{i,l} = \phi_{min,l} + \frac{i}{n_{\phi_l}-1}(\phi_{max,l} - \phi_{min,l})$
25: $\phi_{j,r} = \phi_{min,r} + \frac{j}{n_{\phi_r}-1}(\phi_{max,r} - \phi_{min,r})$
26: **return** $\{\phi_{i,l}, \phi_{j,r}\}$
27: **end function**

28: **function** __DEVICE__ **ODESIM2OBJ**($\mathbf{X}_0, \mathbf{D}_0, \Gamma_0, i, j$)
29: `Obj` $= 0$, `Con` $= 0$
30: $\mathbf{U}_{i,j} \leftarrow$ **Grid2D**($i, j$)
31: **for** $t_{loop} = 0 : h : T_l$ **do**
32: $\mathbf{X}^+ \leftarrow \mathbf{X}_0 + hf(\mathbf{X}_0, \mathbf{U}_{i,j}, \mathbf{D}_0)$
33: **if** $\mathbf{X}^+ \notin \Omega_{(\mathbf{X},\mathbf{U})} \vee \mathbf{X}^+ \notin \Omega_{(\mathbf{X})}$ **then**
34: `Obj` $=$ `MAX`
35: `Con` = `Con` + **N**($\Omega_{(\mathbf{X},\mathbf{U})}, \Omega_{(\mathbf{X})}$)
36: **else**
37: `Obj` $=$ `Obj` $+ h(\Gamma_0^1 J_{l_l}^{acc} + \Gamma_0^2 J_{l_l}^{roll})$
38: **end if**
39: $\mathbf{X}_0 = \mathbf{X}^+$
40: **end for**
41: **return** {`Obj` + `Con`}
42: **end function**

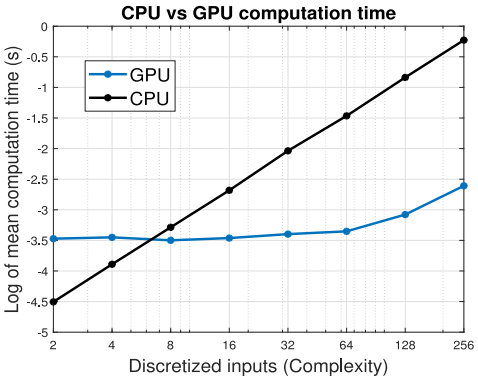


Fig. 1. CPU vs GPU pNMPC computation time.

TABLE I
ACADO-QPOASES NMPC CONTROLLER

| $N_s$ | FA | Mean CT (ms) | Max CT (ms) | NCLO |
|---|---|---|---|---|
| 5 | ✗ | 0.70 | 1.6 | ——— |
| 10 | ✗ | 0.95 | 2.2 | ——— |
| 15 | ✓ | 1.3 | 2.9 | 0.6484 |
| 20 | ✓ | 1.5 | 3.0 | 0.6412 |
| 25 | ✓ | 1.9 | 3.9 | 0.6317 |

TABLE II
PARALLELIZED PNMPC METHOD

| $\{n_{\phi_l}, n_{\phi_r}\}$ | FA | Mean CT (ms) | Max CT (ms) | NCLO |
|---|---|---|---|---|
| {2, 2} | ✓ | 0.35 | 0.62 | 0.4679 |
| {4, 4} | ✓ | 0.35 | 0.60 | 0.4640 |
| {8, 8} | ✓ | 0.35 | 0.61 | 0.4646 |
| {16, 16} | ✓ | 0.36 | 0.55 | 0.4588 |
| {32, 32} | ✓ | 0.41 | 0.67 | 0.4568 |

recorded. The complexity parameter for ACADO-qpOASES NMPC controller was the number of Newton iterations ($N_s$), i.e., `IMPLICIT_INTEGRATOR_NUM_ITS` (see [16]) and other settings were set to default and the number of discretization points $\{n_{\phi_l}, n_{\phi_r}\}$ (left/right corner of the vehicle) for parallelized pNMPC method. The road profile utilized was the same as mentioned in Section V-A and the objective was comfort ($\Gamma_0^1 = 1, \Gamma_0^2 = 0$).

The ✗ and ✓ indicates the infeasibility and feasibilty of the imposed constraints for the system. The recorded readings are listed in Table I and II and from the tables, it is evident that the proposed parallelized pNMPC method performs better than ACADO-qpOASES NMPC controller in all criteria.

### C. Road Profile Simulation Test - Ride Handling

The experiment involved a lopsided bump road profile with an amplitude of $4mm$ with ride handling objective ($\Gamma_0^1 = 0, \Gamma_0^2 = 1$) and the duration of simulation was $10s$. The road profile is illustrated in Fig. 2. Three different methods were adopted to analyze the performance of the system which are 1) Nominal damping, 2) ACADO-qpOASES NMPC controller and 3) Parallelized pNMPC method. The settings for each method are listed below

1) Nominal damping (Passive): The PWM-DC input for both left and the right corner was set to a constant value 0.225, i.e., the case when the damper control is switched off and the system is passive.

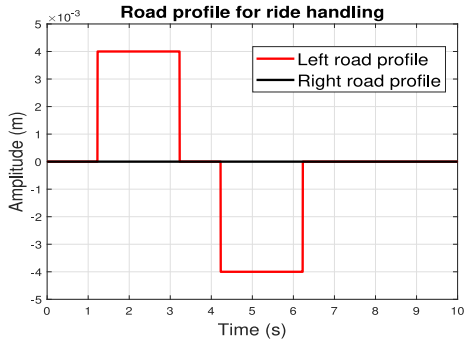


Fig. 2. Road profile for ride handling.

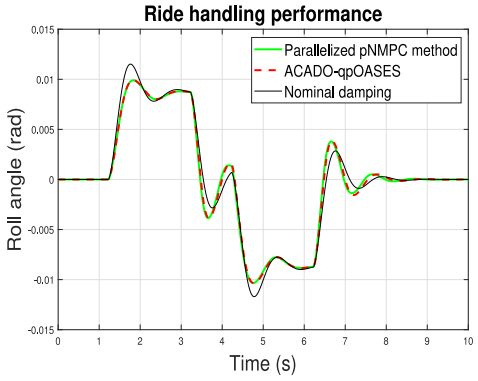


Fig. 3. Ride handling - roll angle $\theta$.

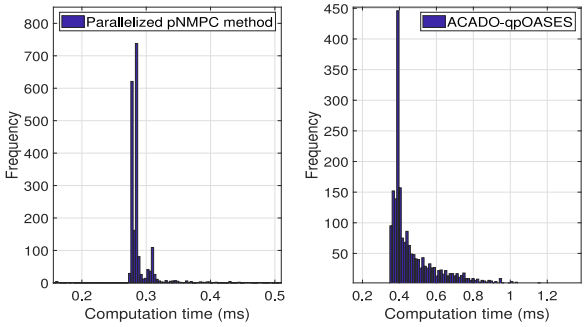


Fig. 4. Computational time distribution.

2) ACADO-qpOASES NMPC controller: The corresponding best setting was utilized, **Integrator** - 4th order Runge Kutta integrator, **QP solver** - qpOASES, **Hessian approximation** - Gauss-Newton, **Discretization** - Multiple shooting, **Discretization intervals** - 5 and for the rest, default parameters were utilized.
3) Parallelized pNMPC controller: The number of discretization points (complexity) was $\{n_{\phi_l}, n_{\phi_r}\} = \{8, 8\}$ for both left and right corners of the vehicle.

Fig. 3 illustrates the roll angle ($\theta$) and it is clearly seen that the overshoot in the nominal damping is larger than the other two methods. In comparing the ACADO-qpOASES NMPC and proposed parallelized pNMPC method, the latter performs slightly better than the former. However the key aspects here are the computation time and the input profile.

The histogram of computation time for both the methods are displayed in Fig. 4. The mean computation time are

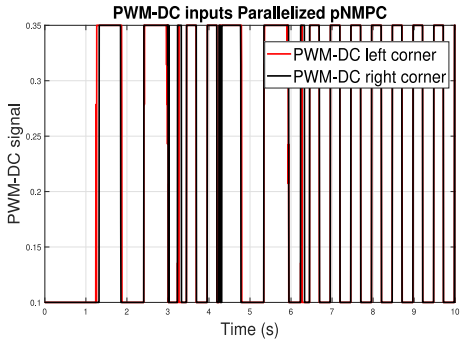


Fig. 5. PWM-DC input for Parallelized pNMPC method.

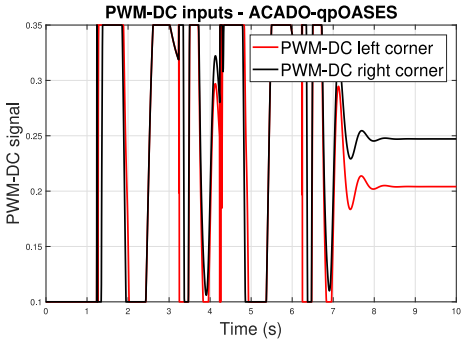


Fig. 6. PWM-DC input for ACADO-qpOASES NMPC controller.

0.28 ms and 0.48 ms and maximum computation time are 0.51 ms and 1.6 ms for parallelized pNMPC method and ACADO-qpOASES NMPC controller respectively. A significant reduction in the computation time is observed for nearly the same performance of the system. The PWM-DC input for parellelized pNMPC method and ACADO-qpOASES NMPC controller are displayed in Fig. 5 and Fig. 6. A key point to be noted is that the input profile rendered by ACADO-qpOASES NMPC controller is unrealistic and cannot be practically injected into the real SA suspension system (INOVE test platform) due to the limitations of the set of operational input levels of the PWM-DC signal, i.e., quantized levels. On the contrary, this can be easily included into the proposed parallelized parameterized NMPC (pNMPC) method and this is illustrated in Fig. 5.

## VI. Conclusion and Future Works

This letter was conducted under the purview of the ITEA3 European project EMPHYSIS, i.e., Embedded system with Physical models in the production code Software with the premise on demanding needs from industrial partners for optimal performance at fast sampling rates for highly nonlinear automotive systems. The proposed parallelized pNMPC method fares well vis-á-vis to the best off the shelf NMPC framework - ACADO-qpOASES and also aligns with the project goals. Since, the proposed parallelized pNMPC method is amenable with any nonlinear model, in a real vehicle setting the nonlinear simulation model developed by the industrial partners can directly be embedded into the proposed control method without any modification. Thus, the proposed method would provide much better performance as it tackles the control problem directly with the developed model rather than shaping the dynamics, objectives or constraints to fit a specific optimization framework. For the future works, the following are in the pipeline - a) Extend the method for full car model, b) Perform HIL tests with dSPACE MicroAutoBox II with ACADO-qpOASES and CPU version of pNMPC method, c) Include complex road models to reflect real world scenario, d) Utilize GPU for implementing the proposed parallelized pNMPC method for INOVE test platform [12].

## References

[1] M. Alamir, "The PDF-MPC package: A free-MATLAB-coder package for real-time nonlinear model predictive control," *CoRR*, vol. abs/1703.08255, 2017. [Online]. Available: http://arxiv.org/abs/1703.08255

[2] S. M. Savaresi, C. Poussot-Vassal, C. Spelta, O. Sename, and L. Dugard, *Semi-Active Suspension Control Design for Vehicles*, Amsterdam, The Netherlands: Elsevier, 2010.

[3] M. Canale, M. Milanese, and C. Novara, "Semi-active suspension control using 'fast' model-predictive techniques," *IEEE Trans. Control Syst. Technol.*, vol. 14, no. 6, pp. 1034–1046, Nov. 2006.

[4] N. Giorgetti, A. Bemporad, H. E. Tseng, and D. Hrovat, "Hybrid model predictive control application towards optimal semi-active suspension," *Int. J. Control*, vol. 79, no. 5, pp. 521–533, 2006.

[5] K. M. M. Rathai, M. Alamir, O. Sename, and R. Tang, "A parameterized NMPC scheme for embedded control of semi-active suspension system," *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 301–306, 2018.

[6] H. E. Tseng and D. Hrovat, "State of the art survey: Active and semi-active suspension control," *Veh. Syst. Dyn.*, vol. 53, no. 7, pp. 1034–1062, 2015.

[7] M. Koegel and R. Findeisen, "Parallel architectures for model predictive control," in *Proc. 4th IFAC Nonlin. Model Predictive Control Conf.*, vol. 4, 2012, pp. 138–143.

[8] A. K. Sampathirao, P. Sopasakis, A. Bemporad, and P. P. Patrinos, "GPU-accelerated stochastic predictive control of drinking water networks," *IEEE Trans. Control Syst. Technol.*, vol. 26, no. 2, pp. 551–562, Mar. 2018.

[9] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 1433–1440.

[10] N. F. Gade-Nielsen, B. Dammann, and J. B. Jørgensen, "Interior point methods on GPU with application to model predictive control," Ph.D. Dissertation, Dept. Appl. Math. Comput. Sci., Tech. Univ. Denmark, Lyngby, Denmark, 2014.

[11] S. Ohyama and H. Date, "Parallelized nonlinear model predictive control on GPU," in *Proc. IEEE 11th Asian Control Conf. (ASCC)*, 2017, pp. 1620–1625.

[12] O. Sename. *Integrated Approach for Observation and Control of Vehicle Dynamics (INOVE)*. Accessed: Apr. 23, 2019. [Online]. Available: http://www.gipsa-lab.fr/projet/inove/index.html

[13] A. E. Baumal, J. J. McPhee, and P. H. Calamai, "Application of genetic algorithms to the design optimization of an active vehicle suspension system," *Comput. Methods Appl. Mech. Eng.*, vol. 163, nos. 1–4, pp. 87–94, 1998.

[14] M. Doumiati, J. Martinez, O. Sename, L. Dugard, and D. Lechner, "Road profile estimation using an adaptive Youla–Kučera parametric observer: Comparison to real profilers," *Control Eng. Pract.*, vol. 61, pp. 270–278, Apr. 2017.

[15] J. Sanders and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming*. Upper Saddle River, NJ, USA: Addison-Wesley Prof., 2010.

[16] D. Ariens, B. Houska, H. Ferreau, and F. Logist, "ACADO for MATLAB user's manual," *Optim. Eng. Center*, vol. 1, 2010.