2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Real-Time Constrained Nonlinear Model Predictive Control on $\mathrm{SO}(3)$ for Dynamic Legged Locomotion*

Seungwoo Hong[1], Joon-Ha Kim[1], Hae-Won Park[1]

***Abstract*— This paper presents a constrained nonlinear model predictive control (NMPC) framework for legged locomotion. The framework assumes a legged robot as a floating base single rigid body with contact forces being applied to the body as external forces. With consideration of orientation dynamics evolving on the rotation manifold $\mathrm{SO}(3)$, analytic Jacobians which are necessary for constructing the gradient and the Gauss-Newton Hessian approximation of the objective function are derived. This procedure also includes the reparameterization of the robot orientation on $\mathrm{SO}(3)$ to orientation error in the tangent space of that manifold. Obtained gradient and Gauss-Newton Hessian approximation are utilized to solve nonlinear least squares problems formulated from NMPC in a computationally efficient manner. The proposed algorithm is verified on various types of legged robots and gaits in a simulation environment.**

I. Introduction

Animals in the real world are capable of traversing in a precarious environment. For instance, an ibex can climb nearly vertical cliffs by controlling and coordinating their dexterous hooves. Those incredible capabilities in nature motivate the realm of legged robotics to mimick the highly dynamic locomotion that animals can achieve. However, controlling the sophisticated dynamic locomotion of robots is a demanding issue because of the nonlinear dynamics and the under-actuation of the floating base that can only be controlled indirectly by the internal motion of the robot and the external wrenches exerted on the robot. This difficulty is further complicated by the constraints such as friction cone constraints that should be imposed on those external wrenches to avoid slip motion. One of the promising approaches that can solve this problem is the model predictive control (MPC) approach that has recently shown remarkable performance.

Successful implementations of whole-body MPC on legged robots [1], [2] have demonstrated the capability of MPC to stabilize a variety of complicated motions on hardware platforms. However, the complex dynamics of legged robots with high degree-of-freedom results in a complicated non-linear problem with constraints and solving this problem in a real-time manner is computationally challenging. To overcome this issue, many researchers tried to approximate the whole-body dynamics with simpler approximated dynamics models. Various examples for approximated dynamics include linear inverted pendulum model [3], single rigid body dynamics model [4], [5], [6], [7], [8], [9], [10], and centroidal dynamics model [11].

Among these examples, single rigid body dynamics provides not only tractable model accuracy but also computational efficiency under the condition that the leg inertia is negligible compared to the total mass of the robot. However, it is not straightforward to properly parameterize the nonlinear orientation dynamics evolving on $\mathrm{SO}(3)$ to perform dynamic maneuvers involving large angular motions or singular configurations such as back-flip and wall-climbing motions using model predictive control. Some researchers adopted local parameterization of $\mathrm{SO}(3)$, such as Euler angles [5], [6], [7], [8]. Despite their intuitive expression of orientation, using Euler angles as parameterization for rotations is not properly invariant under rigid transformations [12], and also known to have singularities in specific configurations. To avoid these issues, a variation-based linearization approach [13] is adopted to linearize the nonlinear orientation dynamics around the operating point [9] or reference trajectory [10] to obtain a singularity-free linear dynamics. However, the linearization depends on the prediction step time being contained within the small range to guarantee the predicted variables being close to the operating point [9]; the locally-valid domain of attraction for the linearization depends on the dynamic system [10].

Therefore, in this study, we address the manifold configuration of the rotation group $\mathrm{SO}(3)$ [12], [14], [15] to properly consider the nonlinear orientation dynamics of the robot. We parameterize the orientation error in the tangent space of $\mathrm{SO}(3)$ manifold. Based on that, a novel NMPC framework is formulated as a constrained nonlinear least squares problem, which is solved by using an efficient algorithm that enables real-time calculation of optimal solutions. Specifically, we derive gradient and Gauss-Newton Hessian approximation of the objective function of the NMPC problem while taking into account the manifold configuration of the rotation group.

The remainder of the paper is structured as follows. Section II describes the dynamics model and the NMPC framework in detail. Section III introduces the derivation of Jacobians required for constructing the gradient and the Gauss-Newton Hessian approximation of the nonlinear objective function. Section IV describes an efficient algorithm for solving the NMPC problem as a constrained nonlinear least squares problem. Section V shows the results of the proposed NMPC applied to various types of legged robots

*This research was supported by the Defense Challengeable Future Technology Program of Agency for Defense Development, Republic of Korea

[1]The authors are with the Humanoid Robot Research Center, School of Mechanical, Aerospace & Systems Engineering, Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Yuseong-gu, Daejeon 34141, Republic of Korea. email: seungwoohong@kaist.ac.kr, kjhpo226@kaist.ac.kr, haewonpark@kaist.ac.kr

in simulation environments. Finally, Section VI briefly discusses the conclusion of the paper.

II. Nonlinear Model Predictive Control

In this section, we formulate a legged locomotion control problem in terms of an NMPC problem. Consider an optimal control problem of a discrete-time deterministic system that consists of states $\mathbf{x}_k \in \mathbb{R}^n$ and control inputs $\mathbf{u}_k \in \mathbb{R}^m$ with a finite time horizon N. This optimal control problem consists of three parts. The first part is a nonlinear model $\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$, $k \in \{0, 1, \cdots, N\}$ that represents the dynamics of the robot.

The second part is the objective function J, which is the sum of weighted deviations of the states and control inputs from a set of desired quantities in a least-squares sense over the entire time horizon considered

$$J = \sum_{k=1}^{N} \frac{1}{2} \|\mathbf{r}_{\mathbf{x}_k}\|^2_{\widetilde{\mathbf{Q}}_k} + \sum_{k=0}^{N-1} \frac{1}{2} \|\mathbf{r}_{\mathbf{u}_k}\|^2_{\widetilde{\mathbf{R}}_k} \tag{1}$$

where $\mathbf{r}_{\mathbf{x}_k} = \mathbf{h}(\mathbf{x}_k)$ is a nonlinear residual error of state, and $\mathbf{r}_{\mathbf{u}_k} = \mathbf{u}_k - \mathbf{u}_k^d$ is a linear residual error of control input at time step k, respectively; $\widetilde{\mathbf{Q}}_k \in \mathbb{S}^n_+$ and $\widetilde{\mathbf{R}}_k \in \mathbb{S}^m_{++}$ are the corresponding weight matrices.

The last part contains constraints on control inputs, and only linear constraints

$$\mathbf{A}_k \mathbf{u}_k \leq \mathbf{b}_k, \mathbf{C}_k \mathbf{u}_k = \mathbf{d}_k \tag{2}$$

are considered in this paper.

Next, we eliminate the state variables from the decision variables by representing them as a function of only $\mathbf{u}$ using model dynamics. For notational convenience, we define $\mathbf{f}_{\mathbf{u}_k}(\mathbf{x}) := \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ and the collection of control inputs and desired control inputs up to the time step $k-1$, $\mathbf{U}_k := [\mathbf{u}_0^T, \cdots, \mathbf{u}_{k-1}^T]^T$ and $\mathbf{U}_k^d := [\mathbf{u}_0^{dT}, \cdots, \mathbf{u}_{k-1}^{dT}]^T$. Then, the state $\mathbf{x}_k$ can be expressed as

$$\mathbf{x}_k = \boldsymbol{\phi}_k(\mathbf{U}_k) = \mathbf{f}_{\mathbf{u}_{k-1}} \circ \mathbf{f}_{\mathbf{u}_{k-2}} \circ \cdots \circ \mathbf{f}_{\mathbf{u}_0}(\mathbf{x}_0) \tag{3}$$

From now on, we will express the collection of any indexed quantities $\mathbf{a}_i$ from index 0 to $k-1$, $[\mathbf{a}_0^T, \cdots, \mathbf{a}_{k-1}^T]^T$, as $\mathbf{A}_k$.

Substituting (3) into the objective function (1) with $\mathbf{h}_k(\mathbf{x}) := \mathbf{h}(\mathbf{x}_k)$ yields

$$J(\mathbf{U}_N) = \sum_{k=1}^{N} \frac{1}{2} \|\mathbf{h}_k(\boldsymbol{\phi}_k(\mathbf{U}_k))\|^2_{\widetilde{\mathbf{Q}}_k} + \sum_{k=0}^{N-1} \frac{1}{2} \|\mathbf{u}_k - \mathbf{u}_k^d\|^2_{\widetilde{\mathbf{R}}_k} \tag{4}$$

With the objective function (4), the optimal control problem becomes

$$\begin{aligned} \min_{\mathbf{U}_N} \quad & \sum_{k=1}^{N} \frac{1}{2} \|\mathbf{h}_k(\boldsymbol{\phi}_k(\mathbf{U}_k))\|^2_{\widetilde{\mathbf{Q}}_k} + \sum_{k=0}^{N-1} \frac{1}{2} \|\mathbf{u}_k - \mathbf{u}_k^d\|^2_{\widetilde{\mathbf{R}}_k} \\ \text{s.t.} \quad & \mathbf{A}_k \mathbf{u}_k \leq \mathbf{b}_k, \mathbf{C}_k \mathbf{u}_k = \mathbf{d}_k \end{aligned} \tag{5}$$

which is in the form of a constrained nonlinear least squares problem.

In general, most of the derivative-based algorithms for solving this constrained nonlinear least squares problem require derivative information of the objective function to find a good search direction. Therefore, we will mainly focus on deriving the associated derivatives in the remaining sections.

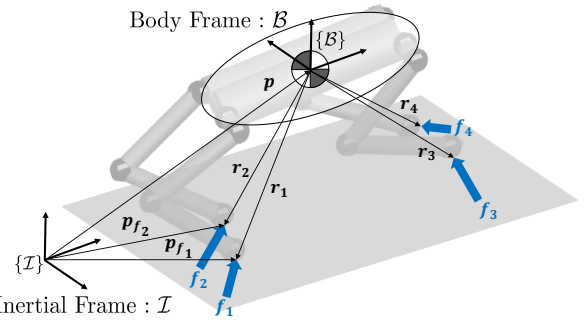

Fig. 1: Illustration of a single rigid body dynamics model for a quadruped robot. Note that the number of legs can be changed to model other types of legged robots.

The gradient of the objective function (4) is given by

$$\mathbf{J} = \sum_{k=1}^{N} \frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N}^T \widetilde{\mathbf{Q}}'_k \mathbf{h}_k(\boldsymbol{\phi}_k(\mathbf{U}_k)) + \boldsymbol{\Phi}_{\widetilde{\mathbf{R}}}(\mathbf{U}_N - \mathbf{U}_N^d) \tag{6}$$

where $\widetilde{\mathbf{Q}}'_k = (\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}})^T \widetilde{\mathbf{Q}}_k$, and $\boldsymbol{\Phi}_{\widetilde{\mathbf{R}}}$ is a block diagonal matrix formed with diagonal elements as $\{\widetilde{\mathbf{R}}_0, \cdots, \widetilde{\mathbf{R}}_{N-1}\}$. Similarly, the Gauss-Newton Hessian approximation of the objective function (4) is given by

$$\mathbf{H}_{GN} = \sum_{k=1}^{N} \frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N}^T \widetilde{\mathbf{Q}}''_k \frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N} + \boldsymbol{\Phi}_{\widetilde{\mathbf{R}}} \tag{7}$$

where $\widetilde{\mathbf{Q}}''_k = (\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}})^T \widetilde{\mathbf{Q}}_k (\frac{\partial \mathbf{h}_k}{\partial \mathbf{x}})$.

The first term of $\mathbf{J}$ and $\mathbf{H}_{GN}$ has $\frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{U}_N} = \begin{bmatrix} \frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{u}_0} & \cdots & \frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{u}_{N-1}} \end{bmatrix} \in \mathbb{R}^{n \times Nm}$ and $\frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{u}_j}$ is given by

$$\frac{\partial \boldsymbol{\phi}_i}{\partial \mathbf{u}_j} = \boldsymbol{\pi}_{i,j} \frac{\partial \mathbf{f}_j}{\partial \mathbf{u}} \tag{8}$$

where,

$$\boldsymbol{\pi}_{i,j} = \begin{cases} \prod_{k=j+1}^{i-1} \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}} & i \geq j+2 \\ \mathbf{I}^{n \times n} & i = j+1 \\ \mathbf{0}^{n \times n} & \text{otherwise} \end{cases} \tag{9}$$

with $\frac{\partial \mathbf{f}_j}{\partial \mathbf{x}} := \frac{\partial \mathbf{f}(\mathbf{x}_j, \mathbf{u}_j)}{\partial \mathbf{x}}$, $\frac{\partial \mathbf{f}_j}{\partial \mathbf{u}} := \frac{\partial \mathbf{f}(\mathbf{x}_j, \mathbf{u}_j)}{\partial \mathbf{u}}$.

The details of formulating the legged locomotion problem as an NMPC framework will be provided in Section II-A, II-B, II-C, II-D, and all the Jacobians required for constructing the gradient and the Gauss-Newton Hessian approximation of the objective function will be introduced in Section III.

A. Model Dynamics

By approximating a legged robot as a single rigid body with point foot, as shown in Figure 1, the dynamics together with kinematics are as follows

$$\dot{\mathbf{p}} = \mathbf{v} \tag{10a}$$

$$\dot{\mathbf{v}} = \frac{1}{m}\sum_i \mathbf{f}_i + \mathbf{g} \tag{10b}$$

$$\dot{\mathbf{R}} = \mathbf{R}\cdot\widehat{\mathbf{w}} \tag{10c}$$

$$\dot{\mathbf{w}} = \mathbf{I}^{-1}(\mathbf{R}^T(\sum_i \mathbf{r}_i \times \mathbf{f}_i) - \mathbf{w}\times(\mathbf{I}\mathbf{w})) \tag{10d}$$

where $\mathbf{p} \in \mathbb{R}^3$ and $\mathbf{v} \in \mathbb{R}^3$ are the position and velocity of body COM, respectively; m is the mass of the robot; $\mathbf{f}_i \in \mathbb{R}^3$ is the ground reaction force (GRF) exerted on the i^{th} contact point; $\mathbf{g} \in \mathbb{R}^3$ is the gravitational acceleration; $\mathbf{R} \in \mathrm{SO}(3)$ is a 3-D rotation matrix, which is an element of Lie group, representing spatial orientation of the body frame $\mathcal{B}$; $\mathbf{I} \in \mathbb{R}^{3\times3}$ and $\mathbf{w} \in \mathbb{R}^3$ are the intertia tensor at the nominal posture and body angular velocity expressed in the body frame $\mathcal{B}$ respectively; $\mathbf{r}_i \in \mathbb{R}^3$ is the vector from the COM to the i^{th} contact point. Note that $\mathbf{p}$, $\mathbf{f}_i$, $\mathbf{g}$, and $\mathbf{r}_i$ are expressed in the inertial frame $\mathcal{I}$. The *hat* operator $\widehat{(\cdot)} : \mathbb{R}^3 \to \mathfrak{so}(3)$ converts elements of 3-D vector into elements of Lie algebra consists of skew-symmetric matrices such that $\widehat{\mathbf{a}}\mathbf{b} = \mathbf{a}\times\mathbf{b}$ for all $\mathbf{a},\mathbf{b} \in \mathbb{R}^3$, where $\times$ is the vector cross product. This dynamic model is not restricted to model only quadrupeds. The number of legs can be changed to model other types of legged robots.

We discretize the continuous-time dynamics (10) using forward Euler integration with sampling time Δt to describe the discrete-time model dynamics as follows

$$\mathbf{R}_{k+1} = \mathbf{R}_k \exp(\widehat{\mathbf{w}}_k \Delta t) \tag{11a}$$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \dot{\mathbf{w}}_k \Delta t \tag{11b}$$

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \mathbf{v}_k \Delta t + \frac{1}{2}\dot{\mathbf{v}}_k \Delta t^2 \tag{11c}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \dot{\mathbf{v}}_k \Delta t \tag{11d}$$

where $\exp : \mathfrak{so}(3) \to \mathrm{SO}(3)$ represents the exponential map around the identity, which coincides with the Rodrigues' formula, transforms elements of Lie algebra to elements of Lie group [14], [15].

The discrete-time state of the system at time step k is now defined as $\mathbf{x}_k := [\mathbf{R}_k, \mathbf{w}_k, \mathbf{p}_k, \mathbf{v}_k] \in \mathcal{X} = \mathrm{SO}(3)\times\mathbb{R}^3\times\mathbb{R}^3\times\mathbb{R}^3$, where we use a rotation matrix, which provides global parameterization of $\mathrm{SO}(3)$, to represent the robot orientation. The discrete-time control input to the system at time step k is defined in terms of GRFs as $\mathbf{u}_k := \left[\mathbf{f}_{1_k}^T, \cdots, \mathbf{f}_{c_k}^T\right]^T \in \mathbb{R}^{3c}$.

B. Objective Function

As the state contains a 3-D rotation matrix which evolves on a $\mathrm{SO}(3)$ manifold, it is difficult to properly define the residual error in a least-squares sense. To tackle this problem, we express the orientation error in terms of the exponential coordinates, which coincides with the tangent space around the identity element of the manifold $\mathrm{SO}(3)$. Thus, we define the residual errors in (1) as $\mathbf{h}(\mathbf{x}_k) := [\mathbf{h}_{\boldsymbol{\varphi}_k}^T, \mathbf{h}_{\mathbf{w}_k}^T, \mathbf{h}_{\mathbf{p}_k}^T, \mathbf{h}_{\mathbf{v}_k}^T]^T \in \mathbb{R}^{12}$ with each term defined as

$$\mathbf{h}_{\boldsymbol{\varphi}_k} = \log(\mathbf{R}_k^{dT}\mathbf{R}_k)^\vee \tag{12a}$$

$$\mathbf{h}_{\mathbf{w}_k} = \mathbf{w}_k - \mathbf{R}_k^T\mathbf{w}_k^d \tag{12b}$$

$$\mathbf{h}_{\mathbf{p}_k} = \mathbf{p}_k - \mathbf{p}_k^d \tag{12c}$$

$$\mathbf{h}_{\mathbf{v}_k} = \mathbf{v}_k - \mathbf{v}_k^d \tag{12d}$$

where $\mathbf{R}_k^d \in \mathrm{SO}(3)$, and $\mathbf{w}_k^d \in \mathbb{R}^3$, $\mathbf{p}_k^d \in \mathbb{R}^3$, $\mathbf{v}_k^d \in \mathbb{R}^3$ are the corresponding desired motions represented in the inertial frame $\mathcal{I}$. The operator $\log : \mathrm{SO}(3) \to \mathfrak{so}(3)$ represents the logarithm map [14], [15], which is the inverse of exponential map. The *vee* operator $(\cdot)^\vee : \mathfrak{so}(3) \to \mathbb{R}^3$ is the inverse of *hat* operator.

In this work, no desired control input is provided to the robot, thereby $\mathbf{u}_k^d = \mathbf{0}$ for all k.

C. Foot Reference Generation

We use a foot placement strategy for the swing foot similar to the one introduced in [16], [6],

$$\mathbf{p}_{f_i} = \mathbf{p}_{f_i}^{ff} + \mathbf{p}_{f_i}^{fb} = \mathbf{p}_{h_i} + \frac{1}{2}\mathbf{v}^d T_{st} + \sqrt{\frac{p_z}{g}}(\mathbf{v} - \mathbf{v}^d) \tag{13}$$

where $\mathbf{p}_{f_i}$ and $\mathbf{p}_{h_i}$ are the touch down location of i^{th} swing foot and the position of i^{th} hip expressed in the inertial frame $\mathcal{I}$, respectively; T_{st} is the stance time; p_z is the height of the body COM.

D. Constraints

To prevent the foot slip motion occured as well as to avoid the GRF generating excessive or negative force, we impose the linearized friction cone and box constraints on i^{th} contact foot, i.e., $\forall i \in \{\text{Stance Phase}\}$

$$|f_{i_x}| \le \mu f_{i_z}, \quad |f_{i_y}| \le \mu f_{i_z}, \quad f_{min} \le f_{i_z} \le f_{max} \tag{14}$$

where μ is the friction coefficient given as 0.6 in this work. In order to make the robot comply with the desired gait sequence, we impose all feet forces in swing phase to zero

$$\mathbf{f}_i = \mathbf{0},\ \forall i \in \{\text{Swing Phase}\} \tag{15}$$

III. Derivation of Analytical Jacobians

In this section, we will mainly focus on deriving the Jacobians for calculating the gradient and Gauss-Newton Hessian approximation of the nonlinear objective function in (5). They will be used in the optimization algorithm in Section IV to find a good search direction of optimal solution.

In general, the derivative-based algorithms iteratively finds the minimizer of the quadratic objective function, which is expressed in terms of state deviation $\delta\mathbf{x}_k$ and control deviation $\delta\mathbf{u}_k$, to update the current iterate $\overline{\mathbf{x}}_k$ and $\overline{\mathbf{u}}_k$. For the state variable $\mathbf{R}_k \in \mathrm{SO}(3)$, the exponential map is selected as the retraction on a manifold [17], i.e., $\mathbf{R}_k = \overline{\mathbf{R}}_k \exp(\widehat{\delta\boldsymbol{\varphi}_k})$ with $\delta\boldsymbol{\varphi}_k \in \mathbb{R}^3$. Throughout, variables with overline represent the nominal state at the associated time step. For the other state variables $\mathbf{p}_k, \mathbf{w}_k, \mathbf{v}_k$ living in the vector space $\mathbb{R}^3$, their corresponding deviations also belong to the vector space

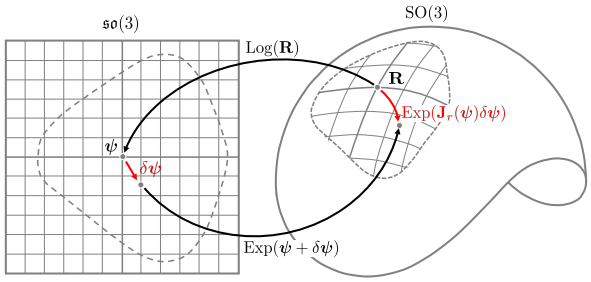

Fig. 2: Illustration of the relation between the manifold $\mathrm{SO}(3)$ and the tangent space $\mathfrak{so}(3)$. The right-Jacobian $\mathbf{J}_r$ relates the variations in the tangent space to the variations on the $\mathrm{SO}(3)$ manifold.

$\mathbb{R}^3$. Thus, we can define the state deviation at time step k as $\delta\mathbf{x}_k := [\delta\boldsymbol{\varphi}_k^T, \delta\mathbf{w}_k^T, \delta\mathbf{p}_k^T, \delta\mathbf{v}_k^T]^T \in \mathbb{R}^{12}$. Similarly, the deviation of control input at time step k can be defined as $\delta\mathbf{u}_k := [\delta\mathbf{f}_{1_k}^T, \cdots, \delta\mathbf{f}_{c_k}^T]^T \in \mathbb{R}^{3c}$.

We now consider the residual error $\mathbf{h} : \mathcal{X} \to \mathbb{R}^{12}$ in (12). A standard approach to find the Jacobian of $\mathbf{h}(\mathbf{x}_k)$ is to introduce a basis for $\mathcal{X}$, find the Jacobian of the corresponding function, and substitute the result back into $\mathcal{X}$. Instead, we will directly derive the first-order approximation of $\mathbf{h}$ at $\overline{\mathbf{x}}_k$.

In the process of derivation, we will use the first-order approximation for the exponential and logarithm adopted from [12], [18],

$$\mathrm{Exp}(\boldsymbol{\psi} + \delta\boldsymbol{\psi}) \approx \mathrm{Exp}(\boldsymbol{\psi})\mathrm{Exp}(\mathbf{J}_r(\boldsymbol{\psi})\delta\boldsymbol{\psi}) \tag{16}$$

$$\mathrm{Log}\left(\mathrm{Exp}(\boldsymbol{\psi})\mathrm{Exp}(\delta\boldsymbol{\psi})\right) \approx \boldsymbol{\psi} + \mathbf{J}_r^{-1}(\boldsymbol{\psi})\delta\boldsymbol{\psi} \tag{17}$$

where $\mathbf{J}_r(\cdot)$ and $\mathbf{J}_r^{-1}(\cdot)$ are the right-Jacobian and inverse of right-Jacobian for exponential coordinates [15], and as shown in Figure 2, they relate the variations between the tangent space and the $\mathrm{SO}(3)$ [12]; $\mathrm{Exp} : \mathbb{R}^3 \to \mathrm{SO}(3)$ and $\mathrm{Log} : \mathrm{SO}(3) \to \mathbb{R}^3$ are compact notations for $\exp(\cdot)$ and $\log(\cdot)$ used in [12], we will also use those compact notations for readability.

Let $\mathbf{x}_k \in \mathcal{X}$ be close to the nominal state $\overline{\mathbf{x}}_k$, and let $\delta\mathbf{x}_k$ assumed to be small. Similarly, let $\mathbf{u}_k \in \mathbb{R}^{3c}$ be close to $\overline{\mathbf{u}}_k$ with $\delta\mathbf{u}_k$ assumed to be small. We have

$$\mathbf{R}_k = \overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k) \tag{18a}$$

$$\mathbf{w}_k = \overline{\mathbf{w}}_k + \delta\mathbf{w}_k \tag{18b}$$

$$\mathbf{p}_k = \overline{\mathbf{p}}_k + \delta\mathbf{p}_k \tag{18c}$$

$$\mathbf{v}_k = \overline{\mathbf{v}}_k + \delta\mathbf{v}_k \tag{18d}$$

$$\mathbf{u}_k = \overline{\mathbf{u}}_k + \delta\mathbf{u}_k \tag{18e}$$

Now, with the above preliminaries, we will derive the Jacobians of $\mathbf{h}(\mathbf{x}_k)$ and $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ in the remaining section.

1) Jacobians of $\mathbf{h}(\mathbf{x}_k)$*:* First, Jacobians of $\mathbf{h}_{\boldsymbol{\varphi}_k}$, $\mathbf{h}_{\mathbf{w}_k}$, $\mathbf{h}_{\mathbf{p}_k}$ and $\mathbf{h}_{\mathbf{v}_k}$ are derived as below.

a) Jacobians of $\mathbf{h}_{\boldsymbol{\varphi}_k}$*:* Since $\mathbf{h}_{\boldsymbol{\varphi}_k}$ is a function of only $\mathbf{R}_k$, thereby the partial derivatives with respect to $\delta\mathbf{w}_k$, $\delta\mathbf{p}_k$, and $\delta\mathbf{v}_k$ are zero matrices. The first-order approximation of $\mathbf{h}_{\boldsymbol{\varphi}_k}$ at $\overline{\mathbf{R}}_k$ is

$$\begin{aligned}
&\mathbf{h}_{\boldsymbol{\varphi}_k}\left(\overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k)\right) \\
&\quad= \mathrm{Log}\left({\mathbf{R}_k^d}^T\overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k)\right) \\
&\quad\overset{(17)}{\approx} {\mathbf{R}_k^d}^T\overline{\mathbf{R}}_k + \mathbf{J}_r^{-1}({\mathbf{R}_k^d}^T\overline{\mathbf{R}}_k)\delta\boldsymbol{\phi}_k
\end{aligned} \tag{19}$$

b) Jacobians of $\mathbf{h}_{\mathbf{w}_k}$*:* Because $\mathbf{h}_{\mathbf{w}_k}$ is linear in $\delta\mathbf{w}_k$, thereby the partial derivative with respect to $\delta\mathbf{w}_k$ is an identity matrix. In addition, $\mathbf{h}_{\mathbf{w}_k}$ is independent of $\mathbf{p}_k$ and $\mathbf{v}_k$, thereby the partial derivatives with respect to $\delta\mathbf{p}_k$ and $\delta\mathbf{v}_k$ are zero matrices. The first-order approximation of $\mathbf{h}_{\mathbf{w}_k}$ at $\overline{\mathbf{R}}_k$ is

$$\begin{aligned}
&\mathbf{h}_{\mathbf{w}_k}\left(\overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k)\right) \\
&\quad= \overline{\mathbf{w}}_k - \overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k)^T\mathbf{w}_k^d \\
&\quad= \overline{\mathbf{w}}_k - \mathrm{Exp}(-\delta\boldsymbol{\varphi}_k)\overline{\mathbf{R}}_k^T\mathbf{w}_k^d \\
&\quad\overset{(a)}{\approx} \overline{\mathbf{w}}_k - (\mathbb{I} - \widehat{\delta\boldsymbol{\varphi}_k})\overline{\mathbf{R}}_k^T\mathbf{w}_k^d \\
&\quad\overset{(b)}{\approx} \overline{\mathbf{w}}_k - \overline{\mathbf{R}}_k^T\mathbf{w}_k^d + \widehat{(\overline{\mathbf{R}}_k^T\mathbf{w}_k^d)}^T\delta\boldsymbol{\varphi}_k
\end{aligned} \tag{20}$$

where (a) we have used the approximation of exponential map up to first-order term $\mathrm{Exp}(\delta\boldsymbol{\varphi}_k) \approx \mathbb{I} + \widehat{\delta\boldsymbol{\varphi}_k}$ with $\mathbb{I} \in \mathbb{R}^{3\times3}$ representing an identity matrix; (b) we have used the property $\widehat{\mathbf{a}}\mathbf{b} = -\widehat{\mathbf{b}}\mathbf{a}$.

c) Jacobians of $\mathbf{h}_{\mathbf{p}_k}$ *and* $\mathbf{h}_{\mathbf{v}_k}$*:* It is straightforward that $\mathbf{h}_{\mathbf{p}_k}$ and $\mathbf{h}_{\mathbf{v}_k}$ are linear in $\delta\mathbf{p}_k$ and $\delta\mathbf{v}_k$ respectively, thereby the corresponding partial derivatives with respect to $\delta\mathbf{p}_k$ and $\delta\mathbf{v}_k$ are identity matrices, while the other partial derivatives are zero matrices.

As a result, the Jacobian of $\mathbf{h}(\mathbf{x}_k)$ is given by

$$\frac{\partial\mathbf{h}_k}{\partial\mathbf{x}} = \begin{bmatrix} \mathbf{J}_r^{-1}({\mathbf{R}_k^d}^T\overline{\mathbf{R}}_k) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \widehat{(\overline{\mathbf{R}}_k^T\mathbf{w}_k^d)}^T & \mathbb{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbb{I} \end{bmatrix} \tag{21}$$

2) Jacobians of $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$*:* Next, Jacobians of $\mathbf{f}_{\boldsymbol{\varphi}_k}$, $\mathbf{f}_{\mathbf{w}_k}$, $\mathbf{f}_{\mathbf{p}_k}$ and $\mathbf{f}_{\mathbf{v}_k}$ are derived as below.

a) Jacobians of $\mathbf{f}_{\boldsymbol{\varphi}_k}$*:* We first note that $\mathbf{f}_{\boldsymbol{\varphi}_k}$ is independent of $\mathbf{p}_k$, $\mathbf{v}_k$, $\mathbf{u}_k$, thereby the partial derivatives with respect to $\delta\mathbf{p}_k$, $\delta\mathbf{v}_k$, $\delta\mathbf{u}_k$ are zero matrices. The remaning Jacobians with respect to $\delta\boldsymbol{\varphi}_k$ and $\delta\mathbf{w}_k$ will be derived. Substituting (18a) at step time $k+1$ and k respectively into (11a) and rearranging the equation, we get

$$\begin{aligned}
&\mathrm{Exp}(\delta\boldsymbol{\varphi}_{k+1}) \\
&\quad\approx \overline{\mathbf{R}}_{k+1}^T\overline{\mathbf{R}}_k\mathrm{Exp}(\delta\boldsymbol{\varphi}_k)\mathrm{Exp}(\overline{\mathbf{w}}_k\Delta t) \\
&\quad\overset{(a)}{=} \overline{\mathbf{R}}_{k+1}^T\overline{\mathbf{R}}_k\mathrm{Exp}(\overline{\mathbf{w}}_k\Delta t)\mathrm{Exp}(\mathrm{Exp}^T(\overline{\mathbf{w}}_k\Delta t)\delta\boldsymbol{\varphi}_k) \\
&\quad\overset{(b)}{=} \mathrm{Exp}(\mathrm{Exp}^T(\overline{\mathbf{w}}_k\Delta t)\delta\boldsymbol{\varphi}_k)
\end{aligned} \tag{22}$$

where (a) we have used the property $\mathrm{Exp}(\boldsymbol{\varphi})\mathbf{R} = \mathbf{R}\mathrm{Exp}(\mathbf{R}^T\boldsymbol{\varphi})$; (b) is because the first two terms cancel each

other by the fact $\overline{\mathbf{R}}_{k+1} = \overline{\mathbf{R}}_k \mathrm{Exp}(\overline{\mathbf{w}}_k \Delta t)$. Taking logarithm map on both sides of (22) yields

$$\delta\boldsymbol{\varphi}_{k+1} \approx \mathrm{Exp}^T(\overline{\mathbf{w}}_k \Delta t)\delta\boldsymbol{\varphi}_k \tag{23}$$

Similarly, plugging (18a) at step time $k+1$ and (18b) at step time k into (11a) and rearranging the equation, we have

$$\begin{aligned}
&\mathrm{Exp}(\delta\boldsymbol{\varphi}_{k+1}) \\
&\quad = \overline{\mathbf{R}}_{k+1}^T \overline{\mathbf{R}}_k \mathrm{Exp}((\overline{\mathbf{w}}_k + \delta\mathbf{w}_k)\Delta t) \\
&\quad \overset{(16)}{\approx} \overline{\mathbf{R}}_{k+1}^T \overline{\mathbf{R}}_k \mathrm{Exp}(\overline{\mathbf{w}}_k \Delta t)\mathrm{Exp}(\mathbf{J}_r(\overline{\mathbf{w}}_k \Delta t)\Delta t \delta\mathbf{w}_k) \\
&\quad \overset{(a)}{=} \mathrm{Exp}(\mathbf{J}_r(\overline{\mathbf{w}}_k \Delta t)\Delta t \delta\mathbf{w}_k)
\end{aligned} \tag{24}$$

where (a) we have used the fact $\overline{\mathbf{R}}_{k+1} = \overline{\mathbf{R}}_k \mathrm{Exp}(\overline{\mathbf{w}}_k \Delta t)$. Taking logarithm map on both sides of (24) yields

$$\delta\boldsymbol{\varphi}_{k+1} \approx \mathbf{J}_r(\overline{\mathbf{w}}_k \Delta t)\Delta t \delta\mathbf{w}_k \tag{25}$$

b) Jacobians of $\mathbf{f}_{\mathbf{w}_k}$*:* Similar to the previous section, $\mathbf{f}_{\mathbf{w}_k}$ is independent of $\mathbf{p}_k$ and $\mathbf{v}_k$, thereby the partial derivatives with respect to $\delta\mathbf{p}_k$ and $\delta\mathbf{v}_k$ are zero matrices. We will focus on the remaning Jacobians with respect to $\delta\boldsymbol{\varphi}_k$, $\delta\mathbf{w}_k$ and $\delta\mathbf{u}_k$. Substituting (18b) at step time $k+1$ and (18a) at step time k into (11b) and rearranging the equation up to first-order terms

$$\delta\mathbf{w}_{k+1} \approx -\mathbf{I}^{-1}\left[\widehat{\delta\boldsymbol{\varphi}_k}\overline{\mathbf{R}}_k^T[\mathbf{r}_k]\overline{\mathbf{u}}_k\right]\Delta t \tag{26}$$

$$= \mathbf{I}^{-1}\left[\widehat{\overline{\mathbf{R}}_k^T[\mathbf{r}_k]\overline{\mathbf{u}}_k}\right]\Delta t\delta\boldsymbol{\varphi}_k \tag{27}$$

where $[\mathbf{r}_k] := [\widehat{\mathbf{r}_{1_k}}, \cdots, \widehat{\mathbf{r}_{c_k}}] \in \mathbb{R}^{3\times 3c}$.

Similarly, plugging (18b) at step time $k+1$ and at step time k respectively into (11b) and rearranging the equation up to first-order terms, we have

$$\begin{aligned}
\delta\mathbf{w}_{k+1} &\approx \delta\mathbf{w}_k - \mathbf{I}^{-1}\left[\widehat{\overline{\mathbf{w}}}_k(\mathbf{I}\delta\mathbf{w}_k) + \widehat{\delta\mathbf{w}}_k(\mathbf{I}\overline{\mathbf{w}}_k)\right]\Delta t \\
&= \left(\mathbb{I} - \mathbf{I}^{-1}\left[\widehat{\overline{\mathbf{w}}}_k\mathbf{I} - (\widehat{\mathbf{I}\overline{\mathbf{w}}_k})\right]\Delta t\right)\delta\mathbf{w}_k
\end{aligned} \tag{28}$$

Plugging (18b) at step time $k+1$ and (18e) at step time k into (11b) and rearranging the equation up to first-order terms

$$\delta\mathbf{w}_{k+1} \approx \mathbf{I}^{-1}\left(\overline{\mathbf{R}}_k^T[\mathbf{r}_k]\right)\Delta t\delta\mathbf{u}_k \tag{29}$$

c) Jacobians of $\mathbf{f}_{\mathbf{p}_k}$ *and* $\mathbf{f}_{\mathbf{v}_k}$*:* Repeating application of the first-order approximation on (12c) and (12d), we get

$$\delta\mathbf{p}_{k+1} \approx \Delta t\delta\mathbf{v}_k \tag{30a}$$

$$\delta\mathbf{p}_{k+1} \approx (\frac{\Delta t^2}{2m}[\mathbb{I}])\delta\mathbf{u}_k \tag{30b}$$

$$\delta\mathbf{v}_{k+1} \approx (\frac{\Delta t}{m}[\mathbb{I}])\delta\mathbf{u}_k \tag{30c}$$

where $[\mathbb{I}] := [\mathbb{I}, \cdots, \mathbb{I}] \in \mathbb{R}^{3\times 3c}$ with $\mathbb{I} \in \mathbb{R}^{3\times 3}$ representing an identity matrix.

In summary, the Jacobians of $\mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ are

$$\frac{\partial \mathbf{f}_k}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial\delta\boldsymbol{\varphi}_{k+1}}{\partial\delta\boldsymbol{\varphi}_k} & \frac{\partial\delta\boldsymbol{\varphi}_{k+1}}{\partial\delta\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\delta\mathbf{w}_{k+1}}{\partial\delta\boldsymbol{\varphi}_k} & \frac{\partial\delta\mathbf{w}_{k+1}}{\partial\delta\mathbf{w}_k} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{I} & \mathbb{I}\Delta t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbb{I} \end{bmatrix} \tag{31}$$

with each partial derivatives specified as $\frac{\partial\delta\boldsymbol{\varphi}_{k+1}}{\partial\delta\boldsymbol{\varphi}_k} = \mathrm{Exp}^T(\overline{\mathbf{w}}_k\Delta t)$, $\frac{\partial\delta\boldsymbol{\varphi}_{k+1}}{\partial\delta\mathbf{w}_k} = \mathbf{J}_r(\overline{\mathbf{w}}_k\Delta t)\Delta t$, $\frac{\partial\delta\mathbf{w}_{k+1}}{\partial\delta\boldsymbol{\varphi}_k} = \mathbf{I}^{-1}[\widehat{\overline{\mathbf{R}}_k^T[\mathbf{r}_k]\overline{\mathbf{u}}_k}]\Delta t$, $\frac{\partial\delta\mathbf{w}_{k+1}}{\partial\delta\mathbf{w}_k} = (\mathbb{I} - \mathbf{I}^{-1}[\widehat{\overline{\mathbf{w}}}_k\mathbf{I} - (\widehat{\mathbf{I}\overline{\mathbf{w}}_k})]\Delta t)$, and

$$\frac{\partial \mathbf{f}_k}{\partial \mathbf{u}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I}^{-1}\left(\overline{\mathbf{R}}_k^T[\mathbf{r}_k]\right)\Delta t \\ \frac{1}{2}(\frac{1}{m}[\mathbb{I}]\Delta t^2) \\ \left(\frac{1}{m}[\mathbb{I}]\Delta t\right) \end{bmatrix} \tag{32}$$

IV. Proximal Gauss-Newton Method

This section briefly summarizes an optimization algorithm to solve for constrained nonlinear least squares problem introduced in Section II. We combine Gauss-Newton method for nonlinear least squares problem with a proximal operator to handle inequality constraints. The gradient and Gauss-Newton Hessian approximation obtained in Section II and III are utilized in this optimization algorithm.

A. Proximal Operator

Proximal methods minimize an objective function in the form of $g + f$ with a differentiable function g and a non-differentiable function f [19]. To handle the non-differentiable part of the objective function f, a proximal operator of f at $\mathbf{v}$ is introduced,

$$\mathbf{prox}_f(\mathbf{v}) = \arg\min_{\mathbf{z}}\left(f(\mathbf{z}) + \frac{1}{2}\|\mathbf{z}-\mathbf{v}\|^2\right). \tag{33}$$

When f is the indicator function of the set $\mathcal{C}$, *i.e.,* $I_{\mathcal{C}}(\mathbf{v}) = 0$ if $\mathbf{v} \in \mathcal{C}$ and $I_{\mathcal{C}}(\mathbf{v}) = +\infty$ if $\mathbf{v} \notin \mathcal{C}$ which is convex for a closed nonempty convex set $\mathcal{C}$, the output value of the proximal operator (33) at $\mathbf{v}$ is equivalent to the solution of the following optimization problem,

$$\min_{\mathbf{z}\in\mathcal{C}} \quad \frac{1}{2}\|\mathbf{z}-\mathbf{v}\|^2 \tag{34}$$

Geometrically, when $\mathbf{v} \notin \mathcal{C}$, the proximal operator maps the point $\mathbf{v}$ to the closest point in the set $\mathcal{C}$, whereas the output of the operator is the same as $\mathbf{v}$ when $\mathbf{v} \in \mathcal{C}$. This proximal operator can be utilized to modify search directions in the line search algorithms to optimize composite functions with non-differentiable part f. For example, proximal gradient descent algorithm for the Lasso regression was discussed in [20] which modifies search directions obtained from the gradient of the continuous part of the objective function using the proximal operator. Also, a proximal Newton-type method is introduced in [21] to modify the Newton step obtained from the Hessian and gradient of the differentiable part of the objective function g. In this method, a scaled proximal operator is introduced,

$$\mathbf{prox}_f^{\mathbf{P}}(\mathbf{v}) = \arg\min_{\mathbf{z}}\left(f(\mathbf{z}) + \frac{1}{2}\|\mathbf{z}-\mathbf{v}\|_{\mathbf{P}}^2\right). \tag{35}$$

where, the positive definite matrix $\mathbf{P}$ is chosen as the Hessian of the differentiable part of the objective function $\frac{\partial^2 g}{\partial \mathbf{v}^2}$. Compared to the proximal gradient descent algorithm,

the proximal Newton-type algorithm showed an improved convergence rate.

If the set $\mathcal{C}$ is in the form of $\{\mathbf{v} \,|\mathbf{A}\mathbf{v} \leq \mathbf{b}, \mathbf{C}\mathbf{v} = \mathbf{d}\,\}$, then the value of the scaled proximal operator in (35) can be obtained by solving the following Quadratic Program (QP),

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\|\mathbf{z}-\mathbf{v}\|^2_{\mathbf{H}} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{z} \leq \mathbf{b}, \mathbf{C}\mathbf{z} = \mathbf{d}, \end{aligned} \tag{36}$$

and many state-of-the-art solvers are available to solve this problem in a very efficient manner.

B. Proximal Gauss-Newton Algorithm

The Gauss-Newton algorithm solves nonlinear least squares problems $\min_{\mathbf{v}} J(\mathbf{v}) = \frac{1}{2}\sum_i \|\mathbf{r}_i(\mathbf{v})\|^2_{\mathbf{Q}_i}$. Unlike the Newton's algorithm using a Hessian, the Gauss-Newton algorithm uses Gauss-Newton Hessian approximation, $\mathbf{H}_{GN} := \sum_i \frac{\partial \mathbf{r}_i(\mathbf{v})}{\partial \mathbf{v}}^T \mathbf{Q}_i \frac{\partial \mathbf{r}_i(\mathbf{v})}{\partial \mathbf{v}}$, to obtain line search directions.

This Gauss-Newton Hessian approximation has desirable properties such as guaranteed positive semidefiteness and low computational complexity due to ignorance of the second-order derivative of $\mathbf{r}_i$. In this paper, the Gauess-Newton algorithm will be extended to handle following constrained nonlinear least squares problems,

$$\begin{aligned} \min_{\mathbf{v}} \quad & \frac{1}{2}\sum_i \|\mathbf{r}_i(\mathbf{v})\|^2_{\mathbf{Q}_i} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{v} \leq \mathbf{b}, \mathbf{C}\mathbf{v} = \mathbf{d} \end{aligned} \tag{37}$$

using the scaled proximal operator.

First, the constrained nonlinear least squares problem (37) is converted into the equivalent unconstrained optimization problem with a composite objective function,

$$\min_{\mathbf{v}} \quad g(\mathbf{v}) + I_{\mathcal{C}}(\mathbf{v}) \tag{38}$$

where $g(\mathbf{v}) = \frac{1}{2}\sum_i \|\mathbf{r}_i(\mathbf{v})\|^2_{\mathbf{Q}_i}$ and $\mathcal{C} = \{\mathbf{A}\mathbf{v} \leq \mathbf{b}, \mathbf{C}\mathbf{v} = \mathbf{d}\}$. A search direction can be obtained using the Gauss-Newton Hessian approximation and gradient of the continuous part of the objective function $\delta\mathbf{v} = -\mathbf{H}^{-1}_{GN}\mathbf{J}$, where $\mathbf{J}$ is the gradient of the objective function given by $\sum_i \frac{\partial \mathbf{r}_i(\mathbf{v})}{\partial \mathbf{v}}^T \mathbf{Q}_i \mathbf{r}_i(\mathbf{v})$.

To handle the non-differentiable part of the objective function $I_{\mathcal{C}}(\mathbf{v})$, this search direction is modified using the scaled proximal operator in (35),

$$\delta\mathbf{v} = \mathbf{prox}^{\mathbf{H}_{GN}}_{I_{\mathcal{C}}}\left(\mathbf{v}^{(k)} - \delta\mathbf{v}\right) - \mathbf{v}^{(k)} \tag{39}$$

where, $\mathbf{v}^{(k)}$ is the current iterate.

With this modified search direction, the proximal Gauss-Newton algorithm is shown in Algorithm 1.

In Algorithm 1, $t^{(k)}$ is the step length to determine the amount to move along the modified search direction and its values are obtained using a backtracking line search algorithm based on the Armijo condition. The detailed description of the backtracking line search algorithm can be found in [21] and omitted here due to the limitation of space.

Algorithm 1 Proximal Gauss-Newton Algorithm

Require: $\min \frac{1}{2}\sum_i \|\mathbf{r}_i(\mathbf{v})\|^2_{\mathbf{Q}_i}$ s.t.$\mathbf{A}\mathbf{v} \leq \mathbf{b}, \mathbf{C}\mathbf{v} = \mathbf{d}$
1: Choose MAX_ITER $\gg 0$ and $0 < \epsilon \ll 1$
2: Initialize $\mathbf{v}$ with some $\mathbf{v}^{(0)}$
3: **while** 1 **do**
4: Obtain $\delta\mathbf{v}$ using (39)
5: Obtain $t^{(k)}$ using backtracking line search
6: $\mathbf{v}^{(k+1)} = \mathbf{v}^{(k)} + t^{(k)}\delta\mathbf{v}$
7: **if** $i \geq$ MAX_ITER or $\|\mathbf{v}^{(k)} - \mathbf{v}^{k+1}\| \leq \epsilon$ **then**
8: **break**
9: **end if**
10: $k = k + 1$
11: **end while**

C. Efficient Calculation of Search Direction and Proximal Operator

With the derived Jacobians in Section III, $\mathbf{H}_{GN}$ and $\mathbf{J}$ can be calculated analytically using equations (6) and (7). Utilizing these matrices, search direction $\delta\mathbf{U}_N = -\mathbf{H}^{-1}_{GN}\mathbf{J}$ can be obtained and proximal operator can be evaluated using (39). However, the construction of the matrix $\mathbf{H}_{GN}$ is computationally challenging for real-time implementation in the case of a complex nonlinear MPC problem. Furthermore, because $\mathbf{H}_{GN}$ is a dense matrix, solve time of QP for proximal operator will increase quickly as the size of $\mathbf{H}_{GN}$ increases. Hence, we propose a method to obtain the search direction and the value of proximal operator while avoiding the construction of $\mathbf{H}_{GN}$.

First, consider $\mathbf{y}_k := \frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N}\delta\mathbf{U}_N$, then $\mathbf{y}_k$ can be obtained through the following recursive relation,

$$\mathbf{y}_i = \frac{\partial \mathbf{f}_{i-1}}{\partial \mathbf{x}}\mathbf{y}_{i-1} + \frac{\partial \mathbf{f}_{i-1}}{\partial \mathbf{u}}\delta\mathbf{u}_{i-1}, \mathbf{y}_0 = 0, i = 1, \cdots, k. \tag{40}$$

Also, we can easily identify that search direction $\delta\mathbf{U}_N = -\mathbf{H}^{-1}_{GN}\mathbf{J}$ is the solution of the following optimization problem,

$$\min_{\delta\mathbf{U}_N} \quad \frac{1}{2}\delta\mathbf{U}_N^T\mathbf{H}_{GN}\delta\mathbf{U}_N + \delta\mathbf{U}_N{}^T\mathbf{J} \tag{41}$$

by the first-order optimality condition for a positive definite $\mathbf{H}_{GN}$.We introduce auxiliary variables $\mathbf{y}_k$ into (41) to replace $\frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N}\delta\mathbf{U}_N$ in (41) with $\mathbf{y}_k$ and recursive relations in (40). With this procedure, an equivalent optimization problem can be obtained as follows,

$$\begin{aligned} \min_{\delta\mathbf{u}_k, \mathbf{y}_k} \quad & \sum_{k=1}^{N} \frac{1}{2}\alpha_k(\mathbf{y}_k) + \beta_k(\mathbf{y}_k) \\ & + \sum_{k=0}^{N-1} \frac{1}{2}\gamma_k(\delta\mathbf{u}_k) + \xi_k(\delta\mathbf{u}_k) \\ \text{s.t.} \quad & \mathbf{y}_{k+1} = \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\mathbf{y}_k + \frac{\partial \mathbf{f}_k}{\partial \mathbf{u}}\delta\mathbf{u}_k, \ \mathbf{y}_0 = \mathbf{0} \end{aligned} \tag{42}$$

with $\alpha_k(\mathbf{y}_k) = \|\mathbf{y}_k\|^2_{\widetilde{\mathbf{Q}}_k''}$, $\beta_k(\mathbf{y}_k) = \mathbf{y}_k^T\widetilde{\mathbf{Q}}_k'\mathbf{h}_k(\boldsymbol{\phi}_k(\mathbf{U}_k))$, $\gamma_k(\delta\mathbf{u}_k) = \|\delta\mathbf{u}_k\|^2_{\widetilde{\mathbf{R}}_k}$, and $\xi_k(\delta\mathbf{u}_k) = \delta\mathbf{u}_k^T\widetilde{\mathbf{R}}_k(\mathbf{u}_k - \mathbf{u}_k^d)$. For

this optimization problem, a sparse QP formulation can be easily obtained where the sparsity pattern can be efficiently utilized to significantly decrease solve time.

Now, we formulate a sparse QP formulation with a solution equivalent to the $\mathbf{prox}_{I_C}^{\mathbf{H}_{GN}}(\mathbf{U}_N - \delta\mathbf{U}_N)$ in (39). Let $\tilde{\mathbf{U}}_N := \mathbf{U}_N - \delta\mathbf{U}_N$, then the value of the scaled proximal operator can be obtained by solving the following dense QP,

$$\begin{aligned} \min_{\mathbf{z}_k} \quad & \frac{1}{2}\|\mathbf{Z}_n - \tilde{\mathbf{U}}_N\|^2_{\mathbf{H}_{GN}} \\ & = \frac{1}{2}(\mathbf{Z}_n - \tilde{\mathbf{U}}_N)^T\mathbf{H}_{GN}(\mathbf{Z}_n - \tilde{\mathbf{U}}_N) \\ \text{s.t.} \quad & \mathbf{A}_k\mathbf{z}_k \le \mathbf{b}_k, k = 0, \cdots, N-1, \\ & \mathbf{C}_k\mathbf{z}_k = \mathbf{d}_k, k = 0, \cdots, N-1. \end{aligned} \tag{43}$$

Likewise, $\frac{\partial \boldsymbol{\phi}_k}{\partial \mathbf{U}_N}(\mathbf{Z}_n - \tilde{\mathbf{U}}_N)$ in the objective function can be replaced with the recursive relation in (40) by substituting $\delta\mathbf{U}_N$ with $\mathbf{Z}_n - \tilde{\mathbf{U}}_N$ and the introduction of auxiliary variables $\mathbf{y}_k$. Through this process, following equivalent sparse QP formulation is obtained,

$$\begin{aligned} \min_{\mathbf{z}_k, \mathbf{y}_k} \quad & \sum_{k=1}^{N}\frac{1}{2}\|\mathbf{y}_k\|^2_{\tilde{\mathbf{Q}}_k''} + \sum_{k=0}^{N-1}\frac{1}{2}\|\mathbf{z}_k - \tilde{\mathbf{u}}_k\|^2_{\tilde{\mathbf{R}}_k} \\ \text{s.t.} \quad & \mathbf{y}_{k+1} = \frac{\partial \mathbf{f}_k}{\partial \mathbf{x}}\mathbf{y}_k + \frac{\partial \mathbf{f}_k}{\partial \mathbf{u}}(\mathbf{z}_k - \tilde{\mathbf{u}}_k),\ \mathbf{y}_0 = \mathbf{0}, \\ & \mathbf{A}_k\mathbf{z}_k \le \mathbf{b}_k, \mathbf{C}_k\mathbf{z}_k = \mathbf{d}_k. \end{aligned} \tag{44}$$

With the proper choice of QP solver, sparsity patterns in (42) and (44) can be efficiently utilized. We chose *qpSWIFT* proposed in [22] as a QP solver since it is specifically designed to utilize sparsity patterns of QP formulation.

V. Simulation Results

The proposed NMPC framework with the proximal Gauss-Newton algorithm will be evaluated in this section.

A. Simulation Setup

The NMPC framework as well as the efficient proximal Gauss-Newton Algorithm proposed in this work were tested in MATLAB simulation. The simulation was implemented on a standard desktop running with Intel® i7-8700 CPU 3.2 GHz processors. For all the simulation experiments represented in this work, we used a horizon length of $N = 12$ with a sampling time $\Delta t = 0.025$ sec. At each step time, we find the optimal control inputs by using Algorithm 1 to update the current iterate. Then we update the state using the continuous-time dynamics (10) via MATLAB function "ode45", and use the updated control inputs as a warm start for the next iterate. To speed up the solve time, the objective function and functions for calculating nominal states and gradient with analytic Jacobians were converted into C-codes. The parameters of the robot model and NMPC controller used in the simulation experiments are listed in Table I.

Parameter	Value	Parameter	Value
m	43 kg	$\tilde{\mathbf{Q}}_{\boldsymbol{\varphi}}$	$[1, 1, 1]$
I_{xx}	0.41 kgm^2	$\tilde{\mathbf{Q}}_{\mathbf{w}}$	$[10^{-2}, 10^{-2}, 10^{-2}]$
I_{yy}	2.1 kgm^2	$\tilde{\mathbf{Q}}_{\mathbf{p}}$	$[1, 1, 50]$
I_{zz}	2.1 kgm^2	$\tilde{\mathbf{Q}}_{\mathbf{v}}$	$[10^{-1}, 10^{-1}, 10^{-3}]$
Body Length	0.6 m	$\tilde{\mathbf{R}}$	$[10^{-1}, 10^{-1}, 10^{-3}]$
Body Width	0.256 m	f_{min}	10 N
Leg Length	0.34 m	f_{max}	666 N

TABLE I: Parameters of model and NMPC controller

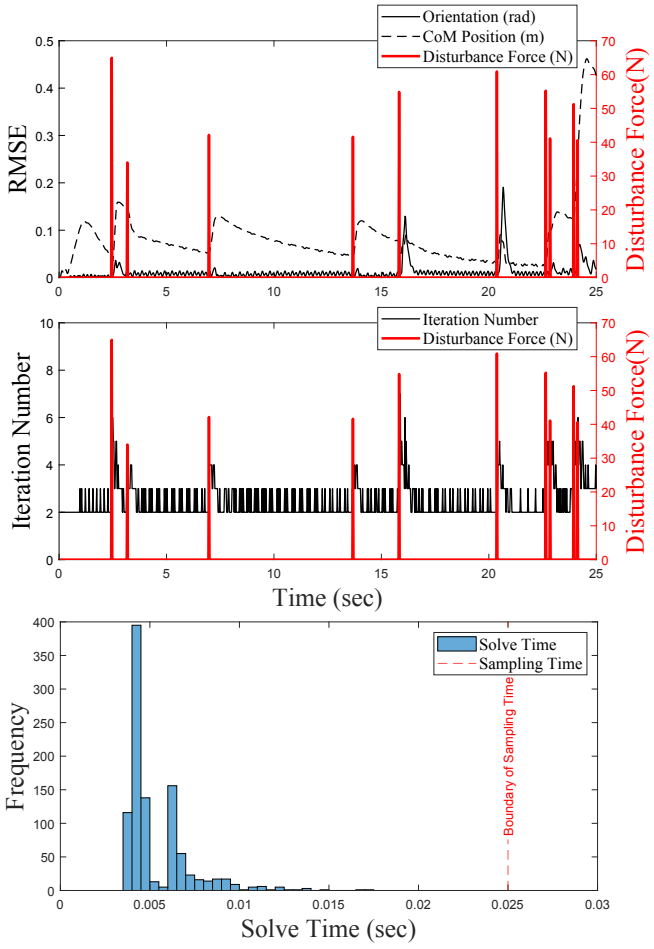

Fig. 3: NMPC of a quadrupedal trotting with the forward speed of 1.5 m/s, while 3-D random disturbance forces of which the magnitude belongs to $[0, 100]$ N were applied. The top figure shows the magnitude of disturbance forces along with root-mean-square-error (RMSE) of orientation and CoM position, respectively. The middle figure displays the iteration number at each step of NMPC during the simulation. The bottom figure displays the histogram of NMPC solve time during the simulation.

B. Dynamic Quadrupedal Locomotion

To verify the performance of the NMPC controller, several dynamic quadrupedal gaits, including trot, pace, bound, and gallop with aerial phase were implemented. Since those simulation results are well described in the supplementary video, here we focus on the trot gait, which can be thought of as a baseline gait. We used $T_{swing} = 0.2$ sec, $T_{stance} = 0.1$ sec, and a set of weight matrices shown in Table I for all the gaits represented in this section.

The robot was instructed with a forward velocity of 1.5 m/s. While moving, a set of 3-D random disturbance forces with a magnitude ranging from 0 N to 100 N were

applied to the robot, as shown in Figure 3. We can identify that the robot can stabilize its balance under strong pushes while tracking the desired quantities well. Even though the iteration number temporarily increases at the moment when a disturbance force is applied to the robot, the NMPC problem could still be solved in real-time in virtue of the efficient proximal Gauss-Newton algorithm proposed in this work.

C. Wall-Climbing Locomotion

Besides the dynamic locomotion on the flat ground, dynamic wall-climbing locomotion was implemented to evaluate the benefit of the orientation parameterization on $\mathrm{SO}(3)$ manifold. In this application, we used $T_{swing} = 0.15$ sec, $T_{stance} = 0.25$ sec, and the same weight matrices in Section V-B except for the weight matrix of position, which was set as $\widetilde{\mathbf{Q}}_{\mathbf{p}} = [20, 1, 20]$. In contrast to the experiments performed in Section V-B, we assume that the sole of the robot can generate negative normal forces on the wall by means of magnets or suction pads. For this reason, the constraints in (14) should be modified accordingly as

$$|f_{i_t}| \leq \mu(f_{pull} - f_{i_n}), \quad |f_{i_n}| \leq min(f_{pull}, f_{max})$$

where f_{i_t} and f_{i_n} are tangential and normal components of the wall reaction forces exerted on i^{th} sole, respectively; f_{pull} is a pulling force generated at the sole in stance phase, and given as $f_{pull} = f_{max} = 666$ N. Under the condition that the pulling force f_{pull} is large enough to sustain the robot's weight, the robot could stabilize its balance while smoothly tracking the desired reference trajectory without falling, and the results can be seen in the supplementary video.

D. Application of Other Legged Robots

In addition to quadrupedal locomotion presented in Section V-B, various types of legged robots, including monopod, biped, tripod, and quintaped robots were successfully performed to further evaluate the performance of proposed NMPC controller. Those simulation results can be seen in the supplementary video.

VI. Conclusion

In this work, a novel NMPC framework for dynamic legged locomotion and an efficient algorithm to solve this constrained nonlinear optimization problem are presented. Moreover, the orientation of the robot among the components that make up the objective function of the optimization problem adopts the manifold configuration of the rotation group. Through the reparameterization, dynamic motions that include configurations in which local parameterizations such as Euler angles suffer singular problems are properly implemented. Furthermore, we derive the analytic Jacobians required for constructing the gradient and the Gauss-Newton Hessian approximation matrix used in the process of solving constrained nonlinear least squares problems. The computational efficiency obtained through the proposed control framework has given a great advantage in real-time implementation of dynamic locomotion control of various types of legged robots. The effects are verified by showing robust stabilization in simulation environments, including push recovery and acrobatic situations such as wall-climbing.

References

[1] J. Koenemann, A. Del Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard, "Whole-body model-predictive control applied to the HRP-2 humanoid," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 3346–3351.

[2] M. Neunert, M. Stäuble, M. Giftthaler, C. D. Bellicoso, J. Carius, C. Gehring, M. Hutter, and J. Buchli, "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1458–1465, 2018.

[3] A. Herdt, H. Diedam, P.-B. Wieber, D. Dimitrov, K. Mombaur, and M. Diehl, "Online walking motion generation with automatic footstep placement," *Advanced Robotics*, vol. 24, no. 5-6, pp. 719–737, 2010.

[4] H.-W. Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems*, 2015.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT cheetah 3 through convex model-predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[6] G. Bledt and S. Kim, "Implementing regularized predictive control for simultaneous real-time footstep and ground reaction force optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 6316–6323.

[7] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli, "Real-time motion planning of legged robots: A model predictive control approach," in *IEEE-RAS International Conference on Humanoid Robotics*, 2017, pp. 577–584.

[8] R. Grandia, F. Farshidian, A. Dosovitskiy, R. Ranftl, and M. Hutter, "Frequency-aware model predictive control," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1517–1524, 2019.

[9] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *International Conference on Robotics and Automation*, May 2019, pp. 8484–8490.

[10] M. Chignoli and P. M. Wensing, "Variational-based optimal control of underactuated balancing for dynamic quadrupeds," *IEEE Access*, vol. 8, pp. 49 785–49 797, 2020.

[11] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous robots*, vol. 40, no. 3, pp. 429–455, 2016.

[12] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[13] G. Wu and K. Sreenath, "Variation-based linearization of nonlinear systems evolving on $\mathrm{SO}(3)$ and $\mathbb{S}^2$," *IEEE Access*, vol. 3, pp. 1592–1604, 2015.

[14] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge University Press, 2017.

[15] G. S. Chirikjian, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[16] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *IEEE international conference on Robotics and automation*, 2013, pp. 3287–3292.

[17] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

[18] T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, 2014.

[19] A. Beck, *First-order methods in optimization*. SIAM, 2017, vol. 25.

[20] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[21] J. D. Lee, Y. Sun, and M. A. Saunders, "Proximal newton-type methods for minimizing composite functions," *SIAM Journal on Optimization*, vol. 24, no. 3, pp. 1420–1443, 2014.

[22] A. G. Pandala, Y. Ding, and H.-W. Park, "qpSWIFT: A real-time sparse quadratic program solver for robotic applications," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3355–3362, 2019.