A linearization method based on Lie algebra for pose estimation in a time horizon

Nicolas Torres Alberto, Lucas Joseph, Vincent Padois, David Daney

A linearization method based on Lie algebra for pose estimation in a time horizon

Nicolas TORRES ALBERTO, Lucas JOSEPH, Vincent PADOIS and David DANEY

Abstract In this paper, a strategy for linear pose estimation over a time horizon is presented. This linearization is crucial for the computationally efficient formulation of predictive and optimization-based control problems in robotics. The proposed approach is based on a truncation of the Magnus expansion for the approximation of the exponential map derivative and employs Lie algebra to represent position and orientation, allowing for a unified vectorial representation in vector form that can be integrated linearly over time, offering a convenient formulation for optimization solvers. The method shows promising results for precision and computation times.

1 Introduction

Collaborative robotics has fueled a growing demand for more complex control schemes [1]. Indeed, during human-robot interactions, robots evolve in a changing environment. Consequently, they are required to adapt reactively to unpredictable events while satisfying a set of constraints in order to provide safety guarantees.

In this context of complex robotic applications, optimization-based controllers offer a convenient way to formulate tasks to be achieved under constraints. These problems can be formulated over one control time step, i.e. reactively [5], but can also lead to much more robust behaviours when considering control over receding time horizons. Most humanoid control approaches rely on such a Model Predictive Control (MPC) paradigm [6] but these type of approaches remain to be proven useful in other complex robotics contexts.

One of the key issues when considering a receding horizon formulation is the ability to locally predict the evolution of the robot state given a control horizon while maintaining the computation cost low enough to allow for frequent optimization updates. While recent

Nicolas TORRES ALBERTO
Stellantis, Centre Technique Vélizy, France e-mail: nicolas.torres@stellantis.fr,
Lucas JOSEPH
CNRS, Solvay, Univ. Bordeaux, France e-mail: lucas.joseph-ext@solvay.com,
Vincent PADOIS
Auctus, Inria Bordeaux, Talence, France e-mail: vincent.padois@inria.fr,
David DANEY
Auctus, Inria Bordeaux, Talence, France e-mail: david.daney@inria.fr

works have shown the ability to perform such locally optimal computation in a non linear way [7], linear formulations remain advantageous both from a computational and control problem formulation point of view. Yet, to the best of our knowledge, an efficient linearization method for both 3D position + orientation[1] while offering a convenient formulation for optimization algorithms remains to be proposed.

The objective of this paper is to present a way to estimate, at each time step, the robot pose trajectory in a receding horizon by reconstructing it from the initial pose and a known input twist horizon.

The proposed method relies on the Lie algebra $\mathfrak{se}^3$ associated to the tangential space of a pose described in $\mathbb{SE}^3$. This method (based on the Magnus Expansion (ME) [4]) leads to a linear expression of the transformation between two poses, using the logarithmic map. In its truncated form, it leads to a linear propagation/integration formula which has an explicit algebraic form and takes a vector form adapted to the writing of recursive state propagation general formulas.

In Section 2, some fundamentals about Lie groups are presented. Section 3 presents the linearization strategy. Finally, Section 4 presents a simple kinematic simulation to showcase the results.

2 Fundamentals

This Section introduces the fundamental modeling aspects underlying the general objectives of this article. Lie groups and their algebras are first presented in vector and matricial forms. Afterwards, the receding horizon linearized system form pursued in this paper is presented.

2.1 Lie Groups and Algebras

A pose is composed of a position $\boldsymbol{p} \in \mathbb{R}^3$ and an orientation. For this paper, orientations are represented as a rotation matrix $\boldsymbol{R}$ that belongs to the Special Orthogonal Group ($\mathbb{SO}^3$), a Lie group. The robot pose $\boldsymbol{\chi} = (\boldsymbol{R}, \boldsymbol{p})$ is composed of two elements, belonging to the Special Euclidean Group ($\mathbb{SE}^3$), another Lie group.

Given a Lie group $\mathcal{G}$, there exists a Lie algebra $\mathfrak{g}$ which is the logarithm map of the Lie group $\mathcal{G}$ such that $\log : \mathcal{G} \to \mathfrak{g}$, with an inverse operation $\exp : \mathfrak{g} \to \mathcal{G}$. This algebra corresponds to the tangent space of an infinitesimal increments.

In the case of pose, a infinitesimal increment is called a twist $\boldsymbol{T} \in \mathfrak{se}^3$, composed of an infinitesimal orientation increment (or rotational speed) $\boldsymbol{\omega} \in \mathfrak{so}^3$ (that in term corresponds to the Lie algebra of $\mathbb{SO}^3$) and linear component (linear velocity) $\boldsymbol{v} \in \mathbb{R}^3$.

Given $\boldsymbol{\xi} = \log \boldsymbol{\chi}$, $\boldsymbol{\xi}$ represents the equivalent pose increment that will produce the same linear and rotational displacement as the homogenous transformation $\boldsymbol{\chi}$(also referred as a pose here) when integrated for 1s. By abuse of language, $\boldsymbol{\xi}$ might also be referred as a pose throughout this paper (as it can be used to retrieve the pose with $\boldsymbol{\chi} = e^{\boldsymbol{\xi}}$). Even though both $\boldsymbol{T}$ and $\boldsymbol{\xi}$ both belong in $\mathfrak{se}^3$, only $\boldsymbol{T}$ will be used to represent infinitesimal increments throughout this paper. The same can be said for $\boldsymbol{\phi} = \log \boldsymbol{R}$, with $\boldsymbol{\omega}$ and $\boldsymbol{\phi}$ both belonging in $\mathfrak{so}^3$, but only $\boldsymbol{\omega}$ representing an infinitesimal increment.

It is important to remark that $\boldsymbol{\phi} \in \mathfrak{so}^3$ yields a one to one correspondence to $\boldsymbol{R} \in \mathbb{SO}^3$ through the log map as long as $||\boldsymbol{\phi}|| \leq \pi$. This in turn extends to $\boldsymbol{\xi}$ and $\boldsymbol{\chi}$ with $\mathfrak{se}^3$ and $\mathbb{SE}^3$.

[1] Typically of the robot end-effector.

Due to the skew-symmetric nature of $\mathfrak{so}^3$, $\boldsymbol{\omega}$ admits a vector parametrization in $\mathbb{R}^3$ and a matricial form in $\mathbb{R}^{3\times3}$. One can go from one to the other through the reciprocal operations: the hat map $\cdot^\wedge : \mathbb{R}^3 \to \mathbb{R}^{3\times3}$ and the vee map $\cdot^\vee : \mathbb{R}^{3\times3} \to \mathbb{R}^3$. This extends to $\mathfrak{se}^3$ between $\mathbb{R}^6$ and $\mathbb{R}^{4\times4}$ as:

$$\boldsymbol{T}^\wedge = \begin{bmatrix} \boldsymbol{\omega}^\wedge & \boldsymbol{v} \\ 0 & 0 \end{bmatrix} \text{ with } \boldsymbol{\omega}^\wedge = \begin{bmatrix} 0 & -\boldsymbol{\omega}_3 & \boldsymbol{\omega}_2 \\ \boldsymbol{\omega}_3 & 0 & -\boldsymbol{\omega}_1 \\ -\boldsymbol{\omega}_2 & \boldsymbol{\omega}_1 & 0 \end{bmatrix} \tag{1}$$

$$\text{with: } \boldsymbol{\omega} = [\omega_1, \omega_2, \omega_3]^T, \boldsymbol{v} = [v_1, v_2, v_3]^T, \boldsymbol{T} = [\boldsymbol{v}^T \ \boldsymbol{\omega}^T]^T \tag{2}$$

Even though the two forms are equivalent, the vector form is employed in this article[2], unless specified.

The Lie algebra is equipped with the commutation operator $[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{XY} - \boldsymbol{YX} = \mathrm{ad}_{\boldsymbol{X}}\boldsymbol{Y}$ for $\boldsymbol{X}, \boldsymbol{Y} \in \mathfrak{g}$. This is also referred as the *adjoint action* endomorphism, as it is a linear map onto itself such that $\mathrm{ad}_{\mathfrak{g}} \cdot : [\mathfrak{g}, \mathfrak{g}] \to \mathfrak{g}$.

For the case of: $\boldsymbol{\phi} \in \mathfrak{so}^3$ and $\boldsymbol{\xi} = [\boldsymbol{\rho}^T, \boldsymbol{\phi}^T]^T \in \mathfrak{se}^3$ (with $\boldsymbol{\rho} \in \mathbb{R}^3$, the linear component of $\boldsymbol{\xi}$), the ad operation is:

$$\text{for } \mathfrak{so}^3\text{: } \mathrm{ad}_{\boldsymbol{\phi}_1}\boldsymbol{\phi}_2 = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2] = \boldsymbol{\phi}_1 \times \boldsymbol{\phi}_2 = \boldsymbol{\phi}_1{}^\wedge.\boldsymbol{\phi}_2 \qquad \text{with: } \mathrm{ad}_{\boldsymbol{\phi}_1} = \boldsymbol{\phi}_1{}^\wedge \tag{3}$$

$$\text{for } \mathfrak{se}^3\text{: } \mathrm{ad}_{\boldsymbol{\xi}_1}\boldsymbol{\xi}_2 = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2] = \begin{bmatrix} \boldsymbol{\phi}_1 \times \boldsymbol{\rho}_2 + \boldsymbol{\rho}_1 \times \boldsymbol{\phi}_2 \\ \boldsymbol{\phi}_1 \times \boldsymbol{\phi}_2 \end{bmatrix} \quad \text{with: } \mathrm{ad}_{\boldsymbol{\xi}_1} = \begin{bmatrix} \boldsymbol{\phi}_1{}^\wedge & \boldsymbol{\rho}_1{}^\wedge \\ \boldsymbol{0} & \boldsymbol{\phi}_1{}^\wedge \end{bmatrix} \tag{4}$$

For a more comprehensive introduction into Lie groups, refer to [11].

2.2 Receding horizon

Let us define a dynamic system with vector state $\boldsymbol{x}$ and input $\boldsymbol{u}$. Discretizing a time horizon in H time steps of duration Δt, ones obtains $\boldsymbol{x}(t_k) = \boldsymbol{x}_k$, $\boldsymbol{u}(t_k) = \boldsymbol{u}_k$ for $t_k = k\Delta t$ where $k = 0, 1, \ldots, H$. Notice t_0 (the starting time of the horizon) can be chosen arbitrarily without loss of generality.

One can formulate a linear receding horizon expression as:

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k \text{ for } k = 0, 1, \ldots, H-1 \tag{5}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are respectively the state system and input matrix. This linear discrete time dynamic model expression can straightforwardly be written in a recursive matricial form over the whole horizon. This form is central to the formulation of MPC as a constrained Quadratic Progam (QP).

3 Linear pose estimation

In this Section, the propagation of the pose through a manifold is proposed.

3.1 Pose estimation in a horizon

For a robot pose $\boldsymbol{\mathcal{X}} \in \mathbb{SE}^3$, let us reconstruct a pose trajectory $\boldsymbol{\mathcal{X}}(t)$ in time given some initial horizon state $\boldsymbol{\mathcal{X}}(t_0) = \boldsymbol{\mathcal{X}}_0$ and some input twist trajectory $\boldsymbol{T}(t) \in \mathfrak{se}^3$, consisting of infinitesimal increments of pose in time.

[2] The log and exp operations are defined for matrices. By abuse of notation, throughout this article they will also operate on the lie algebra vector form (given the fact that it is easy to retrieve the matrix form through the hat map). For the same reason, assume the results of the log and exp operations are expressed in vector form.

As described in Sect. 2.1, the robot pose belongs to a Lie group and its twist, to its algebra (its tangential space).

Discretizing a time horizon in H time steps of duration Δt, we obtain $\boldsymbol{\mathcal{X}}(t_k) = \boldsymbol{\mathcal{X}}_k$, $\boldsymbol{T}(t_k) = \boldsymbol{T}_k$ for $t_k = k\Delta t$ where $k = 0, 1, \ldots, H$. An infinitesimal twist can be integrated to obtain the next pose as:

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_k e^{\boldsymbol{T}_k \Delta t} \tag{6}$$

This enables expressing any state in the horizon $\boldsymbol{\mathcal{X}}_{k+1}$ as a function of $\boldsymbol{\mathcal{X}}_0$ and the successive $\boldsymbol{T}_k$ applied:

$$\boldsymbol{\mathcal{X}}_{k+1} = \boldsymbol{\mathcal{X}}_0 \underbrace{\prod_{i=0}^{k} e^{\boldsymbol{T}_i \Delta t}}_{\boldsymbol{f}(\boldsymbol{T}_0, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_k)} \tag{7}$$

The focus of this paper is to showcase a linear approximation of $\boldsymbol{\xi}(t)$ (the logarithmic map of $\boldsymbol{\mathcal{X}}(t)$), using Eq. 7.

3.2 From the Magnus Expansion to an estimate of the logarithm map

The objective of this Section is to find an approximation of $\boldsymbol{\xi}(t)$ in the form of Eq. 5, with system state $\boldsymbol{\xi}$ and input $\boldsymbol{T}$, by exploiting Eq. 7.

Consider the following differential equation:

$$\boldsymbol{Y}'(t) = \boldsymbol{Y}(t)\boldsymbol{U}(t), \text{ with } \boldsymbol{Y}(t_0) = \boldsymbol{Y}_0 \tag{8}$$

with $\boldsymbol{Y}(t) \in \mathcal{G}$ and $\boldsymbol{U}(t) \in \mathfrak{g}$ (as proposed in [12]). It is possible to find a solution in the form:

$$\boldsymbol{Y}(t) = \boldsymbol{Y}_0 e^{\boldsymbol{\Omega}(t)} \tag{9}$$

Consequently, $\boldsymbol{Y}'(t) = \boldsymbol{Y}_0 e^{\boldsymbol{\Omega}(t)} \mathrm{dexp}_{\boldsymbol{\Omega}(t)} \boldsymbol{\Omega}'(t)$ from which, based on Eq. 8, one can identify $\boldsymbol{U}(t)$ as

$$\boldsymbol{U}(t) = \mathrm{dexp}_{\boldsymbol{\Omega}(t)} \boldsymbol{\Omega}'(t) \tag{10}$$

where dexp_x denotes the derivative of the exponential map[3] at x.

Actually, the main interest lies in finding $\boldsymbol{\Omega}(t)$ that satisfies Eq. 8. We can retrieve $\boldsymbol{\Omega}'(t)$ from Eq. 10 by inverting dexp:

$$\boldsymbol{\Omega}' = \mathrm{dexp}_{\boldsymbol{\Omega}}^{-1} \boldsymbol{U}(t) = \sum_{n=0}^{\infty} \frac{B_n}{n!} \mathrm{ad}_{\boldsymbol{\Omega}}^n \boldsymbol{U}(t) \tag{11}$$

$$\mathrm{ad}_{\boldsymbol{\Omega}}^n = \underbrace{[\boldsymbol{\Omega}(t), [\ldots, [\boldsymbol{\Omega}(t)}_{n-times}, \boldsymbol{U}(t)]]] \tag{12}$$

where B_n are the Bernoulli numbers. The relation between Eq. 11 and dexp^{-1} is developed in [10] and [9]. Note that $\boldsymbol{\Omega}(t) = \int_{t_o}^{t} \boldsymbol{U}(x)dx$ iif $\boldsymbol{U}$ is constant or commutes with its primitive (i.e. $[\boldsymbol{U}(t), \int \boldsymbol{U}(t)dt] = 0$). Otherwise, $\boldsymbol{\Omega}(t)$ is an infinite series known as the *Magnus expansion* [8][3].

The current paper uses a truncated version of Eq. 11 up to $n = 2$, referred as ME2 from now on. Remembering that $B_n = 0$ for all odd $n > 1$ (which implies that ME3 $\equiv$ ME2), yields:

[3] The derivative of the exponential map is often presented as $\frac{d}{dt} e^{x(t)} = e^{x} \frac{(1 - e^{\mathrm{ad}_x})}{\mathrm{ad}_x} \frac{dx}{dt}$ or, with a different notation but equivalently: $\frac{d}{dt} e^{x(t)} = \mathrm{dexp}_{x(t)} x'(t)$

$$\boldsymbol{\Omega}'(t) \approx \boldsymbol{U}(t) + \frac{[\boldsymbol{\Omega}(t), \boldsymbol{U}(t)]}{2} + \frac{[\boldsymbol{\Omega}(t), [\boldsymbol{\Omega}(t), \boldsymbol{U}(t)]]}{12} \tag{13}$$

Given Eq. 13, we can now see how Eq. 9 is in fact the time continuous version of Eq. 7. Furthermore, $\boldsymbol{U}(t)$ can be interpreted as the time continuous version of the trajectory twist $\boldsymbol{T}(t)$ while $\boldsymbol{\Omega}(t)$ incorporates its cumulative effects in time over the initial pose $\boldsymbol{\chi}_0$. Note that $\boldsymbol{Y}(t)$ and $\boldsymbol{U}(t)$ do not necessarily commute: this means that $\boldsymbol{Y}_0\boldsymbol{U}_0\boldsymbol{U}_1 \neq \boldsymbol{Y}_0\boldsymbol{U}_1\boldsymbol{U}_0$ (or for any other order for that matter). This is exactly why $\boldsymbol{\Omega}(t)$ cannot be found via a trivial integration and why the Magnus expansion is required.

By discretizing $\boldsymbol{\Omega}_k$, we can construct an equivalent for Eq. 7:

$$\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_0 e^{\boldsymbol{\Omega}_k(\boldsymbol{T}_0, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_k)} \tag{14}$$

and computing the logarithm map of the pose as $\boldsymbol{\xi}_{k+1} = \log(\boldsymbol{\chi}_{k+1})$, we arrive at:

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{K} + \boldsymbol{\Omega}_k(\boldsymbol{T}_0, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_k) \tag{15}$$

where $\boldsymbol{K} \in \mathfrak{se}^3$ corresponds to some initial conditions that will satisfy for $\boldsymbol{\chi}_0$.

Finally, linearizing Eq. 15 at the start of the horizon $\boldsymbol{\xi}_0$ yields:

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_0 + \Delta t . \mathrm{dexp}_{\boldsymbol{\xi}_0}^{-1} . (\boldsymbol{T}_0 + \boldsymbol{T}_1 + \ldots + \boldsymbol{T}_k) \tag{16}$$

Two remarkable properties about this equation are:

- $\boldsymbol{\xi}$ is a 6-vector in $\mathfrak{se}^3$, a more space-efficient (equivalent) representation of pose, compared to a 4×4 matrix in $\mathbb{SE}^3$, allowing to be more easily embedded in optimization problems.
- because the commutator is an $\mathfrak{se}^3$ operation (refer to Sect. 2.1), the result of Eq. 16 also lies in $\mathfrak{se}^3$ (for small enough Δt).

This linear approximation implies the assumption that $\mathrm{dexp}_{\boldsymbol{\xi}_0}$ does not change much during the horizon, which is an accepted compromise for any linearization approach and is also why the approximation gets worse with longer horizons.

It can be seen that equation Eq. 16 resembles Eq. 5, enabling its use in linear MPC.This equation constitutes the main contribution of this paper.

4 Results

The goal is to estimate, at each time step, the robot pose trajectory in a receding horizon $\boldsymbol{\chi}_k$ by reconstructing it from the initial horizon pose $\boldsymbol{\chi}_0$ and a perfectly known input twist $\boldsymbol{T}_k$.

The objective is not to estimate $\boldsymbol{\chi}_k$ directly in a matricial form. Instead, one can estimate $\boldsymbol{\xi}_k$ using Eq. 16. The pose can be retrieved from $\boldsymbol{\chi}_k = e^{\boldsymbol{\xi}_k}$.

Let the starting and ending poses of a robot end-effector path be $\boldsymbol{\chi}_0, \boldsymbol{\chi}_f \in \mathbb{SE}^3$ as depicted in Fig. 1. In this experiment, to represent a typical movement of a Franka Emika Panda in its workspace, two paths with different positional and rotational displacement were generated. For trajectory 1:$||\boldsymbol{p}_f - \boldsymbol{p}_0|| = 0.71$m and $||ang(\delta)|| = 0.94$rad ; while for trajectory 2:$||\boldsymbol{p}_f - \boldsymbol{p}_0|| = 0.32$m and $||ang(\delta)|| = 0.90$rad . The rotational displacement (the norm of the angular component $ang(\delta)$) is computed from $\delta = \log(\boldsymbol{\chi}_0^{-1}\boldsymbol{\chi}_f)$.

Defining $\alpha \in [0, 1]$ to discretize the path between both poses as: $\boldsymbol{\chi}_k^* = \boldsymbol{\chi}_0 e^{\alpha_k \delta}$ yields a ground truth reference velocity twist $\boldsymbol{T}_k^* = \frac{\log(\boldsymbol{\chi}_k^{*-1}\boldsymbol{\chi}_{k+1}^*)}{\Delta t}$.

This implies that the reference trajectory can be reconstructed with:

$$\boldsymbol{\chi}_{k+1} = \boldsymbol{\chi}_k e^{\Delta t \boldsymbol{T}_k^*} \tag{17}$$

$$\text{where: } \log(\boldsymbol{\chi}_k) = \boldsymbol{\xi}_k \tag{18}$$

The Ruckig library [2] is used[4] to compute a normalized $\alpha(t)$ in time to find the time-optimal trajectory, following a trapezoidal trajectory (in velocity), as shown in Fig. 2.

We showcase the precision of linear integration of a robot's end-effector pose by using Eq. 16 throughout an H-step horizon and comparing it against Eq. 17, the ground truth. Fig. 3 depicts the obtained errors with respect to the perfect propagation.

Assuming a receding horizon approach where only horizon estimation matters, the pose is reset back to ground truth at the end of each horizon (every 300ms) to showcase the maximum error obtained with the method and to exemplify how the precision worsens towards the end of the horizon.

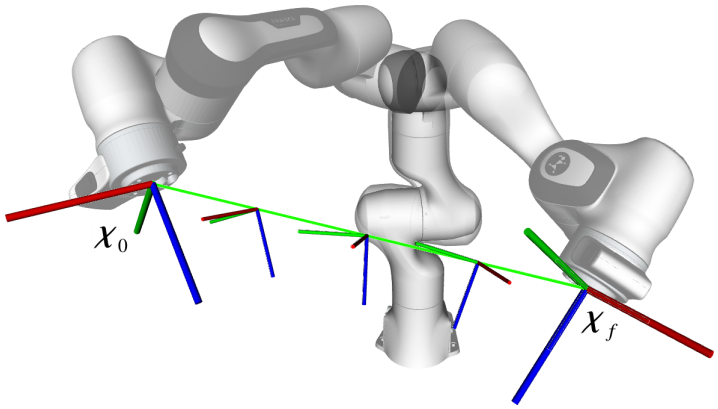
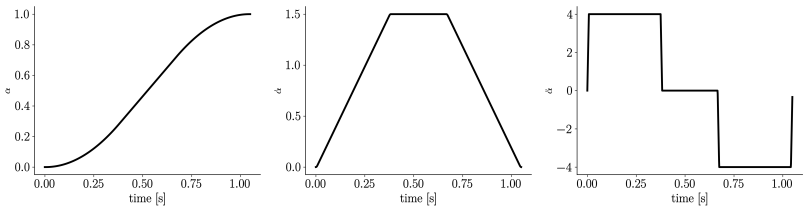


Fig. 1: Depiction of the path taken taking the end-effector of a Franka Emika Panda from a starting to the ending poses $\boldsymbol{\chi}_0, \boldsymbol{\chi}_f \in \mathbb{SE}^3$.

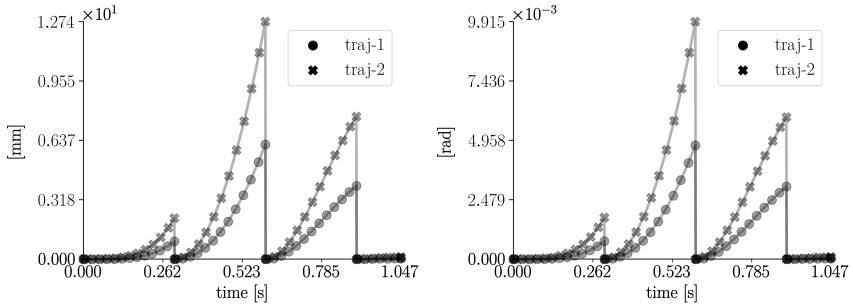


(a) $\alpha(t)$ (b) $\dot{\alpha}(t)$ (c) $\ddot{\alpha}(t)$

Fig. 2: Normalized $\alpha(t)$ for a time-optimal trajectory generated with a trapezoidal profile (in velocity).

In table 1, the computation time and error averages obtained for the ME2 approximation are presented. The method achieves a linear and angular error of 1.09 ± 6.1mm and $0.048 \pm 0.27°$(for the longest trajectory), respectively. This is done in about two thirds of the time it takes to do it with the exponential (exact but non linear). ME2 offers not only a precision level practical for many purposes but it could offer a speed improvement when

[4] The code used for experiments in this paper is publicly available at: https://gitlab.inria.fr/auctus-team/publications/shared-paper-code/ark2022-lin



(a) Norm of linear error

(b) Norm of angular error

Fig. 3: Variable speed integration using ME2 for a 300ms horizon.

Trajectory	**Compute Time**			**Linear Error**			**Angular Error**		
	Avg	Rel	Max	Avg	Max	Std	Avg	Max	Std
ref-1	0.23μs	100.0%	1.57μs						
traj-1	0.15μs	62.4%	3.74μs	1.09mm	6.15mm	±1.47mm	0.05°	0.27°	±0.06°
ref-2	0.18μs	100.0%	0.82μs						
traj-2	0.13μs	72.3%	0.23μs	2.26mm	12.74mm	±2.99mm	0.10°	0.57°	±0.13°

Table 1: **Shows the average compute time and error for the horizons shown in Fig. 3.**
To showcase the trade off between precision and computation time, this comparison shows the error between the ground truth reference (Eq. 17) and the proposed ME-based method (Eq. 16). The results are showed for 2 generated trajectories with their respective references.
Compute time refers to the average for each Δt step. Relative computation time is calculated with respect to each reference average computation time.

both methods of integration are possible (the exponential integration method cannot be embedded in a QP, which is the focus of this paper). This computational time improvement could also be exploited on robots with limited resources.

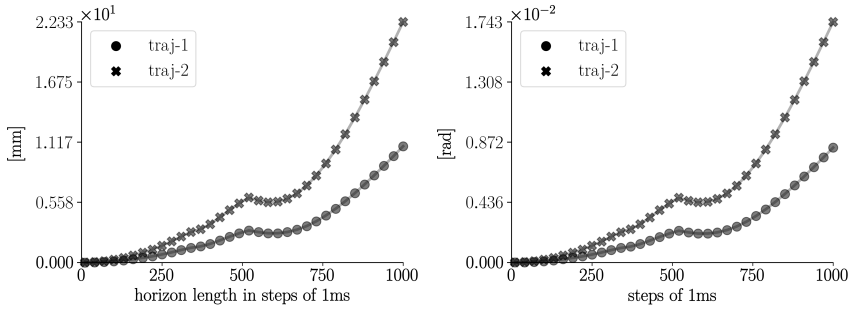
To further illustrate the effects of enlarging the horizon duration, we show in Fig. 4 the same experiment run multiple times for different values of H (number of time steps). It can be seen that the longer the horizon, the greater the error is. Thus, a compromise must be made in order to define a horizon that allows to predict further enough in time without compromising precision.

5 Conclusions

Lie algebras offer a convenient framework for pose description and system linearization on manifolds, allowing for vector expressions that can be embedded in a linear MPC problem, unlike their matricial forms.

The variety of strategies available offers an opportunity for the community to develop the tools required for more intelligent robotics applications that can optimize through predicted horizons to solve complex problems like safety in human-robot interactions, energy efficiency and trajectory feasibility of dynamically constrained systems.

In future research, we would like to employ this algorithm for the dynamic replanification (based on acceleration increments instead of velocity) of end-effector trajectories subject to kino-dynamic constraints.



(a) Norm of linear error

(b) Norm of angular error

Fig. 4: Average error for different horizon lengths.

The estimation of a bound on the maximum error induced by the proposed linear approximation is also part of the future work.

Acknowledgements This work was funded by the ANRT and Stellantis Group and performed within the framework of the OpenLab AI.

References

1. Ajoudani, A., Zanchettin, A.M., Ivaldi, S., Albu-Schäffer, A., Kosuge, K., Khatib, O.: Progress and prospects of the human–robot collaboration. Autonomous Robots **42**(5), 957–975 (2018)
2. Berscheid, L., Kröger, T.: Jerk-limited Real-time Trajectory Generation with Arbitrary Target States. Robotics: Science and Systems XVII (2021)
3. Blanes, S., Casas, F., Oteo, J., Ros, J.: The Magnus expansion and some of its applications. Physics Reports, 470(5-6), 151-238. Physics Reports **470**, 151–238 (2008). DOI 10.1016/j.physrep.2008.11.001
4. Casas, F., Iserles, A.: Explicit Magnus expansions for nonlinear equations. J. Phys. A: Math. Gen **39**, 5445–5461 (2006). DOI 10.1088/0305-4470/39/19/S07
5. Escande, A., Mansard, N., Wieber, P.B.: Hierarchical quadratic programming: Fast online humanoid-robot motion generation. The International Journal of Robotics Research **33**(7), 1006–1028 (2014)
6. Ibanez, A., Bidaud, P., Padois, V.: Optimization-Based Control Approaches to Humanoid Balancing, pp. 1541–1567. Springer Netherlands, Dordrecht (2019). DOI 10.1007/978-94-007-6046-2_71. URL `https://doi.org/10.1007/978-94-007-6046-2_71`
7. Kleff, S., Meduri, A., Budhiraja, R., Mansard, N., Righetti, L.: High-frequency nonlinear model predictive control of a manipulator. In: ICRA 2021 (2021)
8. Magnus, W.: On the exponential solution of differential equations for a linear operator (1954). DOI 10.1002/CPA.3160070404
9. Rossmann, W.: Lie Groups: An Introduction Through Linear Groups. Oxford University Press (2006)
10. Schur, F.: Zur Theorie der endlichen Transformationsgruppen. Mathematische Annalen **38**, 263–286 (1891). URL `https://eudml.org/doc/157545`
11. Solà, J., Deray, J., Atchuthan, D.: A micro Lie theory for state estimation in robotics. ArXiv (2018)
12. Zanna, A.: Collocation and Relaxed Collocation for the Fer and the Magnus Expansions. Siam Journal on Numerical Analysis - SIAM J NUMER ANAL **36** (1999). DOI 10.1137/S0036142997326616