Analytical method for differentiation of robot Jacobian

J.-Y. Rhee and B. Lee✉

Model-based control algorithms commonly use joint acceleration as a desired trajectory. As the velocity trajectory in the task space is converted into joint velocity by multiplying using a Jacobian matrix, to derive the acceleration in joint space, Jacobian differentiation is required. Although the numerical method for Jacobian differentiation gives sufficiently accurate approximations, it incurs a high computation cost because this method involves computing the forward kinematics twice and Jacobian derivation for every element of the Jacobian matrix. Consequently, this causes difficulties for real-time control. To resolve this, an analytical differentiation method is proposed. Through recursive computation, differentiation is performed without any approximation, in an acceptably small computational time. Performance of the proposed method was verified by comparison with numerical derivation using a computer simulation.

Introduction: Accurate and fast kinematics computation is essential for precision robot control. More accurate reference trajectories provide better control performance. Therefore, lots of model-based controllers such as a computed torque-like controller require a great deal of information regarding the reference trajectory including desired joint accelerations for its control inputs [1–3]. There are two methods to obtain the desired joint accelerations: the interpolation method and the direct acceleration method. In the interpolation method, two successive joint positions are first divided into several knot points. Second, an approximated trajectory is generated by interpolating the knot points. Polynomial curves are frequently used for approximation [4]. This method provides adequate performance if the knot points are provided, and is typically enough for a classic manipulator robot. When the dynamic effects such as velocity and acceleration of centre of mass are crucial for task planning, a more accurate method is required. The direct acceleration method uses the end-effector's velocity and acceleration in Cartesian space to derive the joint space accelerations [5, 6]

$$\ddot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\ddot{\boldsymbol{x}} - \dot{\boldsymbol{J}}\dot{\boldsymbol{q}}) \tag{1}$$

As shown in (1), to obtain the joint space acceleration Jacobian differentiation is necessary. When the Jacobian differentiation is numerically derived, it is possible to obtain results within the bounds of practical error, though this incurs computational costs that make it difficult to use this method for real-time control. In this Letter, to reduce the computational cost without approximations, an analytical method is proposed to differentiate the Jacobian matrix for derivatives of the joint accelerations.

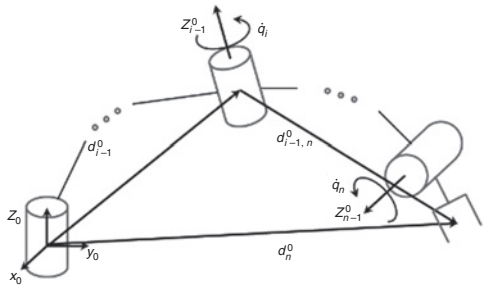

Fig. 1 *Coordinate system of articulated body*

Derivation of algorithm: Fig. 1 shows a kinematic chain describing the joint motion of the articulated links. z^0_{i-1} is the z-axis of the $(i-1)$th coordinate frame from the base frame. Along this axis, the ith link is rotated. The variable designating the joint is q_i. $\boldsymbol{d}^0_{i-1}$ is the origin vector of the $(i-1)$th coordinate frame with respect to the base frame, whereas $\boldsymbol{d}^0_{i-1,n}$ is the displacement from the $(i-1)$th coordinate frame to the nth coordinate frame. From this configuration, the associated Jacobian matrix is described as follows:

$$\boldsymbol{J}_i = \begin{bmatrix} \boldsymbol{z}^0_{i-1} \times \boldsymbol{d}^0_{i-1,n} \\ \boldsymbol{z}^0_{i-1} \end{bmatrix} \tag{2}$$

where $\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 & \dots & \boldsymbol{J}_i & \dots & \boldsymbol{J}_n \end{bmatrix}$ and $\boldsymbol{J}_i$ is the ith column vector corresponding to the ith joint velocity $\dot{q}_i$. Differentiating (2) gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{J}_i = \begin{bmatrix} \left(\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}^0_{i-1}\right) \times \boldsymbol{d}^0_{i-1,n} + \boldsymbol{z}^0_{i-1} \times \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_{i-1,n} \\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}^0_{i-1} \end{bmatrix} \tag{3}$$

Equation (3) can be divided into two parts: the derivative of the rotation axis, $(d/dt)\boldsymbol{z}^0_{i-1}$, and the derivative of the displacement vector, $(d/dt)\boldsymbol{d}^0_{i-1,n}$. The $(i-1)$th rotation axis can be expressed with respect to the $(i-1)$th coordinate frame with rotation matrix from the base frame to the $(i-1)$th frame

Table 1: Algorithm for derivation of Jacobian and its derivative

```
/* Jacobian computation*/
ω^0_0 ← [0 0 0]^T;
z^0_0 ← [0 0 1]^T;
J_1 ← [ z^0_0 × d^0_n ; z^0_0 ];
For i = 2 to n do
J_i ← [ z^0_{i-1} × (d^0_n − d^0_{i-1}) ; z^0_{i-1} ]; // Jacobian of the ith column
ω^0_{i-1} ← ω^0_{i-2} + z^0_{i-2} q̇_{i-1}; // angular velocity for derivative
End
/* Jacobian differentiation */
β ← J_{v,i} q̇_i;
For i = n to 2 do
ż^0_{i-1} ← ω^0_{i-1} × z^0_{i-1};
α ← [0 0 0]^T;
For j = 1 to i − 1 do
α ← α + z^0_{j-1} × (d^0_n − d^0_{i-1}) q̇_j;
End
J̇_i ← [ ż^0_{i-1} × (d^0_n − d^0_{i-1}) + z^0_{i-1} × (α + β) ; ż^0_{i-1} ];
β ← β + J_{v,i-1} q̇_{i-1};
End
J̇_1 ← [ z^0_0 × β ; [0 0 0]^T ];
```

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}^0_{i-1} = \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{R}^0_{i-1}\boldsymbol{z}^{i-1}_{i-1}\right) \tag{4}$$

As z^{i-1}_{i-1} is a constant vector, the differentiation of (4) is simply reduced to a cross-product of the angular velocity $\boldsymbol{\omega}^0_{i-1}$ and the rotation vector $\boldsymbol{z}^0_{i-1}$. The angular velocity is the accumulation of angular motions of all precedent joints. Consequently, (4) becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{z}^0_{i-1} = \left(\sum_{j=1}^{i-1}\boldsymbol{z}^0_{j-1}\dot{q}_j\right) \times \boldsymbol{z}^0_{i-1} \tag{5}$$

The displacement vector is the difference between the position of the end-effector and the $(i-1)$th coordinate origin. Thus

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_{i-1,n} = \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_n - \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_{i-1} \tag{6}$$

The differentiation of the position vector in (6) can be computed by accumulating cross-products of the angular velocities of the precedent joints and their respective displacement vectors from the precedent coordinates to the end-effector's position

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_n = {} & \boldsymbol{z}^0_0 \times (\boldsymbol{d}^0_n - \boldsymbol{d}^0_0)\dot{q}_1 + \dots + \boldsymbol{z}^0_{i-2} \times (\boldsymbol{d}^0_n - \boldsymbol{d}^0_{i-2})\dot{q}_{i-1} \\ & + \boldsymbol{z}^0_{i-1} \times (\boldsymbol{d}^0_n - \boldsymbol{d}^0_{i-1})\dot{q}_i + \dots + \boldsymbol{z}^0_{n-1} \times (\boldsymbol{d}^0_n - \boldsymbol{d}^0_{n-1})\dot{q}_n\end{aligned} \tag{7}$$

Similarly

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{d}^0_{i-1} = \boldsymbol{z}^0_0 \times (\boldsymbol{d}^0_{i-1} - \boldsymbol{d}^0_0)\dot{q}_1 + \dots + \boldsymbol{z}^0_{i-2} \times (\boldsymbol{d}^0_{i-1} - \boldsymbol{d}^0_{i-2})\dot{q}_{i-1} \tag{8}$$

Combining (7) and (8) produces

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{d}_n^0-\boldsymbol{d}_{i-1}^0\right)=\sum_{j=1}^{i-1}\left(\boldsymbol{z}_{j-1}^0\times\left(\boldsymbol{d}_n^0-\boldsymbol{d}_{i-1}^0\right)\dot{q}_j\right)+\sum_{j=i}^{n}\left(\boldsymbol{z}_{j-1}^0\times\left(\boldsymbol{d}_n^0-\boldsymbol{d}_{j-1}^0\right)\dot{q}_j\right)\tag{9}$$

From (5) and (9), Jacobian differentiation can be computed analytically and quickly enough to enable real-time control. This algorithm is summarised in Table 1. The algorithm contains two parts: Jacobian computation and Jacobian differentiation. To increase computational efficiency, the intermediate results obtained during Jacobian derivation are stored and reused for the Jacobian differentiation.

Table 2: DH parameters for three axis robot

Number of link	Joint variable: θ	Offset: d	Link length: l	Twist angle: α
1	$\frac{1}{2}\pi+q_1$	0.1	0	$-\frac{1}{2}\pi$
2	q_2	0	0.3	0
3	q_3	0	0.3	0

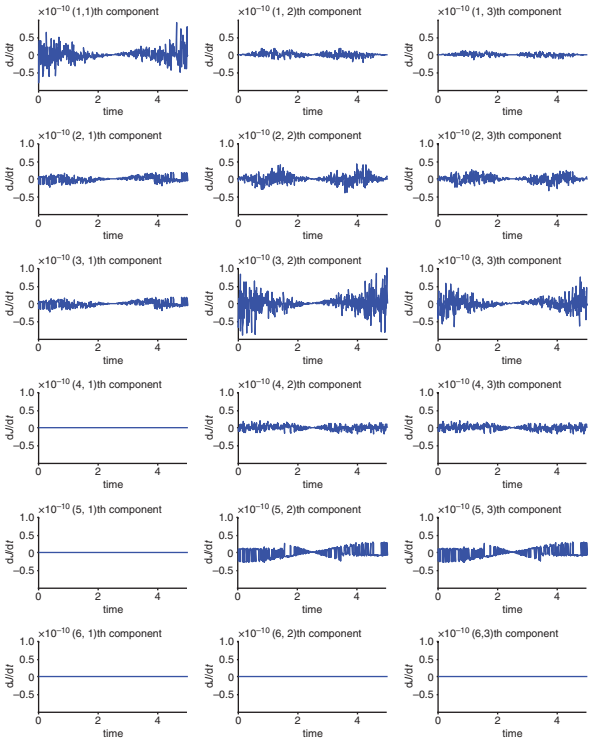

Fig. 2 *Difference between numerical values and analytical values*

Simulation: To verify the proposed method's effectiveness, computer simulations were examined by using a three-link robot. Its Denavit–Hartenberg (DH) parameters are described in Table 2.

The angular position and velocity of the joint variables in the simulation were generated as sinusoidal functions. Using these trajectories, both numerical and analytical methods were used to calculate results. The chain rule was used to calculate the numerical differentiation as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{J}_{ij}=\sum_{k=1}^{n}\left(\frac{\partial}{\partial q_k}\boldsymbol{J}_{ij}\right)\dot{q}_k\tag{10}$$

Partial differentiation of (10) is numerically calculated with the two-point formula, i.e. $(\partial/\partial x)f(x,\,y)\approx\left(\left(f(x+h,\,y)-f(x-h,\,y)\right)/2h\right)$, where h is a small displacement of the independent variable and 10^{-8} is adopted in this simulation. Sampling was performed every 10 ms and the simulation was carried out over 5 s.

In Fig. 2, differences between the numerical and the analytical results are illustrated. As shown in this figure, the differences remain small, implying that the results of both methods are accurate enough to serve as a control reference. The analytical method is definitely more accurate than the numerical one because it does not use approximations. To determine the computational cost, more simulations were performed and the joint complexity was increased from three joints to seven joints. Both methods were tested using MATLAB on Intel Xeon core. As shown in Fig. 3, the analytical method is almost seven times faster than the numerical method for all cases. As the computation increases in proportion to the number of joints, the computational time was also linearly increased as expected.

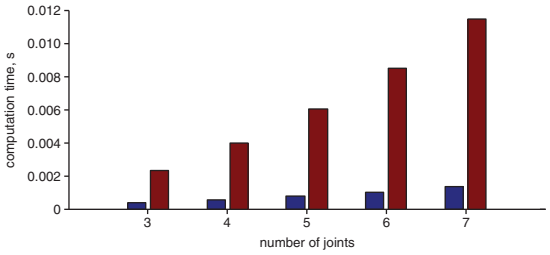

Fig. 3 *Comparison of computation times between proposed analytical method and numerical method*

Conclusion: Recent control algorithms require more and more information about the system model and the control reference to improve performance. To meet this requirement, the differentiation of the Jacobian matrix should be calculated within the control interval. In this Letter, to increase computational efficiency, an analytical method of calculation was formulated. Its effectiveness was demonstrated in comparison with the conventional method, i.e. numerical differentiation. As determined through a computer simulation, the proposed method seems to be almost seven times faster than the numerical method, though it does not have any approximations.

Acknowledgments: This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (no. 2015R1C1A1A02037641).

Submitted: *13 October 2016* E-first: *13 February 2017*
doi: 10.1049/el.2016.3776
One or more of the Figures in this Letter are available in colour online.

J.-Y. Rhee and B. Lee (*Department of Electrical Engineering, Myongji University, San 38-2 Namdong, Yongin 449-728, Gyeonggi-do, South Korea*)

✉ E-mail: bjlee@mju.ac.kr

References

1 Spong, M.W., Hutchinson, S., and Vidyasaga, M.: 'Robot modeling and control' (John Wiley & Sons, Inc., 2005)
2 Kuindersma, S., and Tedrake, R.: 'An efficiently solvable quadratic program for stabilizing dynamic locomotion'. 2014 IEEE Int. Conf. on Robotics and Automation (ICRA), Hong Kong, China, May 2014
3 Tedrake, R., Kuindersma, R., Delts, R., *et al.*: 'A closed-form solution for real-time ZMP gait generation and feedback stabilization'. 2015 IEEE-RAS 15th Int. Conf. on Humanoid Robots (Humanoids), Seoul, Republic of Korea, November 2015
4 Lewis, F., Dawson, D., and Abdullah, C.: 'Robot manipulator control: theory and practice' (CRC Press, Boca Raton, FL, USA, 2003)
5 Wang, J., Li, Y., and Zhao, X.: 'Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm', *Int. J. Adv. Robot. Syst.*, 2010, **7**, (4), pp. 1–9
6 Simaan, N., and Shoham, M.: 'Geometric interpretation of the derivatives of parallel robots' Jacobian matrix with application to stiffness control', *J. Mech. Des.*, 2003, **125**, (1), pp. 33–42