Published in final edited form as:

Adv Eng Softw. 2012 May ; 47(1): 160–163. doi:10.1016/j.advengsoft.2012.01.002.

Computational Development of Jacobian Matrices for Complex Spatial Manipulators

Craig M. Goehler^{a,b,*} and Wendy M. Murray^{b,c,d}

^aDepartment of Mechanical Engineering, Valparaiso University, Valparaiso, IN USA

^bSensory Motor Performance Program, Rehabilitation Institute of Chicago, Chicago, IL, USA

^cDepartments of Biomedical Engineering and Physical Medicine & Rehabilitation, Northwestern University, Chicago, IL, USA

^dResearch Service, Edward Hines Jr., VA Hospital, Hines, IL, USA

Abstract

Current methods for developing manipulator Jacobian matrices are based on traditional kinematic descriptions such as Denavit and Hartenberg parameters. The resulting symbolic equations for these matrices become cumbersome and computationally inefficient when dealing with more complex spatial manipulators, such as those seen in the field of biomechanics. This paper develops a modified method for Jacobian development based on generalized kinematic equations that incorporates partial derivatives of matrices with Leibniz's Law (the product rule). It is shown that a set of symbolic matrix functions can be derived that improve computational efficiency when used in MATLAB® M-Files and are applicable to any spatial manipulator. An articulated arm subassembly and a musculoskeletal model of the hand are used as examples.

Keywords

Jacobian matrix; spatial manipulator; MATLAB®; Leibniz's Law

1 Introduction

An important component of engineering analysis is the ability to quickly and efficiently study the kinematics of various spatial manipulators using software packages such as MATLAB®. Specifically, the level of programming difficulty during the development and implementation of the manipulator Jacobian matrix varies depending on the complexity of the considered system. This becomes particularly apparent in the field of biomechanics, where the system to be studied (the human body) does not always easily conform to traditional mechanical analysis techniques.

One popular method for analyzing manipulator kinematics is the application of Denavit and Hartenberg parameters [1]. While this method has been primarily used for the development

© 2012 Elsevier Ltd. All rights reserved.

*Corresponding Author. Phone: 1-219-464-6811; Fax: 1-219-464-5065. Craig.Goehler@valpo.edu (C.M. Goehler), w-murray@northwestern.edu (W.M. Murray).

Publisher's Disclaimer: This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final citable form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

of Jacobian matrices for man-made systems such as traditional robotic manipulators [2] [3] [4], it has also been utilized to describe the motion of different aspects of the human body, such as the upper extremity [5]. The use of these parameters is most efficient when considering systems where the axes of joint rotations are intersecting and orthogonal to one another, as is the case in most man-made systems. The human body possesses many systems of joints whose axes of rotation are both non-intersecting and non-orthogonal, such as the wrist and digits of the hand. This is why more generalized descriptions of manipulator kinematics are starting to be used for human motion analysis [6]. Using standard techniques, these generalized kinematic equations lead to much more complex and computationally inefficient Jacobian matrices that result in longer program completion times.

The main purpose of this document is to outline a modified procedure for the development of the manipulator Jacobian matrix using generalized kinematic equations. This procedure will combine the ability to take partial derivatives of the homogeneous transformation matrices [2] with Leibniz's Law (product rule) [7] [8]. The resulting symbolic equations are easily implemented into a MATLAB® program and lead to much shorter run-times than using either numerical techniques or the symbolic equations from the standard method of Jacobian development.

2 Calculation

2.1 Manipulator Kinematics

The standard technique in robotics and mechanism analysis for defining manipulator kinematics begins with assigning numbers to each moving body of the desired manipulator and then defining coordinate systems that are fixed to each of these bodies during motion [1]. Each pair of consecutive coordinate systems (i.e. body frames {1} and {2}) can be related mathematically using a set of homogeneous transformation matrices. A separate matrix is developed to relate each set of consecutive body frames from the base frame, {0}, out to the distal frame on the end-effector, {n}. For this document, these matrices are presented in a generalized form in order to facilitate application to any manipulator. Each matrix is comprised of the equivalent angle-axis rotation matrix along with the xyz components of translation between coordinate systems,

$${}_{i-1}^iT = \begin{bmatrix} k_x^2(1 - c\theta) + c\theta & k_xk_y(1 - c\theta) - k_zs\theta & k_xk_z(1 - c\theta) + k_ys\theta & t_x \\ k_xk_y(1 - c\theta) + k_zs\theta & k_y^2(1 - c\theta) + c\theta & k_yk_z(1 - c\theta) - k_xs\theta & t_y \\ k_xk_z(1 - c\theta) - k_ys\theta & k_yk_z(1 - c\theta) + k_xs\theta & k_z^2(1 - c\theta) + c\theta & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1)$$

where $i - 1$ and i represent the pair of consecutive body frames, θ is the angle of rotation, t_x , t_y and t_z represent the xyz components of translation, and $\hat{k} = [k_x, k_y, k_z]^T$ is the axis of rotation (a unit vector). The shorthand notation $c\theta = \cos\theta$ and $s\theta = \sin\theta$ is used throughout this document. Depending on the type of joint between bodies, each matrix will be a function of rotation, translation or a combination of both, resulting in either 1-DOF or 2-DOF (degrees of freedom). The derivation of the system of homogeneous transformation matrices along with the respective degrees of freedom completes the kinematic description of the desired manipulator.

2.2 Jacobian Development

2.2.1 Existing Method—The equations derived in Section 2.1 can be used to instantaneously relate the individual joint velocities to the Cartesian linear and angular velocities of a point of interest on the end-effector [1]. This relationship can be mathematically described using a Jacobian matrix.

The standard technique for deriving the components of the manipulator Jacobian relating the joint velocities to the Cartesian linear velocities starts with identifying a point of interest relative to the most distal coordinate system of the end-effector,

$${}^n P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

(*N.B.* The components of the Jacobian relating the joint velocities to the Cartesian angular velocities can be derived using other methods [2] [4].) The coordinates for this point of interest are then transformed into the base frame through the homogeneous transformation matrices that are derived using Eq. 1,

$${}^0 P = {}^0 T_1 {}^1 T_2 \dots {}^{n-1} T_n {}^n P. \quad (2)$$

The equations for the x , y , and z components of the point of interest in the base frame are then differentiated with respect to each degree of freedom in order to populate the desired Jacobian matrix,

$$J = \begin{bmatrix} \frac{\partial P_x}{\partial \theta_1} & \frac{\partial P_x}{\partial \theta_2} & \dots & \frac{\partial P_x}{\partial \theta_n} \\ \frac{\partial P_y}{\partial \theta_1} & \frac{\partial P_y}{\partial \theta_2} & \dots & \frac{\partial P_y}{\partial \theta_n} \\ \frac{\partial P_z}{\partial \theta_1} & \frac{\partial P_z}{\partial \theta_2} & \dots & \frac{\partial P_z}{\partial \theta_n} \end{bmatrix}. \quad (3)$$

As noted earlier, this method is desirable for manipulators with axes of rotation that are orthogonal and intersecting. For such systems, the resulting symbolic Jacobian matrix typically simplifies nicely and is easily implemented into a standard MATLAB® M-File. However, when dealing with systems comprised of non-intersecting, non-orthogonal axes of rotation, e.g. the human arm, the standard development technique results in excessively large symbolic Jacobian matrices that are not easily simplified and that increase MATLAB® computation time by several orders of magnitude. In order to avoid this computational issue, the following procedure was developed using a combination of a well-known mathematical principle and linear algebra.

2.2.2 Modified Method—Recognizing that the total transformation from the distal frame to the base frame can be broken into individual transformation matrices that are functions of the individual joint angles (note that this can be similarly derived where a translation is the degree of freedom), we can apply Leibniz's Law to the transformation equation and arrive at both the same numerical Jacobian matrix and a more computationally efficient symbolic Jacobian matrix. Eq. 2 can be rewritten as,

$${}^0 P = {}^0 T_1(\theta_1) {}^1 T_2(\theta_2) \dots {}^{n-1} T_n(\theta_n) {}^n P, \quad (4)$$

where each homogeneous transformation matrix is defined as a function of a single degree of freedom (the joint angle). It is possible to take the partial derivative of each matrix with respect to each degree of freedom [9]. Using the product rule on Eq. 4 along with the fact that each degree of freedom only appears in a single matrix, the equations for each column of the Jacobian matrix can be derived rather than calculating the individual elements (*cf.* Eq. 3),

$$\begin{aligned}
J(:, 1) &= \frac{\partial^0 T(\theta_1)}{\partial \theta_1} {}^1_2 T(\theta_2) \dots {}^{n-1}_n T(\theta_n) {}^n P \\
J(:, 2) &= {}^0_1 T(\theta_1) \frac{\partial^1 T(\theta_2)}{\partial \theta_2} \dots {}^{n-1}_n T(\theta_n) {}^n P \\
&\vdots \\
J(:, n) &= {}^0_1 T(\theta_1) {}^1_2 T(\theta_2) \dots \frac{\partial^{n-1} T(\theta_n)}{\partial \theta_n} {}^n P,
\end{aligned} \tag{5}$$

which results in the following matrix,

$$J = \begin{bmatrix} J(:, 1) & J(:, 2) & \dots & J(:, n) \end{bmatrix}. \tag{6}$$

This method allows the programmer to input the symbolic homogeneous transformation matrices along with the symbolic partial derivatives of each matrix and solve for the desired Jacobian matrix using Eqs. 5 & 6. This process results in a drastic reduction in MATLAB[®] computation time compared to either using the symbolic Jacobian from Eq. 3 or trying to differentiate the original kinematic equations in Eq. 2 using built-in MATLAB[®] functions. Numerically, the Jacobian matrix calculated using Eq. 6 is identical to the matrix calculated using Eq. 3.

3 Results and Discussion

The benefits of the modified method outlined in Section 2.2.2 will be highlighted using two different examples. The first example will examine an articulated arm subassembly, a standard robotic manipulator. The purpose of this example will be to illustrate how both the standard method and the modified method result in the same symbolic Jacobian matrix for a simple 3-DOF system. The second example will concentrate on a 6-DOF musculoskeletal model of the hand that is used to study the effects of wrist motion on thumb-tip pinch force. The purpose of this example will be to demonstrate the improved computational efficiency of the modified method using a complex physical system. A single set of numerical values will also be used to show that the numerical Jacobian matrix will instantaneously be the same using either method.

3.1 Example 1: Articulated Arm Subassembly

This example will use previously developed kinematic equations for a 3-DOF articulated arm subassembly [3]. Using the method outlined in Section 2.2.1, the manipulator Jacobian matrix was developed,

$$J = \begin{bmatrix} -zc\theta_1 - As\theta_1 & -c\theta_1 (a_2 s\theta_2 + B) & -Bc\theta_1 \\ -zs\theta_1 + Ac\theta_1 & -s\theta_1 (a_2 s\theta_2 + B) & -Bs\theta_1 \\ 0 & -A & -C \end{bmatrix}, \tag{7}$$

where

$$A = a_2 c\theta_2 + xc\theta_{23} - ys\theta_{23}, \quad B = xs\theta_{23} + yc\theta_{23}, \quad C = xc\theta_{23} - ys\theta_{23}.$$

Following the method outlined in Section 2.2.2, the symbolic Denavit-Hartenberg Transformation matrices [1] can be differentiated with respect to the individual degrees of freedom for the robotic manipulator. In this example, all three of the degrees of freedom are joint angles: θ_1 , θ_2 , & θ_3 . Therefore, it is more computationally efficient to utilize a generalized partial derivative with respect to θ_i ,

$$\frac{\partial^{i-1} T(\theta_i)}{\partial \theta_i} = \begin{bmatrix} -s\theta_i & -c\theta_i & 0 & 0 \\ c\theta_i c\alpha_{i-1} & -s\theta_i c\alpha_{i-1} & 0 & 0 \\ c\theta_i s\alpha_{i-1} & -s\theta_i s\alpha_{i-1} & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (8)$$

Due to the simplicity of the robotic manipulator, it is feasible to report the symbolic equations for the columns of the Jacobian matrix that are determined using Eq. 5 along with the differentiated homogeneous transformation matrices from Eq.8,

$$J(:, 1) = \begin{bmatrix} -zc\theta_1 - s\theta_1(a_2c\theta_2 + xc\theta_{23} - ys\theta_{23}) \\ -zs\theta_1 + c\theta_1(a_2c\theta_2 + xc\theta_{23} - ys\theta_{23}) \\ 0 \end{bmatrix}, \quad (9)$$

$$J(:, 2) = \begin{bmatrix} -c\theta_1(a_2s\theta_2 + xs\theta_{23} + yc\theta_{23}) \\ -s\theta_1(a_2s\theta_2 + xs\theta_{23} - yc\theta_{23}) \\ -a_2c\theta_2 - xc\theta_{23} + ys\theta_{23} \end{bmatrix}, \quad (10)$$

$$J(:, 3) = \begin{bmatrix} -c\theta_1(xs\theta_{23} + yc\theta_{23}) \\ -s\theta_1(xs\theta_{23} + yc\theta_{23}) \\ -xc\theta_{23} + ys\theta_{23} \end{bmatrix}. \quad (11)$$

The symbolic Jacobian matrix that is composed of the column vectors from Eqs. 9, 10 & 11 is identical to the symbolic Jacobian matrix in Eq. 7. This result demonstrates how the standard and modified methods for developing the manipulator Jacobian matrix can be used interchangeably for systems that are kinematically described using Denavit and Hartenberg parameters or similar analysis techniques.

3.2 Example 2: Musculoskeletal Hand Model

The second example considers a musculoskeletal model of the hand that adds the contributions of wrist movement (flexion and deviation) to an existing 4-DOF model of the thumb [10]. This augmentation actually adds four separate rotations about four non-intersecting, non-orthogonal axes of rotation resulting in a 3×8 manipulator Jacobian matrix. Due to coupled bone movements in the wrist, two of these rotations are dependent on the other two independent joint rotations. Therefore, there is an addition of only 2-DOF to the total system.

Applying the method outlined in Section 2.2.1 to this model resulted in an extremely large symbolic Jacobian matrix where individual elements reached 145 pages in length, even after being simplified using the *Mathematica* software package. Equations of that length are obviously unacceptable when trying to run computer simulations that include thousands of combinations of manipulator configurations. However, it was still possible to numerically evaluate this matrix at instantaneous manipulator configurations. Using a sample posture with neutral wrist flexion and deviation (all four wrist joint angles = 0°), 20° of carpometacarpal extension, 25° of carpometacarpal abduction, 45° of metacarpophalangeal flexion and 10° of interphalangeal flexion, the manipulator Jacobian matrix was numerically calculated (rounded for clarity),

$$J = \begin{bmatrix} -0.053 & 0.021 & -0.013 & -0.007 & 0.0737 & 0.0285 & 0.040 & 0.016 \\ -0.046 & 0.031 & 0.015 & 0.025 & 0.017 & 0.011 & -0.016 & -0.009 \\ 0.089 & -0.086 & -0.064 & -0.086 & 0.020 & -0.027 & 0.029 & 0.013 \end{bmatrix}. \quad (12)$$

The method outlined in Section 2.2.2 provides the programmer with a set of symbolic matrix functions that can be easily implemented into a MATLAB® M-File. The first matrix function is the generalized homogeneous transformation matrix in Eq. 1. The other function comes from differentiating Eq. 1,

$$\frac{\partial^{i-1} T(\theta_i)}{\partial \theta_i} = \begin{bmatrix} k_x^2 s\theta_i - s\theta_i & k_x k_y s\theta_i - k_z c\theta_i & k_x k_z s\theta_i + k_y c\theta_i & 0 \\ k_x k_y s\theta_i + k_z c\theta_i & k_y^2 s\theta_i - s\theta_i & k_y k_z s\theta_i - k_x c\theta_i & 0 \\ k_x k_z s\theta_i - k_y c\theta_i & k_y k_z s\theta_i + k_x c\theta_i & k_z^2 s\theta_i - s\theta_i & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (13)$$

The columns of the Jacobian matrix can be determined using Eq. 5 along with the differentiated homogeneous transformation matrices from Eq.13. The entire computer function for the resulting Jacobian matrix is now less than a single page in length and is much more manageable when iterating through all of the possible combinations of thumb configurations. When executed for the single set of numerical values defined previously, this new MATLAB® function runs to completion instantly and outputs a Jacobian matrix identical to the one reported in Eq. 12. This example demonstrates how the modified method for developing the manipulator Jacobian matrix is computationally more efficient than the standard method for complex manipulators and yet both methods result in the same numerical Jacobian matrix.

3.3 Conclusion

This document has introduced a modified method for developing the manipulator Jacobian matrix using generalized kinematic equations with the purpose of improving programming efficiency for complex spatial manipulators. This procedure utilizes various mathematical principles in order to create a set of symbolic functions that are easily implemented into MATLAB® M-Files. Two examples of varying kinematic complexity were presented that demonstrate improved computational efficiency over the standard method while maintaining mathematical accuracy.

- Product rule can be applied to homogeneous transformation matrices to compute manipulator Jacobian matrices.
- Use of equivalent angle-axis rotation matrices allows application of proposed method to any manipulator.
- Proposed method is easily implemented within MATLAB.

Acknowledgments

This work was funded by NIH R01 HD046774.

References

1. Craig, JJ. Introduction to Robotics: Mechanics and Control, Second Edition. New York: Addison-Wesley Publishing Company, Inc.; 1989.

2. Goldenberg AA, Benhabib B, Fenton RG. A Complete Generalized Solution to the Inverse Kinematics of Robots. *IEEE Journal of Robotics and Automation*. 1985 March; Vol. 1(No. 1):14–20.
3. Stanisic MM, Goehler CM. Singular planes of serial wrist-partitioned manipulators and their singularity metrics. *Mechanism and Machine Theory*. 2007; Vol. 42:889–902.
4. Tourassis VD, Ang MH. Identification and Analysis of Robot Manipulator Singularities. *International Journal of Robotics Research*. 1992; Vol. 11(No. 3):248–259.
5. Yang J, Abdel-Malek K, Nebel K. Reach Envelope of a 9-Degree-of-Freedom Model of the Upper Extremity. *International Journal of Robotics and Automation*. 2005; Vol. 20(No. 4):248–259.
6. Khatib O, Demircan E, De Sapia V, Sentis L, Besier T, Delp S. Robotics-based synthesis of human motion. *Journal of Physiology - Paris*. 2009; Vol. 103:211–219.
7. Leibniz G. *Mathematics Schriften*. 1858; Vol. 1(Part 2):377–382.
8. Osler TJ. A Further Extension of the Leibniz Rule to Fractional Derivatives and Its Relation to Parseval's Formula. *SIAM J. Math. Anal.* 1972 February; Vol. 3(No. 1):211–219.
9. Vetter WJ. Derivative Operations on Matrices. *IEEE Transactions on Automatic Control*. 1970; Vol. 15(No. 2):241–244.
10. Goehler CM, Murray WM. The sensitivity of endpoint forces produced by the extrinsic muscles of the thumb to posture. *Journal of Biomechanics*. 2010; Vol. 43:1553–1559. [PubMed: 20303085]