Journal of Mechanics Engineering and Automation 5 (2015) 503-509
doi: 10.17265/2159-5275/2015.09.004

Kinematic Model Extraction for Serial Manipulators Which Have in Different Topologies in Vpython Environment

Ece Yıldırım
Department of Mechanical Engineering, Gazi University, Ankara 06500, Turkey

Abstract: In this study, we aim at obtaining inverse kinematic model of a serial manipulator using spatial operator algebra. For testing the inverse kinematic algorithm, the Vpython software program which has simultaneous view and software working, is used. The aim is to measure the inverse kinematics modeling work on different serial manipulator mechanisms with spatial vector algebra. The algorithm is used with the same reference inputs on the recursive, exact and nonrecursive manipulators. During the tests, the permitted error tolerance is 0.01 cm. The graph plots show that the algorithm is fit for the error tolerance.

Key words: Spatial operator algebra, inverse kinematic, Vpython.

1. Introduction

In this study, by using spatial operator algebra, the inverse kinematical analysis of exact, redundant and nonredundant manipulators with the structure of serial topological is made. Analytical methods related to inverse kinematic solution can be resolved as symbolic or numeric. The great disadvantage of symbolic solution comes into existence in the case of the manipulator DOF (degree of freedom) increase. Particularly, statements that become more complex in differential operation come into being. In this sense, it is known that numeric Jacobian matrix is in practice. Obtaining Jacobian matrix with the methods known in literature and used prevalently such as Denavit-Hertenberg [1] is not easy, especially for manipulators which have high DOF. In spatial operator algebra, we get Jacobian matrix numerically with a systematic and easy programmable method.

When examining the usage of spatial operator algebra in literature, it takes part in literature with the works done by Rodriguez and his team [2, 3] in JPL (Jet Propulsion Laboratory) which is a center of NASA. When examined generally, this method's basis is known as "screw theory" [4]. Starting from this basis, Featerstone [5] and Angeles [6]'s studies take place in literature. Implementation of spatial operator algebra to various fields is maintained by mainly in JPL Jain and his team [7-9] that study in this field. By using this theory, Yeşiloğlu [10] develops appropriate dynamic modeling for using various control methods on loads which are transported by cooperation manipulators. As a contribution to spatial operator algebra theory, Yeşiloğlu [11] solves the force distribution problem in the case of kinematic inadequate manipulators taking place in closed cycle topology.

Vpython used in the part of software is a programming language which Scherer and his team [12] add three-dimensional visual library to Python in 1991. In 1990, Python [13] has been developed by Guido van Rossum in Amsterdam. Recently, it is continued to be developed by the volunteers who become members at Python software foundation. It is object oriented and a modular programming language. The modular structure supports both the class string (system) and all kinds of data entry area. It can work in almost any platform.

Corresponding author: Ece Yıldırım, M.Sc., researcher, research field: kinematic modelling of manipulators. E-mail: ece.cosgun@gazi.edu.tr.

When examining the usage of Vpython in literature, Sands [14] mentions Vpython's algorithm structure is vector oriented and explains that it can be used in mechanism technique. In their study, Wochal et al. [15] make a visualized form of a mobile robot in Czestochowa and actualize control via Vpython program. Practically, usage of Python in literature is much more. Pedrogosa and his team [16] actualize a machine learning program by using Python. In their study, Cock and his team [17] introduce Biopython and tell there is a module of Python for computational molecular biology.

2. The Kinematic Modeling with Spatial Operator Algebra

Spatial operators are 6D (six dimensional) vectors including 3D angular and 3D linear velocity operators. Modeling are calculated in the way that these velocity operators' distributions from fixed or moving platform to gripper. In this method, the symbols which are used to calculate kinematic modelling, are shown in Fig. 1.

Fig. 1 shows that any two joints are attached with a link.

The joint types can be prismatic or rotary although whole calculations are the same. The used symbols represent spatial axis vector, joint velocity operator, joint torque and joint force for ${}_{k+1}^{i}\vec{h}$, ${}_{k+1}^{i}\theta$, ${}_{k+1}^{i}\vec{\tau}$, ${}_{k+1}^{i}\vec{f}$, respectively. It will explain the calculation of each symbols at following equations. τ and f are operated at dynamical modelling. Therefore, in this study, those are not used.

$$\vec{\vec{V}} = \begin{bmatrix} \vec{\omega}_a \\ \vec{V}_a \end{bmatrix}_{(6\times1)} \tag{1}$$

$$\vec{\omega}_a = \vec{\omega}_{a-1} + \vec{h}_a \dot{\theta}_a \tag{2}$$

$$\vec{V}_a = \vec{V}_{a-1} + \vec{\omega}_{a-1} \times \vec{l}_{a-1,a} \tag{3}$$

Double arrow above the symbol that points the general velocity operator in Eq. (1) indicates that velocity is in the size of (6 × 1). Through the whole calculation, double arrows above the symbol shows that the symbol used is in this size. In spatial operator

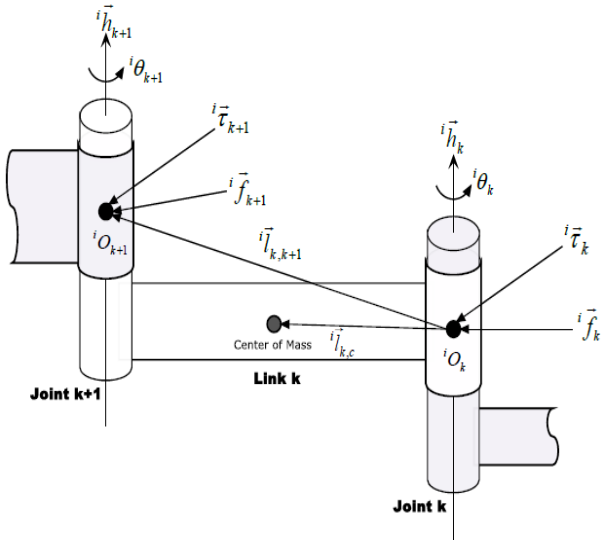

Fig. 1 Vectors associated with link *k* of the manipulator *i* [18].

algebra, it is required to obtain the three basic matrix structures defining this modeling in order to reach the velocity data of link velocity and gripper. These matrix structures are evinced between Eq. (4) and Eq. (6).

Eq. (4) is the link propagation matrix including the data of link's length related to each other. Symbol above the statement $\hat{l}_{a,b}$ is negative (skew) symmetrical matrix of the operator. As shown in Fig. 1, $l_{a,b}$ is 3D distance which is related from the center of link *a* to the center of link *b*.

$$\varphi_{b,a} = \begin{bmatrix} I_{3\times3} & 0_{3\times3} \\ -\hat{l}_{a,b} & I_{3\times3} \end{bmatrix} \tag{4}$$

Eq. (5) is the rotation or translation spatial axis operator including the rotation or translation data of the links. Here, first (3 × 1) part contains the necessary data for rotary joint type; last (3 × 1) part contains the necessary data for prismatic joint type. This operator is updated with multiplication of rotation matrix updated depending on operator joints' move and rotation axis.

$$\vec{\vec{h}} = \begin{bmatrix} \vec{h}_\omega \\ \vec{h}_V \end{bmatrix}_{(6\times1)} \tag{5}$$

Rotation matrix used here is derived from Rodriguez formula stated in Eq. 6. Only stating rotation axis data entered in operator formate and rotation angle is

sufficient to calculate rotation matrix with this formula. Rotation axis can be demonstrated in 3D axis assembly as in Fig. 2. ω sign used in here is just a symbol defining 3D rotation axis. θ is the rotation angle of any axis.

$$R = e^{\hat{\omega}\theta} = I + sin\theta\hat{\omega} + (1 - cos\theta)\hat{\omega}^2 \quad (6)$$

Eq. (7) is joint velocity operator containing the data of joints' angle and linear velocity. In this equation, n states manipulator definition of freedom. In our study, all velocities entered as rotational velocity due to all joints are rotational.Units are provided as in radians/s.

$$\underline{\dot{\theta}} = \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \vdots \\ \vdots \end{bmatrix}_{(n\times 1)} \quad (7)$$

After putting the data belonging to the links into operator assemblies, generally, propagation matrix calculus for the recursive structure of spatial operator algebra and high DOF manipulators is acquired as in Eq. (8). As seen in Eq. (8), the matrix has just the dimensional information about links at bottom right triangle.

$$\emptyset = \begin{bmatrix} I_{6\times6} & 0_{6\times6} & 0_{6\times6} & \dots & 0_{6\times6} \\ \emptyset_{21} & I_{6\times6} & 0_{6\times6} & 0_{6\times6} & \vdots \\ \emptyset_{31} & \emptyset_{32} & \ddots & \vdots & \vdots \\ \emptyset_{41} & \emptyset_{42} & \emptyset_{43} & \ddots & \vdots \\ \vdots & \vdots & \vdots & \emptyset_{54} & I_{6\times6} \end{bmatrix}_{(6\times n,6\times n)} \quad (8)$$

The statement of $\emptyset_{a,c}$ in Eq. (8) is calculated as in Eq. (9). Eqs. (8) and (9) show that algorithms are appropriate for coding vigorously in software. This situation leads to work faster of software. Besides, due to the drawability of joints' positions relative to each other over the course of software, whether the manipulator transiently goes into singularity can be controlled. Anything has been made to prevent the singularity, which is completely subject for further study.

$$\emptyset_{a,c} = \emptyset_{a,b}\emptyset_{b,c} \quad (9)$$

General rotation axes matrix is acquired as in Eq. (10). In the study, the joints' DOF is taken as 1 and calculations are made according to this. Calculations

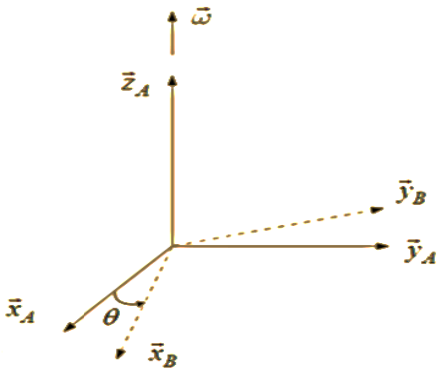

Fig. 2 Rotational axes in spatial operator analyse.

and formulas do not change even in the case of joints' DOF is different from 1.

$$H = \begin{bmatrix} \vec{h}_1 & \vec{0} & \vec{0} & \dots \\ \vec{0} & \vec{h}_2 & \vdots & \vdots \\ \vec{0} & \vec{0} & \ddots & \vdots \\ \vdots & \vdots & \vec{0} & \ddots \end{bmatrix}_{(6\times n,n)} \quad (10)$$

Spatial velocity operator containing joints' velocity data is calculated in Eq. (11). The velocity in here is at the size of (6n, 1). Hereby, angular and linear velocity at the joints in the course of software is obtained simultaneous without having to make any calculations. Like Denavit-Hartenberg [1] which is popular at kinematic modelling permits to reach just the position of joints and links. Also, at this method if the velocity of the links is needed, Jacobian matrix has to be calculated.

$$\underline{V} = \emptyset H\underline{\dot{\theta}} \quad (11)$$

The velocity matrix of the end effector is calculated with multiplication joint velocity and the propogotion matrix of end effector's 3D distances. The propogotion matrix of the end effector is calculated in Eq. (12).

$$\emptyset_t = [0_{6\times6} \quad 0_{6\times6} \quad \dots \quad \dots \quad \dots \quad \varphi_n]_{(6,6\times n)} \quad (12)$$

The velocity matrix of the end effector can be obtained with multiplication vectors and matrixes are calculated in advance. This matrix is seen in Eq. (13).

$$\vec{V}_t = \emptyset_t\emptyset H\underline{\dot{\theta}} \quad (13)$$

The velocity of the end effector in closed form is

calculated with multiplication Jacobian matrix that provides relation between cartesian space, joint space and joint velocities. This state can be seen in Eq. (14). Eqs. (13) and (14) are examined that Jacobian matrix is formed by multiplying the previously calculated matrix. Therefore, this matrix can be obtained in a numerical and systematic way. Algorithm that reduces the computational load (real-time) spatial vector algebra is well suited to the in real-time work. Jacobian matrix is calculated as shown in Eq. (15).

$$\vec{V}_t = J\dot{\underline{\theta}} \tag{14}$$

$$J = \emptyset_t \emptyset H \tag{15}$$

In the kinematic study, it is prompted to reach generally from the position and velocity of the end effector to the position and velocity of all joints. For this, the inverse of jacobian matrix and the velocity of the end effector must be multiplied each other. $J^{\#}$ is the pseudo inverse of the Jacobian matrix. If the dimension of Jacobian matrix is not square, the pseudo inverse matrix is used. For taking pseudo inverse operation, the least-squares method [19] is used. In our study, while the operation is not necessary for 6 DOF manipulator, it is essential for 4 and 8 DOF manipulators. At Python for taking the pseudo inverse of Jacobian matrix, the code particle which is added the program module is used. Since the system is dynamically encoded, Fully pseudo inverse is used regardless of DOF.

$$J^{\#}\vec{V}_t = \dot{\underline{\theta}} \tag{16}$$

The referance input is the desired 3D velocity of the end effector to calculate the joint angles. Afterwards, the required matematical calculations are operated up to reach the given referance background. As the zero position of the manipulators causes singularity during simulation and testing, the initial angles of the joints are taken 0.01 radian. Specified cycle continues until arriving desired error tolerance. The algoritm which is the kinematic model extraction of serial manipulator is shown in Fig. 3.

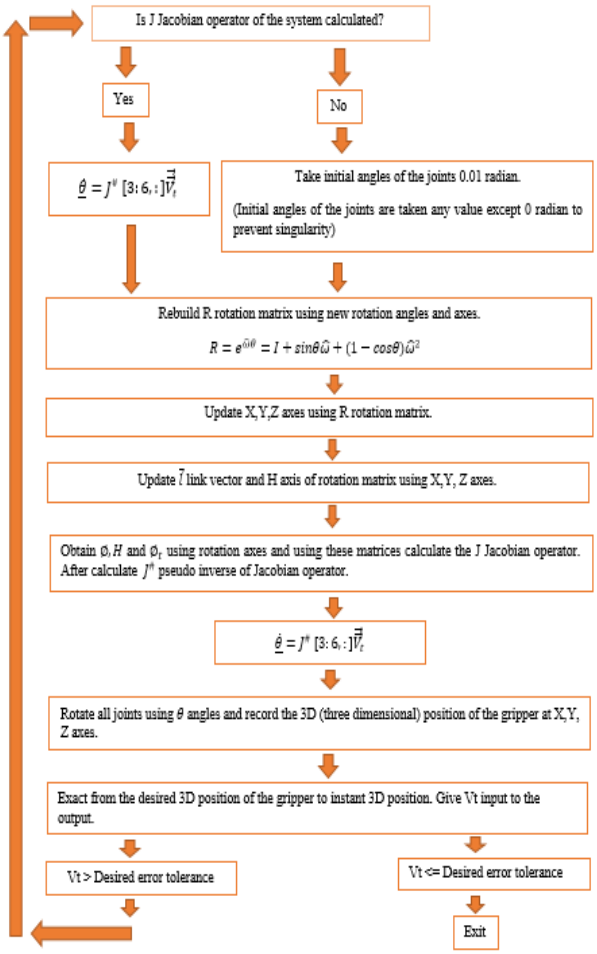

Fig. 3 The algoritm.

3. Simulation Results

Testing the inverse kinematic algorithm, it is concluded simulation studies on four, six and eight DOF manipulators. The same reference value is given to these three manipulators as a access for a real comparison. Access reference value is the position clicked on the screen via mouse by the user. In the program, 0.01 cm fault tolerance in the axis is allowed for the referance value. In Fig. 4, there are manipulator models used for simulation tests. Because of the fact that it is an academic study, in selecting the position of the manipulators' joints, mini extraction of the manipulator existing in the market does not need.

Fig. 5 shows approach of the six DOF manipulator to (1, 4.5, 4) m referance position with the inverse kinematic algorithm approach. Fig. 6 shows the

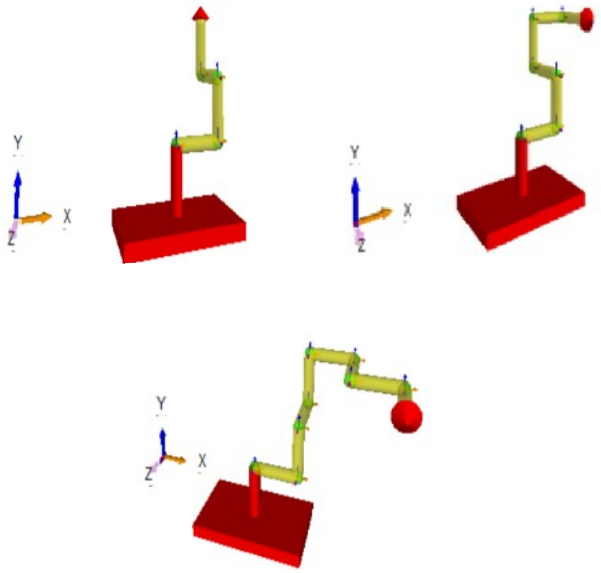

Fig. 4 Respectively, nonredundant, exact and redundant manipulators.

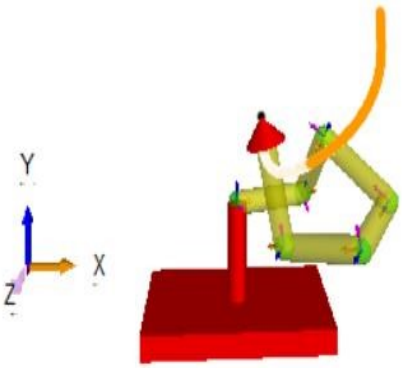

Fig. 5 Six dof manipulator's accession to (1, 4.5, 4) m position.

position and orientation data's graphic outputs of gripper during this trajectory. Because at the position that manipulator's gripper stand, gripper's orientation data on x, y, z coordinates is not asked. In the graphs, there are just the orientations on each axis at the moment of manipulators' orientation.

Fig. 7 shows approach of the eight dof manipulator which is redundant to the same referance position with the inverse kinematic algorithm approach. Fig. 8 shows the position and orientation data's graphic outputs of gripper during this trajectory.

Fig. 9 shows approach of the eight dof manipulator which is nonredundant to the same referance position with the inverse kinematic algorithm approach. Fig. 10

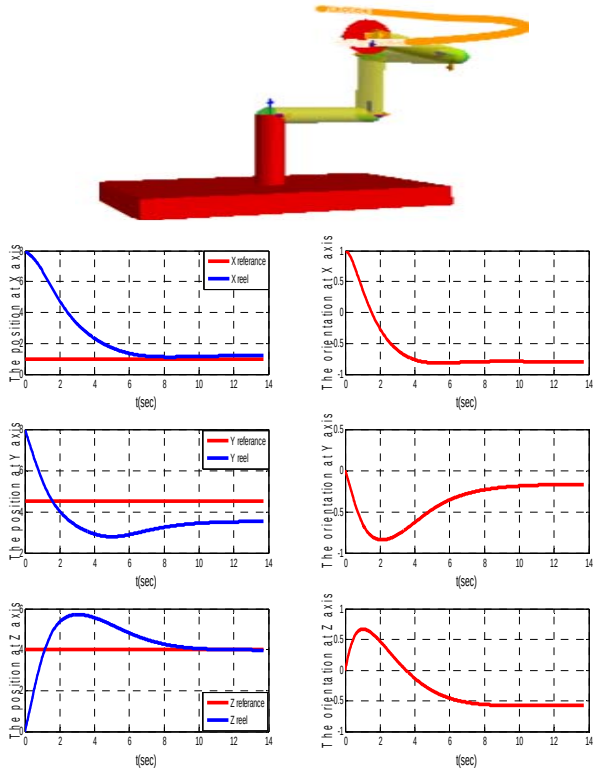

Fig. 6 Six dof manipulator's reference tracking graph.

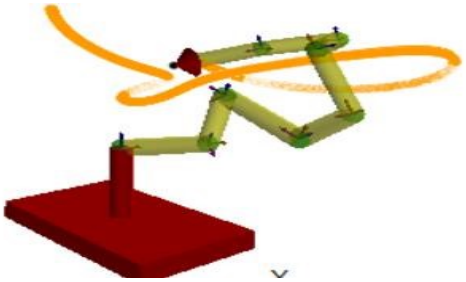

Fig. 7 Eight dof manipulator's accession to (1, 4.5, 4) m.

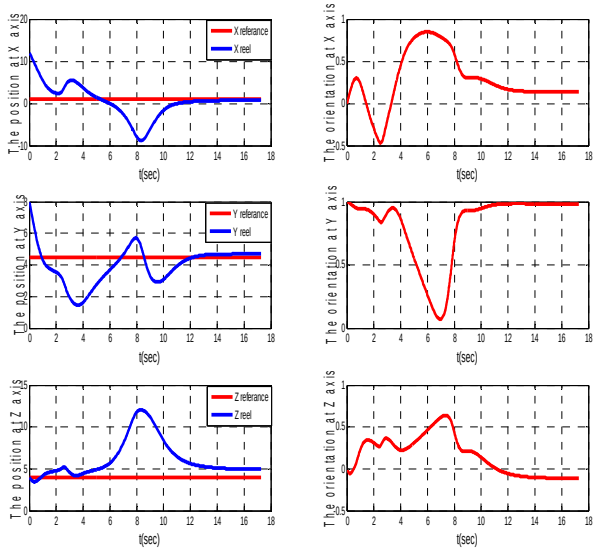

Fig. 8 Eight dof manipulator's reference tracking graph.

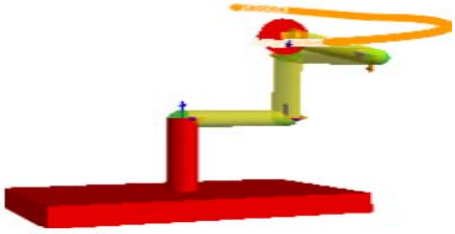

Fig. 9 Four dof manipulator's accession to (1, 4.5, 4) m.

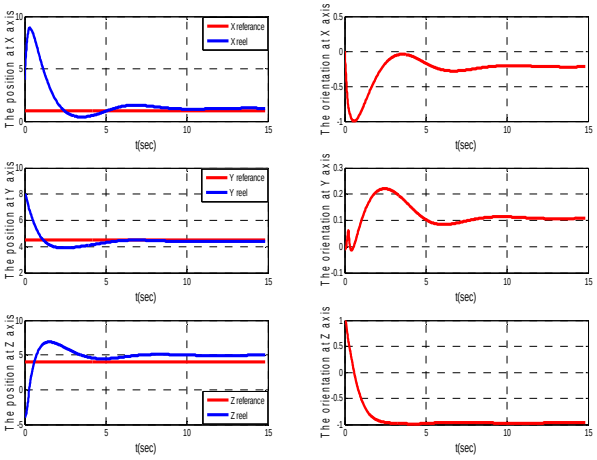

Fig. 10 4 DOF manipulator's reference tracking graph.

shows the position and orientation data's graphic outputs of gripper during this trajectory.

4. Conclusions

The study which is for testing the inverse kinematics algorithm of the spatial vector algebra is applied three basic topologic manipulators on fixed platform. During testing, all manipulators are given same referance inputs for logical comparison. Vpython which is used for simulations, has not a visual library for the convenience of users. Since it allows to work simultaneous view and software codes, it is suitable for real-time studies. At simulation tests conducted in several reference values, it is seen that the algoritm operates robust all manipulator kinds (nonredundant, exact and redundant manipulator) if the given referance input is in working space of the manipulators. In addition, as it can be seen from the graphs the manipulators which have same structure follow same trajectory no matter DOF. 8 DOF manipulator has arrived the target longer and following unnecessary path. It is caused that the axes of rotation are selected randomly. Consequently, the links and joints can prevent each other.

Acknowledgments

The author would like to gratefully thank Fatma Nur Dinçoğlu for the support of academic translation and also Oktay Yıldırım for his support which essentials to the development of this work.

References

[1] Hartenberg, R. S., and Denavit, J. 1964. *Kinematic Synthesis of Linkages.* New York: McGraw-Hill.

[2] Rodriguez, G., Jain, A., and Kreutz-Delgado, K. 1992. "Spatial Operator Algebra for Multibody System Dynamics." *Journal of the Astronautical Sciences* 40 (1): 27-50.

[3] Rodrigues, O. 1840. "Des Lois Géométriques qui Régissent les Déplacements d'un Système Solide dans l'espace: et de la Variation des Cordonnées Provenant de Ces Déplacements Considérés Indépendamment des Causes qui Peuvent les Produire." Publisher not identified.

[4] Ohwovoriole, M. S., and Roth, B. 1981. "An Extension of Screw Theory." *Journal of Mechanical Design* 103 (4): 725-35.

[5] Featherstone, R. 1983. "The Calculation of Robot Dynamics Using Articulated-Body Inertias." *The International Journal of Robotics Research* 2 (1): 13-30.

[6] Gosselin, C., and Angeles, J. 1990. "Singularity Analysis of Closed-Loop Kinematic Chains." *IEEE Transactions on Robotics and Automation* 6 (3): 281-90.

[7] Jain, A. 1991. "Unified Formulation of Dynamics for Serial Rigid Multibody Systems." *Journal of Guidance, Control, and Dynamics* 14 (3): 531-42.

[8] Jain, A., and Rodriguez, G. 1993. "An Analysis of the Kinematics and Dynamics of Underactuated Manipulators." *IEEE Transactions on Robotics and Automation* 9 (4): 411-22.

[9] Jain, A., and Rodriguez, G. 1995. "Diagonalized Lagrangian Robot Dynamics." *IEEE Transactions on Robotics and Automation* 11 (4): 571-84.

[10] Yeşiloğlu, S. M., and Temeltas, A. 2010. "Dynamic Modelling of Cooperation Underactuated Manipulators for Space Manipulation." *Advanced Topics* 3 (3): 325-41.

[11] Yeşiloğlu, S. M. 2007. "High Performance Dynamical Modelling of Complex Topology Systems." Ph.D. thesis, Istanbul Technical University.

[12] Scherer, D., Dubois, P., and Sherwood, B. 2000. "VPython: 3D İnteractive Scientific Graphics for Students." *Computing in Science & Engineering* 5: 56-62.

[13] van Rossum, G., and de Boer, J. 1991. "Interactively Testing Remote Servers Using the Python Programming Language." *CWi Quarterly* 4 (4)" 283-303.

[14] Sands, D. 2010. "First Year Mechanics Taught through Modelling in VPython." *New Directions* 6: 47-50.

[15] Wochal, M., Cekus, D., and Warys, P. 2012. "The Application of VPython to Visualization and Control of Robot." *Pomiary, Automatyka, Robotyka* 16 (12): 151-6.

[16] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. 2011. "Scikit-Learn: Machine Learning in Python." *The Journal of Machine Learning Research* 12: 2825-30.

[17] Cock, P. J. A., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. L. 2009. "Biopython: Freely Available Python Tools for Computational Molecular Biology and Bioinformatics." *Bioinformatics* 25 (11): 1422-3.

[18] Yıldız, G. 2014. "Nonlinear Control Methods of Industrial Serial Robots." M.Sc. thesis, Istanbul Technical University.

[19] Wampler, C. W. 1986. "Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods." *IEEE Transactions on Systems, Man and Cybernetics* 16 (1): 93-101.