An Efficient Method for Solving the Direct Kinematics of Parallel Manipulators Following a Trajectory

Roshdy Foaad Abo-Shanab
Kafr Elsheikh University/Department of Mechanical Engineering, Kafr Elsheikh, Egypt
Email: roshdyfoaad@gmail.com

***Abstract*—This paper presents an efficient method for solving the direct kinematics of parallel manipulators that follow a defined singularity-free trajectory. Despite the main problem is to solve the inverse kinematics, calculating the errors between the desired path and the actual path requires solving the forward kinematics which is a challenging problem. The proposed method combines closed-loop Jacobian algorithm with Newton-Raphson method to efficiently identify the desired solution. Inverse kinematics is solved using the Jacobian algorithm whereas forward kinematics is solved using Newton-Raphson method. To avoid the numerical instabilities of the Newton-Raphson method, the current state of the manipulator is used as the initial guess for the next state. This makes the numerical solution converges to the correct and desired solution quickly with a few number of iterations. The proposed method is applied to a 3RRR planar parallel manipulator and the simulation results show the effectiveness of the method. The algorithms presented in this article can be applied to other parallel manipulators.**

***Index Terms*—forward kinematics, inverse kinematics, Jacobian, parallel manipulators.**

I. Introduction

A parallel manipulator typically consists of a moving platform that is connected to a fixed base by several limbs. The number of limbs usually equals the number of degrees of freedom such that every limb is controlled by one actuator and all actuators can be mounted at or near the fixed base. Because the external loads can be shared by the actuators, parallel manipulators tend to have a large load-carrying capacity [1]. Also, the positioning accuracy of the end-effector of parallel manipulators is only slightly affected by errors in the actuators, end-effector of a parallel manipulator is its moving platform. Errors tend to average in the parallel case, whereas they are cumulative for a serial manipulator. All of these factors, and the availability of new control and component technologies, have resulted in the increasing popularity of parallel manipulators.

However, parallel manipulators are difficult to design, since the relationships between design parameters and the workspace, and behavior of the manipulator throughout the workspace, are not intuitive by any means [2]-[3]. This is one of the reasons why Merlet [4] argues that customization of parallel manipulators for each application is absolutely necessary in order to ensure that all performance requirements can be met by the manipulator.

Following a trajectory of parallel manipulators requires solving two kinematics problems, inverse kinematics problem and forward kinematics problem. In inverse kinematics, the position and orientation of the end-effector are given and the active joint variables are to be calculated. In the forward kinematics, the joint variables are known and the position and orientation of the end-effector are to be determined. Thus to follow a trajectory, first we use the desired position and orientation to calculate the active joint variables, i.e., solving the inverse kinematics, and the obtained values are used to calculated the actual position and orientation of the end-effector, i.e., solving the forward kinematics. Then, the error between the desired and actual pose (position and orientation) of the end-effector is calculated. Mostly there is a closed form solution for the inverse kinematics for parallel manipulators but not for direct kinematics [5]. Although the forward kinematics problem has been addressed in numerous works, a major portion of them focuses on finding all the possible solutions to the forward kinematics of certain kinds of parallel manipulators [6]-[9]. These approaches usually use algebraic formulations to generate a high degree of polynomial or a set of nonlinear equations. Then methods such as algebraic elimination, interval analysis, and continuation are used to find the roots of the polynomial. The forward kinematics problem is not fully solved just by finding all the possible solutions. Schemes are further needed to find a unique actual pose of the platform that fits the path from among all the possible solutions [5], [10].

In the present work, a method is developed that combines closed-loop Jacobian algorithm with Newton-Raphson to efficiently identify directly the desired solution that fits the path. The active joints variables are calculated using closed-loop Jacobian algorithm which is preferred over the open-loop Jacobian algorithm as the last one suffers from the drift phenomena of the solution. Forward kinematics is solved using Newton-Raphson method and, to avoid the numerical instabilities of this method, the current state of the manipulator is used as the initial guess for the next state. This forces the numerical solution to converge to the correct and desired solution quickly in a few iterations.

Manuscript received June 29, 2013; revised September 28, 2013.

doi: 10.12720/joace.2.3.228-233

The paper is organized as follows. The manipulator geometry and the relations between the inputs and the outputs of the manipulator are introduced in Section II. In Sections III and IV, the solution of the inverse and forward kinematics problems is presented. Section V is devoted to analyze the manipulator Jacobian. Two Jacobian algorithms are introduced in Section VI. Simulation results and a comparison between the proposed schemes are presented in Section VII. Finally, conclusions can be found in Section VIII.

II. Manipulator Geometry

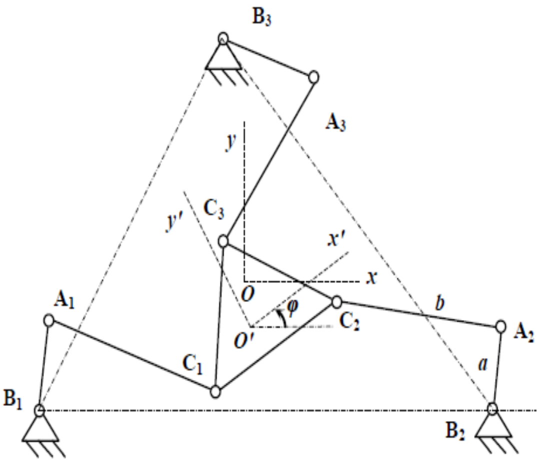


Figure 1. 3 RRR parallel manipulator.

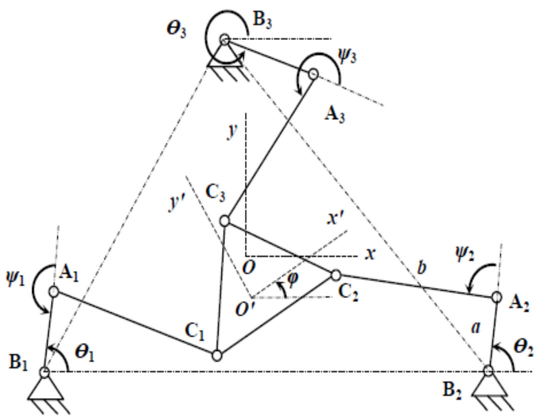


Figure 2. Definition of the position and orientation of the moving platform.

The manipulator considered in this work is a 3RRR planar parallel mechanism. A schematic diagram of the manipulator is shown in Fig. 1. The manipulator geometry was presented by the author in a previous article [11] and will be repeated here for the convenience of the reader. The manipulator consists of a moving equilateral triangular platform of length *h* connected to a fixed equilateral triangular base of length *d* by three limbs. Each limb consists of two links; the first link is connected to the ground by means of a revolute joint identified by the letter B_i and is actuated by a rotary actuator. The three actuators, one for each limb, control the three degrees of freedom of the moving platform (x, y, and φ), see Fig. 2. Two coordinate systems are defined to describe the motion of the moving platform. The first coordinate system is attached to the fixed base (with origin O and axes x and y) and is called the reference frame while the second coordinate system is attached to the moving frame (with origin O' and axes x' and y'). The pose of the end-effector is expressed relative to the reference frame by the position vector $\boldsymbol{Z} = [x\ y\ \varphi]$. The input angles $\boldsymbol{\Theta} = [\theta_1\ \theta_2\ \theta_3]$ is represented by the angular positions of the revolute actuators measured from the x-axis of the reference coordinate system.

III. Inverse Kinematics

From the geometry of the manipulator, shown in Fig. 1 and Fig. 2, a vector loop equation can be written for each limb as

$$OO' = OB_i + B_iA_i + A_iC_i + C_iO' \tag{1}$$

where $i = 1, 2, 3$.

Expanding (1), we get

$$x = x_{Bi} + a\cos\theta_i + b\cos(\theta_i + \psi_i) + l\cos\ (\alpha_i + \varphi) \tag{2}$$

$$y = y_{Bi} + a\sin\theta_i + b\sin(\theta_i + \psi_i) + l\ \sin\ (\alpha_i + \varphi). \tag{3}$$

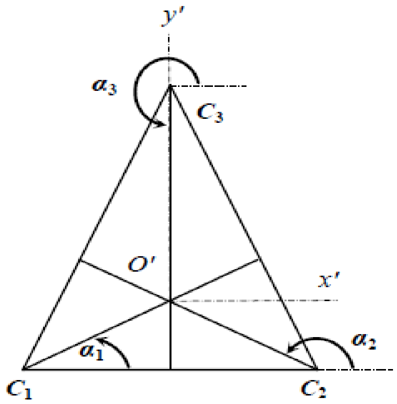


Figure 3. Definition of the moving plate parameters

The definitions of the angles α_1, α_2, and α_3 are shown in Fig. 3. Squaring (2) and (3) and summing the results, we get

$$b^2 = [x - x_{Bi} - a\cos\theta_i - l\ \cos\ (\alpha_i + \varphi)]^2 + [y - y_{Bi} - a\sin\theta_i - l\ \sin\ (\alpha_i + \varphi)]^2. \tag{4}$$

Now, expanding (4) and putting the result in the following form:

$$e_{i1}\sin\theta_i + e_{i2}\cos\theta_i + e_{i3} = 0, \tag{5}$$

where

$$e_{i1} = 2\ a\ l\sin(\alpha_i + \varphi) + 2\ a\ (y_{Bi} - y) \tag{6}$$

$$e_{i2} = 2\ a\ l\cos(\alpha_i + \varphi) + 2\ a\ (x_{Bi} - x) \tag{7}$$

$$e_{i3} = x^2 + y^2 + a^2 + l^2 - b^2 - 2\ x\ x_{Bi} - 2\ y\ y_{Bi} + x_{Bi}^2 + y_{Bi}^2 + 2\ l\cos(\alpha_i + \varphi)\ (x_{Bi} - x) + 2\ l\ \sin(\alpha_i + \varphi)\ (y_{Bi} - y) \tag{8}$$

Substitute the following trigonometric identities in (5)

$$\sin\theta_i = \frac{2t_i}{1+t_i^2},\ \cos\theta_i = \frac{1-t_i^2}{1+t_i^2},\ \text{and}\ t_i = \tan\frac{\theta_1}{2}$$

we obtain

$$(e_{i3} - e_{i2})t_i^2 + 2e_{i1}t_i + (e_{i3} + e_{i2}) = 0. \quad (9)$$

Then

$$\theta_i = 2\tan^{-1}\frac{-e_{i1} \pm \sqrt{e_{i1}^2 + e_{i2}^2 - e_{i3}^2}}{e_{i3} - e_{i2}}. \quad (10)$$

Three cases could be found when solving (10). The first case when the solution gives two different real roots. This means that for each given moving platform location, there are two possible configurations for every limb. The second case, when it yields a double root, this means that this limb is in a fully stretched out or folded back configuration and is called the singular configuration. The third case, when the solution yields no real roots, the specified moving platform location is not reachable, i.e., this location is out of the manipulator workspace [1].

IV. Forward Kinematics

The objective of the forward kinematics is to define a mapping from the known set of the actuated joint angles to the unknown position and orientation of the moving platform. For the present manipulator, the joint angles that are considered known are the angles formed by the input links and the base of the manipulator $\boldsymbol{\Theta} = [\theta_1\ \theta_2\ \theta_3]^T$. The unknown position and orientation of the moving platform is described by the position vector $\boldsymbol{Z} = [x\ y\ \varphi]^T$, which defines the location of O' at the center of the moving platform in the $x'\,y'$ coordinate frame.

Equation (4) describes the relation between the input angles $\boldsymbol{\Theta} = [\theta_1\ \theta_2\ \theta_3]^T$ and the corresponding position and orientation of the moving platform $\boldsymbol{Z} = [x\ y\ \varphi]^T$. To solve the forward kinematics, Equation (4) is written in the following form:

$$\boldsymbol{F} = \boldsymbol{0} \quad (11)$$

where

$$\boldsymbol{F} = [F_1\ F_2\ F_3]^T,$$

and

$$F_i = [x - x_{Bi} - a\cos\theta_i - l\cos(\alpha_i + \varphi))]^2$$
$$+[y - y_{Bi} - a\sin\theta_i - l\sin(\alpha_i + \varphi))]^2 - b^2$$

where $i = 1, 2, 3$. Equation (11) represents a system of nonlinear equations and can be solved using the iterative Newton-Raphson method:

$$\boldsymbol{Z}_{new} = \boldsymbol{Z}_{old} - J_F^{-1}\boldsymbol{F}, \quad (12)$$

where

$\boldsymbol{Z} = [x\ y\ \varphi]^T$, and $\boldsymbol{J_F}$ is the Jacobian matrix of $\boldsymbol{F}$ with respect to $\boldsymbol{Z}$ and it can be computed as follows:

$$\boldsymbol{J}_F = \begin{bmatrix} \frac{\partial F_1}{\partial x} & \frac{\partial F_1}{\partial y} & \frac{\partial F_1}{\partial \varphi} \\ \frac{\partial F_2}{\partial x} & \frac{\partial F_2}{\partial y} & \frac{\partial F_2}{\partial \varphi} \\ \frac{\partial F_3}{\partial x} & \frac{\partial F_3}{\partial y} & \frac{\partial F_3}{\partial \varphi} \end{bmatrix}. \quad (13)$$

Solution of (11) may not be unique. Also, the solution may become divergent, converge to a solution that is not the desired one, or take much time to converge to the correct solution. In this work, as we are following a trajectory, so the solution for the next position is very close to the current position and therefore the current position is used as the initial guess for the next position. This makes the Newton-Raphson method very efficient and converges very fast to the desired solution.

V. Jacobian Analysis of the Manipulator

In this section, the analytical development of the manipulator's Jacobian matrix is presented. For each limb, differentiating (2) and (3), we get:

$$\dot{x} = -a\sin\theta_i\,\dot{\theta}_i - b\sin(\theta_i + \psi_i)\left(\dot{\theta}_i + \dot{\psi}_i\right) - l\sin(\alpha_i + \varphi)\dot{\varphi}, \quad (14)$$

$$\dot{y} = a\cos\theta_i\,\dot{\theta}_i + b\cos(\theta_i + \psi_i)\left(\dot{\theta}_i + \dot{\psi}_i\right) + l\cos(\alpha_i + \varphi)\dot{\varphi}. \quad (15)$$

Solving (14) and (15) to eliminate $\dot{\psi}_i$, we get

$$\cos(\theta_i + \psi_i)\,\dot{x} + \sin(\theta_i + \psi_i)\,\dot{y} - l\sin[(\theta_i + \psi_i) - (\alpha_i + \varphi)]\dot{\varphi} = a\sin\psi_i\,\dot{\theta}_i. \quad (16)$$

Equation (16) is written in the matrix form as follows:

$$\boldsymbol{J_z}\dot{\boldsymbol{Z}} = \boldsymbol{J_\Theta}\dot{\Theta}, \quad (17)$$

where

$$\boldsymbol{J_z} = \begin{bmatrix} \cos\beta_1 & \sin\beta_1 & -l\sin[\beta_1 - (\alpha_1 + \varphi)] \\ \cos\beta_2 & \sin\beta_2 & -l\sin[\beta_2 - (\alpha_2 + \varphi)] \\ \cos\beta_3 & \sin\beta_3 & -l\sin[\beta_3 - (\alpha_3 + \varphi)] \end{bmatrix}, \quad (18)$$

$\beta_i = \theta_i + \psi_i$, and

$$\boldsymbol{J_\Theta} = \begin{bmatrix} a\sin\psi_1 & 0 & 0 \\ 0 & a\sin\psi_2 & 0 \\ 0 & 0 & a\sin\psi_3 \end{bmatrix}. \quad (19)$$

In the above expression, $\boldsymbol{J_z}$ and $\boldsymbol{J_\Theta}$ are two separate Jacobian matrices, these matrices can be combined to obtain a single matrix that establishes the inverse transformation between the input and output velocities:

$$\dot{\boldsymbol{\Theta}} = \boldsymbol{J}\,\dot{\boldsymbol{Z}}, \quad (20)$$

where $\boldsymbol{J} = \boldsymbol{J_\Theta}^{-1}\boldsymbol{J_z}$ corresponding to the inverse Jacobian of a serial manipulator.

VI. Jacobian Algorithms

Two schemes are used to calculate the joint history for following a trajectory. The first scheme uses open loop algorithm, i.e., there is no feedback to correct the deviation of the path away from the desired one and takes the following form:

$$\boldsymbol{\Theta_{new}} = \boldsymbol{\Theta_{old}} + \boldsymbol{J}\ \dot{\boldsymbol{Z}}_{\boldsymbol{d}}\,\Delta t, \quad (21)$$

where $\boldsymbol{J}$ is defined by (20) and $\dot{\boldsymbol{Z}}_d$ is the desired velocity of the end-effector defined in the operational space.

To calculate the deviation of the actual trajectory away from the desired one, forward kinematics problem is solved to calculate the actual position and orientation of the end-effector. Newton-Raphson method, Equation (12), is used to solve the system of nonlinear equations represented by (11). The open-loop scheme, however, suffers from the drift phenomena and as a sequence; the location of the end-effector corresponding to the computed joint variables differs from the desired one. Closed-loop Jacobian algorithm is used in the second scheme to avoid the drift phenomena as follows:

$$\boldsymbol{\Theta}_{new} = \boldsymbol{\Theta}_{old} + \boldsymbol{J}\left(\dot{\boldsymbol{Z}}_{\boldsymbol{d}} + \boldsymbol{K}\,\boldsymbol{e}\right)\Delta t, \qquad (22)$$

where $\boldsymbol{e} = \boldsymbol{Z}_{\boldsymbol{d}} - \boldsymbol{Z}$ is the operational space error between the desired and the actual end-effector position and orientation.

Equation (22) leads to the equivalent linear system:

$$\dot{\boldsymbol{e}} + \boldsymbol{K}\,\boldsymbol{e} = \boldsymbol{0} \qquad (23)$$

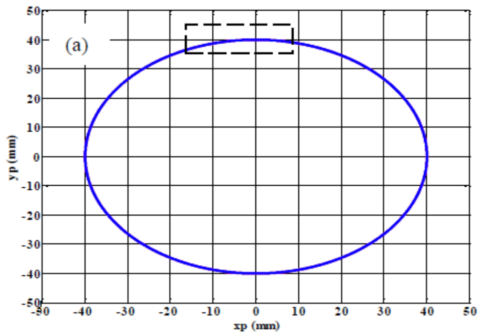
where $\dot{\boldsymbol{e}} = \dot{\boldsymbol{Z}}_{\boldsymbol{d}} - \dot{\boldsymbol{Z}}$ is the time derivative of the error and according to differential kinematics, can be written as $\dot{\boldsymbol{e}} = \dot{\boldsymbol{Z}}_{\boldsymbol{d}} - \boldsymbol{J}^{-1}\dot{\boldsymbol{\Theta}}$.

If $\boldsymbol{K}$ is positive definite (usually diagonal) matrix, the system (23) is asymptotically stable. The error tends to zero along the trajectory with convergence rate that depends on the eigenvalues of matrix $\boldsymbol{K}$; the larger the eigenvalues; the faster the convergence, however, depending on the sampling time, there is a limit for the maximum eigenvalues of $\boldsymbol{K}$ under which asymptotic stability of the error system is guaranteed [12].

VII. Simulation Results

The developed schemes are applied to the present manipulator, shown in Fig. 1. The following numerical values are used for the different manipulator dimensions. $a = 150\ m, b = 337.5\ mm, h = 250\ mm,$ and $B_1B_2 = 600\ mm$. The coordinates of the points of connection of the manipulator with the fixed base are: $B_1(-300, -173.2)\ mm$, $B_2(300, -173.2)\ mm$, and $B_3(0, 346.4)\ mm$. Let the initial location of the end-effector is at $\boldsymbol{Z} = [40\ \ 0\ \ \pi/3]^T$, where x and y are in millimeters and φ is in radians. A circular path of radius $R = 40\ mm$ and a center at the origin (0, 0) is assigned to the end-effector. Let the motion trajectory be

$$\boldsymbol{Z}_d = \begin{bmatrix} R\cos(\omega t) \\ R\sin(\omega t) \\ \pi/3 \end{bmatrix}$$



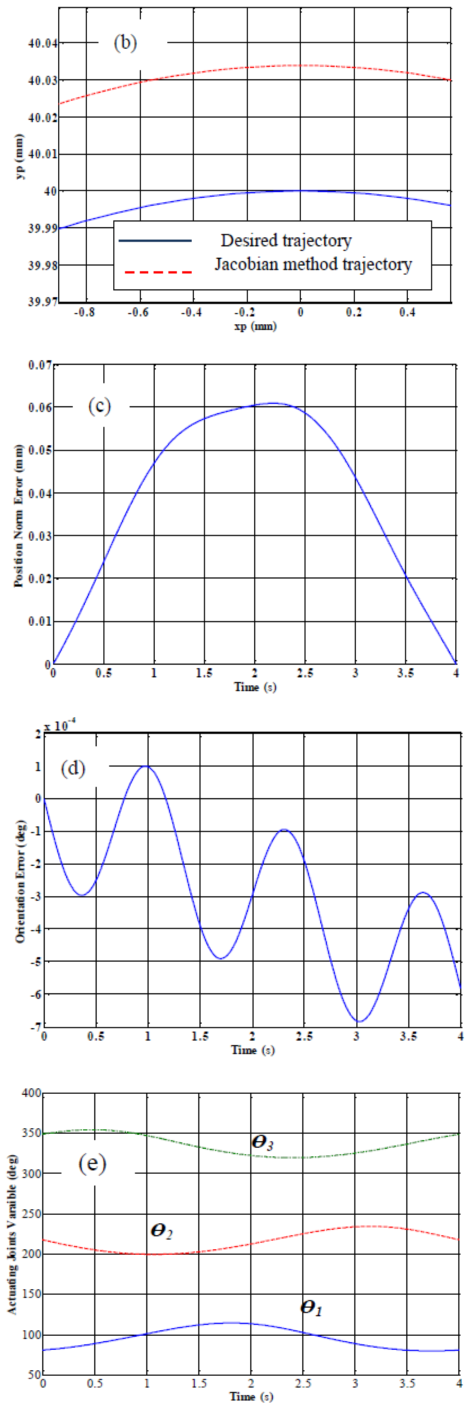
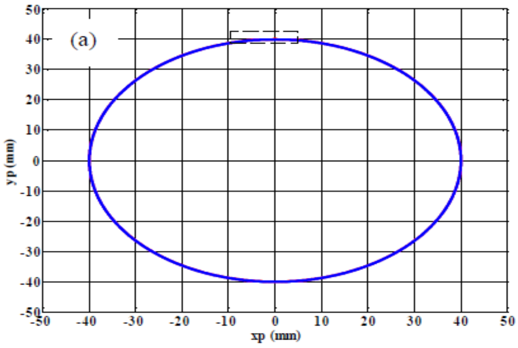
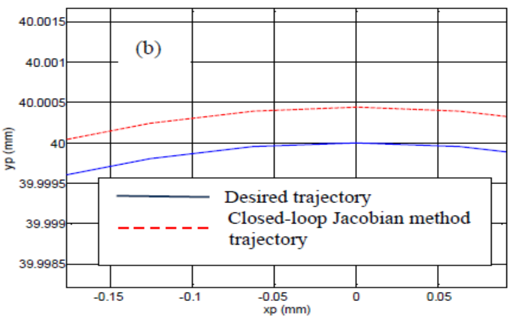


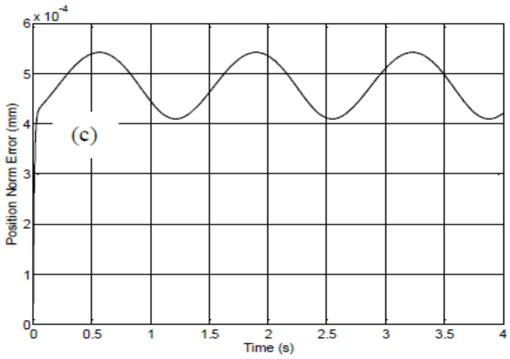
Figure 4. Simulation results using open-loop jacobian scheme

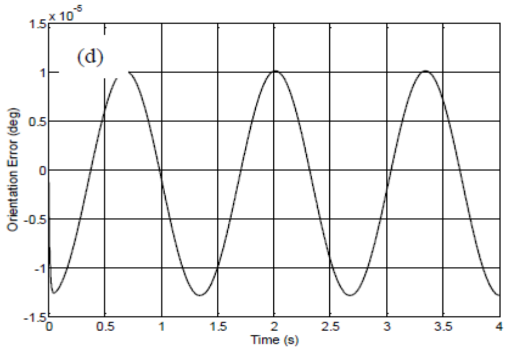
where $\omega = \pi/2$, and the end-effector makes a complete circle in a time of 4 seconds. The mobile platform orientation is kept constant during the motion at $\varphi_d = \pi/3$ and the integration time was chosen to be $\Delta t = 1\ ms$. Fig. 4a shows the actual and desired trajectory of the end-effector, Fig. 4b enlarges a small part of the trajectory to show the difference between the desired and actual trajectory of the end-effector. The norm of the position and orientation errors are shown in Fig. 4c and Fig. 4d, the norm of the position error is bounded but the norm of the orientation error is increasing with time. Fig. 4e shows the time history of the active joints variables.

For the closed-loop scheme, the matrix gain $\boldsymbol{k} = diag\{100, 100, 100\}$ is used. The resulting joint positions and tracking errors are shown in Fig. 5. Fig. 5a shows the actual and desired trajectory of the end-effector, Fig. 5b enlarges a small part of the trajectory to show the difference between the desired and actual trajectory of the end-effector. The position error and orientation errors, Fig. 5c and Fig. 5d, are radically decreased and both are bounded. Fig. 5e shows the time history of the active joints variables.









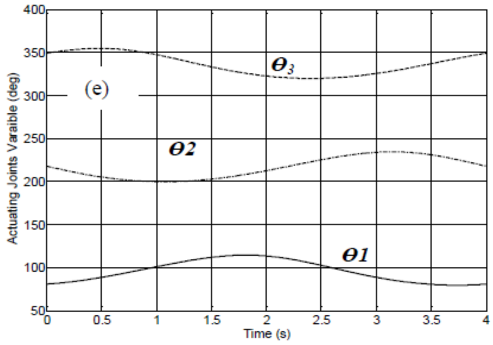


Figure 5. Simulation using closed-loop jacobian scheme.

For both schemes, Newton-Raphson method was used to solve the direct kinematics problem to find the location of the end-effector. It was noticed that the number of iterations for Newton-Raphson never exceeded 3 iterations during the simulation of the whole trajectory, which means that the using the current state as initial guess for the next state forces the solution to converge to the desired solution and greatly enhance the efficiency and accuracy of Newton-Raphson method.

VIII. Conclusion

An efficient method for solving the direct kinematics of parallel manipulators that follow a defined singularity-free trajectory was presented. The proposed method combines closed-loop Jacobian algorithm with Newton-Raphson method to efficiently identify the desired solution.

Solving the forward kinematics of parallel manipulators results in multiple solutions. Previous research, for example [5], focuses on finding all the solutions and then uses another algorithm to find the solution that fits the path. The present method, unlike the previous ones, finds directly the required solution. The method solves the inverse kinematics along the given trajectory using closed-loop Jacobian algorithm. The forward kinematics was solved using Newton-Raphson method and choosing the current state of the manipulator as an initial guess for the next state forces the solution to converge to the required solution that fits the path directly and quickly with a few number of iterations (number of iterations never exceeded three iterations throughout the whole trajectory). The performance of the proposed method was investigated. The method is applied to a *3RRR* planar parallel manipulator and the simulation results showed the efficiency and accuracy of the proposed schemes. The method presented in this article can be applied to other parallel manipulators.

References

[1] L. W. Tsai, *Robot Analysis: The Mechanics of Serial and Parallel Manipulators*, USA: John Wiley and Sons, 1999.

[2] C. Gosselin, L. Perrault, and C. Viallancourt, “Simulation and computer-aided kinematic design of three-degree-of-freedom spherical parallel manipulator,” *Journal of Robotic Systems*, vol. 12, no. 12, pp. 857-869, 1995.

[3] F. Hao and J. P. Merlet, “Multi-criteria optimal design of parallel manipulator based on interval analysis,” *Mechanism and Machine Theory*, vol. 40, pp. 157-171, 2005.

[4] J. P. Merlet, *Parallel Robots*, Dordrecht, Netherland: Kluwer Academic Publishers, 2000.

[5] P. Mahmoodinia, K. Kamali, and A. Akbarzadeh-T, "A novel approach for direct kinematics solution of 3-RRR parallel manipulator following a trajectory," in *Proc. 16th Annual Conference on Mechanical Engineering-ISME*, Shahid Bahonar University of Kerman, Iran, 2008, pp. 1-6.

[6] P. Dietmaier, "The stewart-gough platform of general geometry can have 40 real postures," in *Advances in Robot Kinematics: Analysis and Control*, Strobl, Austria, June 1998, pp.7-16.

[7] M. L. Husty, "An algorithm for solving the direct kinematics of general stewart-gough platforms," *Mechanism and Machine Theory*, vol. 31, no. 4, pp. 365-379, 1996.

[8] P. Ji and H. Wu, "A closed-form forward kinematics solution for the 6-6P Stewart platform," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 522-526, 2001.

[9] J. P. Merlet, "Solving the forward kinematics of a gough-type parallel manipulator with interval analysis," *International Journal of Robotics Research*, vol. 23, no. 3, pp. 221-235, 2004.

[10] Y. Wang, "An incremental method for forward kinematics of parallel manipulators," in *Proc. IEEE International Conference on Robotics, Automation, and Mechatronics*, 2006, pp. 243-247.

[11] R. F. Abo-Shanab, "Optimization of the workspace of a 3R planar parallel manipulator," in *Proc. 2nd International Conference on Mechanical and Electronics Engineering*, Kyoto, Japan, vol. 2, August 1-3, 2010, pp. 429-433.

[12] L. Sciavicco and B. Sicilliano, *Modelling and Control of Robot Manipulaotrs*, Great Britain: Springer-Verlag London, 2000.

Roshdy F. Abo-Shanab received his B.Sc. in Mechanical Engineering with honors from Mansoura University, Egypt, in 1991, M.Sc. in Mechanical Engineering from Assiut University, Egypt, in 1997, and Ph.D. in Mechanical Engineering from University of Manitoba, Manitoba, Canada, in 2003. His major field of study is modeling and simulation of robotic systems.

Since 2007, He has been assistant professor of Mechanical Engineering at Kafr Elsheikh University, Egypt (on leave) and currently he is working with the department of Mechanical Engineering, College of Engineering, Salman Bin Abdulaziz University, KSA.

Dr. Abo-Shanab held various fellowships and awards during his education. He is a member of the Egyptian syndicate for engineers and a member of International Association of Computer Science and Information Technology (IACSIT).