Root-tracking methods and their applications in simulations

Author 1, Author 2

Abstract

This paper presents a comparative study of three different methods used for tracking a particular branch of solution amongst all the solutions arising from solving a set of non-linear equations. Given an initial estimate of the required root, keeping track of the branch it belongs to is a problem that commonly arises in simulating multi-body systems like a parallel manipulator or a cable-driven parallel robot (CDPR). In such cases, these roots represent feasible configurations of the manipulator. Hence, the accuracy and fast computation of the solutions are essential to ensure the safe movement of the manipulator. The primary objective of the paper is to highlight the implementation, present the comparison of three methods of tracking and discuss the context of their application. All the tracking methods are compared with the help of a simulation of the Stewart platform manipulator (SPM) following a desired path. Further, the implementation of such a method allowing a 3-3 CDPR to follow a given path is also demonstrated.

Keywords: Root-tracking, parallel manipulators, cable driven parallel robots (CDPR's)

1. Introduction

(sc:intro)

Unlike linear equations, solving non-linear equations produces multiple sets of feasible solutions or roots. Moreover, these roots do not hold any specific order. There are several methods in the literature to obtain the roots of a set of non-linear equations arising in the kinematics of manipulators, [1, 2, 3, 4](wampler2005numerical, raghavan,Manocha,merlet2009interval). In the case of parametrised set of non-linear equations, the roots keep changing as the parameter is varied. The problem of keeping track of a particular root of interest is essential not only in the kinematics of multi-body systems but also in the problems of speech transmission and direction of arrival [5](starer1992high). In the context of parallel manipulators, their closed-loop architecture

Email addresses: email 1 (Author 1), email 2 (Author 2)

gives rise to the loop-closure constraints. Feasible configurations of such a system are obtained by solving these set of non-linear loop-closure equations. Such a problem is not straightforward and is computationally expensive to obtain the solutions [6](ghosal2006robotics). In practical scenarios, tracking problems are solved either by using sensor-based methods [7, 8](stoughton1991optimal, dallej2012vision) or are avoided by using alternate strategies [9, 10](tempel2015modelling, miermeister2010modelling). Despite these techniques, [11](merlet2017simulation) emphasises the need for methods to solve and track the roots of these equations in developing software platforms for computer simulations of manipulators.

Further, for the simulation of systems like CDPRs where additional constraints are imposed over the kinematic constraints, tracking the branches becomes essential [12](borgstrom2007discrete). In the context of dynamics, an actuator-space formulation is essential for real-time control applications as illustrated in [13](abdellatif2009computational). In such cases, tracking methods enable the computation of the passive joint variables given the actuator variables at each time step from the constraint equations.

Parameter homotopy [14](bates2018paramotopy) is one of the popular methods used in tracking the roots of a parametrised polynomials. The current tracking methods differ from homotopy continuation or parameter homotopy as the later are limited to polynomials, whereas the above-discussed problems might involve a set of generic non-linear equations. In theory, at least the loop-closure equations can be converted to polynomials by the use of trigonometric identities. Such conversion leads to a set of equations with high Bézout's number, thereby increasing the computational burden. Further, such conversions bring in additional parametric singularities which need to be carefully handled. The objective of this paper is to formulate tracking algorithms for a generic set of non-linear equations, ensuring speed and accuracy. To the best of the knowledge of the authors, a comparative study of such root-tracking methods has not been reported extensively in the literature.

The layout of the rest of the paper is: Section 2 deals with the general setting of the problem and algorithms for root-tracking. Section 3 contains the comparative study between these methods and their applicability. Section 4 illustrates the implementation of the algorithms for a path-following problem of a semi-regular Stewart platform manipulator (SRSPM). A demonstration illustrating the advantage of such methods in, a path-following problem of a CDPR is also presented. The conclusion and future work form the Section 5 of the paper.

2. Root-tracking algorithms

(sc:meth)

Consider a set of n non-linear equations in variables $\boldsymbol{x}$ and $\boldsymbol{y}$ of the form,

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{0}, \tag{1}$$

where $\boldsymbol{x}$ is a set of m known and $\boldsymbol{y}$ is a set of n unknown variables implicitly dependent on time. For the system of equations in Eq. (1), there are more than one set of feasible solutions where each set corresponds to a particular branch of solutions. The branches represent the evolution of the roots of the equations ($\boldsymbol{y}$) with the variation of the known variables ($\boldsymbol{x}$). In the context of the kinematics of parallel manipulators, the equations correspond to the loop-closure constraints obtained in terms of the active and passive variables and their solutions represent the forward kinematic (FK) branches which vary with the active variables.

As explained in the Section 1(sc:intro), it is often necessary to keep track of the required branch of roots at each instant as the equations evolve with time, from $t = 0$ to $t = t_f$, i.e., for the entire duration of the simulation. As cited earlier, there are methods in the literature for solving the equations depending on the specific form and nature of the equations, whereas here, we only deal with tracking the roots, given at least one set of solutions, to begin with. For further discussion, it is assumed that roots of the equations are known at time $t = 0$. The following are a few techniques for root-tracking under the above assumptions. Of the n sets of solutions at time t, let the jth solution set represented as $\boldsymbol{y}_j^t$, belong to the required branch. Then the problem is to find the solution belonging to the same branch at time $t = t + \delta t$ amongst all the n branches, $\boldsymbol{y}_1^{t+\delta t}, \boldsymbol{y}_2^{t+\delta t}, \boldsymbol{y}_3^{t+\delta t} \dots \boldsymbol{y}_n^{t+\delta t}$.

2.1. Nearest neighbour method

As explained above, solving a set of non-linear equations would lead to multiple roots or branches of solutions. Generally, in kinematics, one of these branches correspond to the configuration of the physical system whereas the other branches represent the feasible poses. In the nearest neighbour method, the required branch is identified from the physical system. All the roots further obtained by solving the non-linear equations are compared with the solutions of the previous instant. The closest set is then selected as belonging to the branch at each instant of time. Since this method employs the notion of distance for comparison, attention should be given to the space the computed variables belong to. For example, let the variable set involve an angle, say $\theta \in \mathbb{S}^1$. Since the L_2

norm fails to capture the distance between any two elements in $\mathbb{S}^1$, it cannot be used for comparison. In such cases, the variables should be mapped into a subspace where distance can be measured. In the above example, all the angles can be mapped into the sub-space of $[0, 2\pi)$ and the minor and major sector lengths between any two angles can be computed. The smallest amongst the two is chosen to be the distance.

The simulation time is discretised into k finite steps. An algorithm to track the roots through these k discrete steps from time $t = 0$ to $t = t_f$, is presented. It is assumed that the initial configuration of the system is known at $t = 0$. Further, this procedure assumes that there exists a solver (Solve$(\cdot)$) which is capable of computing the solutions of the required set of equations.

Method 1 Root-tracking using the nearest neighbour method

```
Input: Initiate the branch with the intial solution at time $t = 0$, as $\boldsymbol{y}_0$
Output: A list of solutions belonging to the required branch at each instant
1: procedure NearestNeighbour
2:     for $i = 1 \to k$ do
3:         $\boldsymbol{y}_j \leftarrow \text{Solve}(\boldsymbol{f}(\boldsymbol{x}_j, \boldsymbol{y}) = \boldsymbol{0})$    ▷ where $j = 1 \ldots n$ and Solve$(\cdot)$ returns all the roots of the input equations and $\boldsymbol{x}_j$ is the value of the known variables at $jth$ instant
4:         $\boldsymbol{y}_s \leftarrow \text{Select}(\min|\max(\boldsymbol{y}_j - \boldsymbol{y}_0)|)$    ▷ where max$(\cdot)$ and min$(\cdot)$ return the maximum and minimum value among all values of an input list and $|\cdot|$ returns the absolute values of all the elements of the input list. Select$(\cdot)$ selects the element at the index corresponding to the minimum value element of the list
5:         Append$(\boldsymbol{y}_s)$    ▷ Appends $\boldsymbol{y}_s$ to a list of solutions
6:         $\boldsymbol{y}_0 \leftarrow \boldsymbol{y}_s$
7:     end for
8: end procedure
```

Such a method is feasible for manipulators with the loop-closure constraints leading to non-linear equations or polynomials of small order as is the case in [15, 16](agarwal2016dynamic, nasa2011trajectory). Whereas, in the case of manipulators like SRSPM or the 6-$\underline{\text{R}}$SS, this method is computationally intensive as it is required to obtain the roots corresponding to all the branches at each instant. However, if the information of all the branches is available, then the computation time required corresponds only to the time taken in comparison of the roots. Moreover, in reality, computing and updating only the set of solutions corresponding to the required branch is sufficient.

2.2. Newton-Raphson-based root-tracking

Assuming the step size for the update of the known variables to be small and starting from the initial condition corresponding to the required branch, the current method illustrates the Newton-Raphson method employed to track the required root. This method requires the computation of the Jacobian ($\boldsymbol{J}$) of the set of equations, defined as,

$$\boldsymbol{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}. \tag{2}$$

Method 2 Root-tracking using Newton-Raphson method

Input: Initiate the branch with the intial solution at time $t = 0$, as $\boldsymbol{y}_0$
Output: A list of solutions belonging to the required branch at each instant

```
1: procedure NewtonRaphson
2:     for $i = 2 \to k$ do
3:         $\boldsymbol{f}(\boldsymbol{y}) \leftarrow \boldsymbol{f}(\boldsymbol{x}^i, \boldsymbol{y})$  ▷ Substituting $\boldsymbol{x}^i$, value of the known variables $\boldsymbol{x}$ in the ith iteration
4:         $\boldsymbol{J}(\boldsymbol{y}) \leftarrow \boldsymbol{J}(\boldsymbol{x}^i, \boldsymbol{y})$
5:         $\boldsymbol{f} \leftarrow \boldsymbol{f}(\boldsymbol{y}^{i-1})$
6:         while max($|\boldsymbol{f}|$)$\geq \epsilon$ do                ▷ $\epsilon$ is the numerical zero
7:             $\boldsymbol{J} \leftarrow \boldsymbol{J}(\boldsymbol{y}^{i-1})$
8:             $\delta\boldsymbol{y} \leftarrow$ Solve($\boldsymbol{J}\delta\boldsymbol{y} = \boldsymbol{f}$)
9:             $\boldsymbol{y}^i \leftarrow \boldsymbol{y}^{i-1} - \delta\boldsymbol{y}$
10:            $\boldsymbol{f} \leftarrow \boldsymbol{f}(\boldsymbol{y}^i)$
11:        end while
12:    end for
13:    Append($\boldsymbol{y}_i$)                          ▷ Appends $\boldsymbol{y}_i$ to a list of solutions
14: end procedure
```

Where ϵ is the pre-defined numerical zero, i.e., any value a is considered to be zero if $|a| \leq \epsilon$, where $\epsilon \in \mathbb{R}^+$.

The advantages of this method are:

1. Ability to track only the required branch among all the solutions,
2. Ability to tune the accuracy of the solutions,
3. Ability to obtain solutions that strictly satisfy the loop-closure constraints.

An important point to note is, this method ensures that the computed values of the variables satisfy the loop-closure equations to the required accuracy, and hence the obtained roots are always physically feasible. Owing to its advantages, this is the most popular method used in the literature.

2.3. Davidenko method or integration of the first order form of the equations

Consider the given set of equations, Eq. (1). Assuming the equations are scleronomic (loop closure equations are generally scleronomic [17](udwadia2007analytical)), its time derivative is given by,

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}} \dot{\boldsymbol{y}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \dot{\boldsymbol{x}} = \boldsymbol{0}, \tag{3}$$

where $\boldsymbol{y}$ are the unknown variables and $\boldsymbol{x}$ are the known variables. Writing Eq. (3) with $\dot{\boldsymbol{\phi}}$ as the subject gives,

$$\dot{\boldsymbol{y}} = \boldsymbol{J}_{\boldsymbol{yx}}\dot{\boldsymbol{x}}, \quad \text{where } \boldsymbol{J}_{\boldsymbol{yx}} = -\boldsymbol{J}_{\boldsymbol{fy}}^{-1}\boldsymbol{J}_{\boldsymbol{fx}},\ \det(\boldsymbol{J}_{\boldsymbol{fy}}) \neq \boldsymbol{0}, \tag{4}$$

$$\text{and } \boldsymbol{J}_{\boldsymbol{fx}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}},\ \boldsymbol{J}_{\boldsymbol{fy}} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}. \tag{5}$$

Substituting the known values $\boldsymbol{x}$ at each instant in Eq. (4) leads to an initial value problem of the constraint equations in their Pfaffian form in the variable $\boldsymbol{y}$, which is solved as shown. This differential form of the Newton equations considered is called the Davidenko differential equations.

Method 3 Root-tracking by Davidenok's method

Input: Initiate the branch with the intial solution at time $t = 0$ as $\boldsymbol{y}_0$
Output: A list of solutions belonging to the required branch at each instant

```
1: procedure IntegrateConstraint
2:     for $i = 1 \to k$ do
3:         $\Delta\boldsymbol{x} = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$
4:         $\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \boldsymbol{J}_{\boldsymbol{yx}}\Delta\boldsymbol{\theta}$        ▷ Assuming an Euler step for integration
5:         Append($\boldsymbol{y}_{i+1}$)        ▷ Appends $\boldsymbol{y}_i + 1$ to a list of solutions
6:     end for
7: end procedure
```

As opposed to the nearest neighbour method, only the required branch is tracked in this algorithm by supplying the corresponding initial condition. In [18](reddy2016comprehensive) this method is used to track a branch of the solution to the FK problem of automotive suspension for continuous steering and road profile input, which leads to a polynomial of degree 64.

This method works well only for a system that generates lower-order Jacobian matrices, as it involves a matrix inversion in getting to the differential form. The following are the advantages of this method.

1. Newtons-Raphson method follows finite stepping whereas here we have differential equations which are integrated and hence have more control. Generally, the convergence is more reliable in Davidenko equations as compared to the Newton-Raphson method [19](hejase1993use).
2. Rather than using user-selected fixed step, this method relies on error monitoring and step size control of the ODE solver

3. Discussion

(sc:disc) As a measure of the quality of root-tracking, an error metric (e_1) is introduced, which is a measure of the drift of constraint equations, i.e. the measure of how much the constraints are violated, and is defined as,

$$e_1 = \max |\boldsymbol{\eta}(\boldsymbol{q})|, \tag{6}$$

where $\max(\cdot)$ selects the maximum among all the values of a list of elements. It should be noted that all the methods fail at and close to the location of the singularity of the system of equations. In such a situation, one or more branches of solutions merge and hence cannot be resolved using the mentioned tracking methods. The singularity of the system cannot be avoided by the choice of any particular method, it manifests differently in each method. If the singularity is not known, since the computation is done at discrete intervals, there is a possibility that the nearest neighbour method might jump over the singularity without being noticed, as shown in Fig. 1. In the other cases the Jacobian matrix becomes ill-conditioned and hence making it non-invertible. In case of a CDPR, given the high stiffness of the cables, cable tensions change drastically with small changes in cable length but will have no significant effect effects on the position as noted in [11](merlet2017simulation).

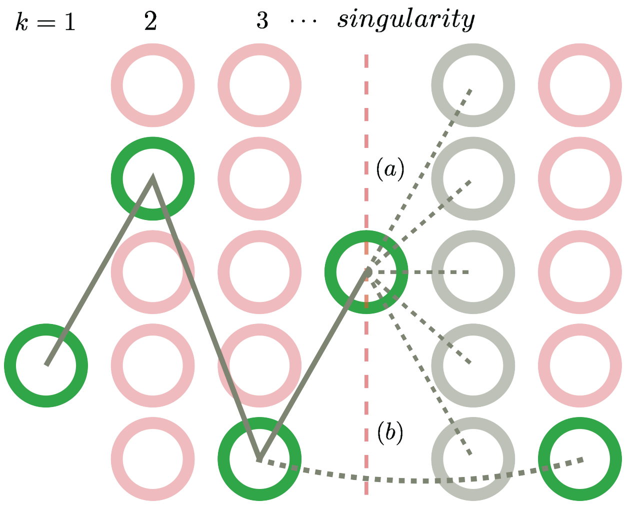

Figure 1: Pictorial representation of the nearest neighbour method. (a) Issue of resolving the branches at a singularity, (b) Possibility of jumping ahead of a singularity

Henceforth, it is assumed that all the manipulators work within their safe working zone (SWZ). The SWZ as described in [20](karnamcomputation) can be summarised as the connected subset of the workspace free from loss-type, gain-type singularities where the manipulator moves within its joint constraints and without self-interference. This would imply that the Jacobian used does not become singular through the working range of the manipulator and can be inverted at all the points in the SWZ. There are several ways to estimating the Jacobians close to the singularity as given in . At a singularity, the individual forward kinematic branches do not carry any significance as two or more of the branches merge and tracking the roots loses meaning. The assumption of motion restricted to within the SWZ allows the existence of the branches uniquely and hence be tracked.

Further, tracking the roots is achieved at discrete time intervals during the simulation as presented in Section 2. Therefore, the size of the discretisation step plays a critical role in determining the quality of the solutions obtained in the methods described. For a given time interval of $[t,\ t+\epsilon]$, the implicit function theorem allows a unique solution for the considered system of non-linear equations in the neighbourhood of the selected solution at time t, $\boldsymbol{y}^t$ given the Jacobian of the system is regular. This ensures that the Newton-Raphson method with the initial guess as $\boldsymbol{y}^t$ will converge to a unique solution $\boldsymbol{y}^{t+\epsilon}$ time $t+\epsilon$ [11](merlet2017simulation).

Since the earlier uses a numerical integration step, there is a trade-off between accuracy and the time taken to compute the solutions. Since the constraint equations are not enforced explicitly, with time, the calculated values diverge from the actual, violating the constraints, i.e., the obtained solutions no longer satisfy the constraint manifold (demonstrated later in Section 4.1.3). The deviation increases with time, and a Newton-Raphson method needs to be employed whenever the drift crosses a predefined threshold value to obtain solutions that satisfy the constraint equations. The choice of numerical integration technique used influences the computational time required and the amount of drift in the roots.

Despite working within the SWZ the root-tracking methods might misbehave for the reasons mentioned above. Moreover, in the case where the above methods fail to converge to a feasible solution, the analytical FK formulation is used to compute all the sets of roots required for tracking in the current step and resumes the use of one of the above mentioned methods from the next instant. This strategy ensures smooth operation of the tracking algorithm. The implementation of such a method is illustrated in [21](vyankatesh2018).

4. Implementation

(sc:impl) The implementation of the above three algorithms is demonstrated by solving a path following problem of two different manipulators, i.e., the SRSPM and the 3-3 CDPR. Further, comparison in terms of accuracy and computational speed of these algorithms is presented in the following sections.

4.1. Semi-Regular Stewart Platform Manipulator (SRSPM)

The Gough-Stewart or the Stewart platform manipulator is a six degree-of-freedom parallel manipulator, as shown in Fig. 2(fig:srspm1). Its construction consists of a fixed platform, connected to six linear actuators (legs) through universal or spherical joints. A moving platform is attached to the other ends of the linear actuators via spherical joints, as shown in Fig. 2(fig:sr-spm1). Since its introduction in [22](stewart1965platform), this manipulator has attracted a large amount of research on the topics of kinematics, dynamics and control. The interest in this partic-

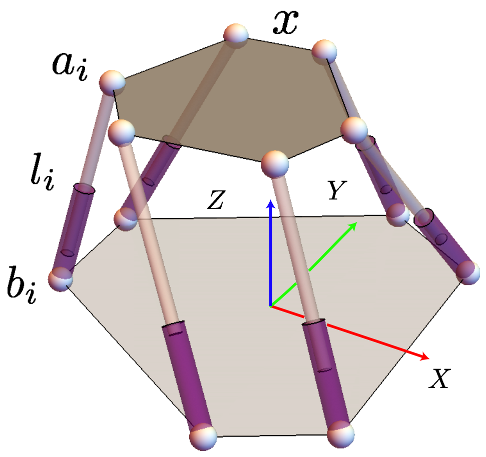

Figure 2: Architecture of a Semi-Regular Stewart Platform Manipulator

ular manipulator comes from its wide range of applications including automotive simulators [23, 24](freeman1995iowa, park2001development), flight simulators [25](pradipta2013development), machine tools [26](lebret1993dynamic), etc. The FK problem of the SRSPM deals with obtaining all the feasible poses of the manipulator given the lengths of prismatic links.

4.1.1. Constraint equations and FK

There are several prior works in solving the forward kinematic problem, e.g., [27, 28, 29](lee2001forward, dasgupta1994canonical, lee2003improved). However, as mentioned in [30](nag2019comparative), solving the FK problem involves arriving at the forward kinematic univariate (FKU), and finding the roots of this 20 degree polynomial for an SPM, which might be computationally expensive. These roots are then tracked as the leg lengths of the manipulator change to follow the given path. In the subsequent sections, implementation of various root-tracking methods discussed above for tracking the configuration of SRSPM as it follows a given path are described. The configuration space of SRSPM is of dimension 18 and consists of,

$$\boldsymbol{q} = [\boldsymbol{\phi}^\top, \boldsymbol{\theta}^\top]^\top, \tag{7}$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ are variables representing the 6 leg lengths and 12 passive joint angles respectively. A set of 12 constraint equations are obtained by enforcing the conditions of rigidity of the top platform as explained in [31](bandyopadhyay2006geometric). This set of constraints are non-linear equations with trigonometric terms. The physical parameters of the manipulator can be found in Appendix A.

The subsequent part of the section illustrates implementation of the three methods described earlier simulated in C++ on an AMD Ryzen 7, with 8 core CPU, 16 GB RAM and a clock speed of 3.6 GHz installed with Ubuntu 18.04.1 LTS. All the execution time details reported are the average values over 1000 trials run using only a single core.

4.1.2. Nearest neighbour method

For the sake of demonstration, the centroid of the moving platform of the SRSPM is required to follow a circular path for which the pose of the moving platform is parametrised as,

$$\boldsymbol{X} = [0.2\cos\alpha, 0.2\sin\alpha, 1.1, 0.2\sin\alpha, 0.2\cos\alpha, 0]^\top.$$

The path is discretised in terms of the parameter α into 50 discrete points at which the leg lengths are obtained by solving the inverse kinematics problem. Given all the set of roots apriori, then this method would be computationally efficient as only comparison is done with no other operations done on the solutions.

4.1.3. Integration of the first order of the equations

For the path given in Section 4.1.2, the described Method 0(Ialgo) is used to obtain the passive joint angle values at each instant. An explicit Euler integration scheme is used to integrate the obtained ODE equations, Eq. (4). The tracking of the complete path discretised into 50 time steps is achieved in 0.452 milliseconds. Further, the error e_1 (Eq. (6)) is plotted against the time instant, and is as shown in Fig. 3. Starting from the initial estimate of the configuration variables, the current method tracks the roots only belonging to this branch at each instant of time. Given that only a few operations are involved in this method, it is generally computationally faster than the Newton-Raphson method for a Euler integration step. As expected and seen from Fig. 3, the solutions start to drift from the actual values violating the loop-closure constraints. A Newton-Raphson correction step is needed whenever a predefined tolerance value is exceeded, compensating for the drift. This ensures that the corrected solutions satisfy the constraint manifold again, as shown in Fig. 4.

4.1.4. Newton-Raphson method for root-tracking

Starting from the initial condition, the passive joint variables required to follow the path described in 4.1.2 are computed using the Method 0. The time taken to follow is found to be

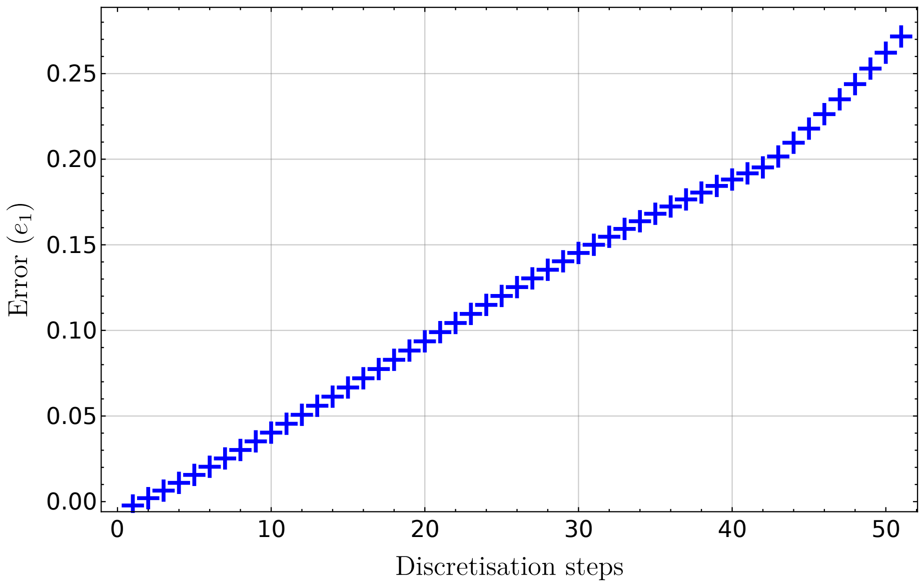

Figure 3: Error measure (e_1) as given in Eq. (6), representing the drift of the solutions obtained from the constraint manifold at each step of integration

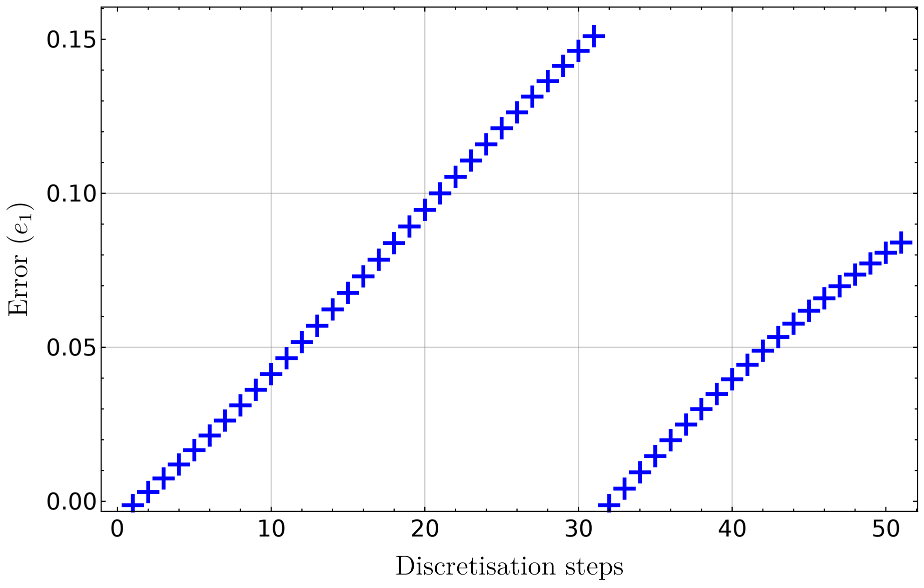

Figure 4: Depiction of correction of the drift in the solutions, using a Newton-Raphson step, bringing them onto the constraint manifold

0.777 milliseconds, for 50 discretisation steps. The order of magnitude of the error defined in Eq. (6) is observed to be 10^{-10}.

The results of all the methods are summarised in the Table 1.

Note that the computational accuracy of the solutions obtained in the nearest neighbour method depend on the FK algorithm used. Special care must be taken and multi-precision computations ([29](lee2003im
are required to compute the roots to higher accuracies, which are computationally expensive but yet are not accounted in the time taken to track the roots.

Table 1: Computational time taken and the accuracy of the solutions in various methods of root-tracking

Root-tracking method	Computational time (ms)	Accuracy (Error e_1)	Tunable parameter
Nearest neighbour	0.004	10^{-10}	FK algorithm used
Constraint integration	0.452	10^{-1}	Euler integration step used
Newton-Raphson	0.777	10^{-10}	Accuracy goal used

4.2. Cable driven parallel robots (CDPR)

Cable-driven parallel manipulators (CDPR) are a class of parallel manipulators with the moving platform connected to the stationary platform via cables. Despite sharing the low mass and better load carrying characteristics of the parallel manipulators, the constraint of maintaining tension in all the cables at each instant adds to the challenges in the analysis. The interest in CDPR's is inspired from its potential applications in material handling [32](bostelman1994applications), exoskeletons [33](garrec2008able), rehabilitation [34](cablerehab) and reconfigurable robotics [35](nguyen2014a

The spatial movement of the end-effector of a CDPR is enabled by changing the cable lengths by actuating their respective winches. The forward kinematics problem deals with the problem of finding the position and orientation of the moving platform given the lengths of the cables. As given in [36, 37](yamamoto1999inverse, gallina2001planar), the computation of the FK solutions at each iteration is essential, especially for the Cartesian space control of the CDPR. Unlike parallel robots with rigid membered links, the loop closure equations must be solved along with the conditions of static equilibrium to obtain a correct FK solution of a CDPR. Further, additional constraints on unilaterality of the cable forces should be imposed to obtain the physically feasible configurations.

4.2.1. The 3-3 cable-driven parallel manipulator

Consider the 3-3 cable-driven manipulator illustrated in [38](carricato2010geometrico), where the mobile platform is connected to a fixed base via 3 cables, as shown in Fig. 5. It is a spatial under-constrained manipulator and the geometrico-static problem that needs to be solved to obtain feasible poses is a difficult problem as discussed in [38](carricato2010geometrico).

It is assumed that the cables are non-deformable. For the treatment of a general redundant manipulator, one should refer to [11], which deals with the problem of changing of the operating cables during the motion. A set of 3 loop-closure equations are derived by enforcing the constraints on the lengths of the three cables, A_iB_i. Following the methodology described in [38](carricato2010geometrico), further, a set of 12 equations are obtained from the conditions ensuring the static equilibrium of the manipulator. These, in the total form a set of 15 equations in 6 variables ($\boldsymbol{X}$), describing the

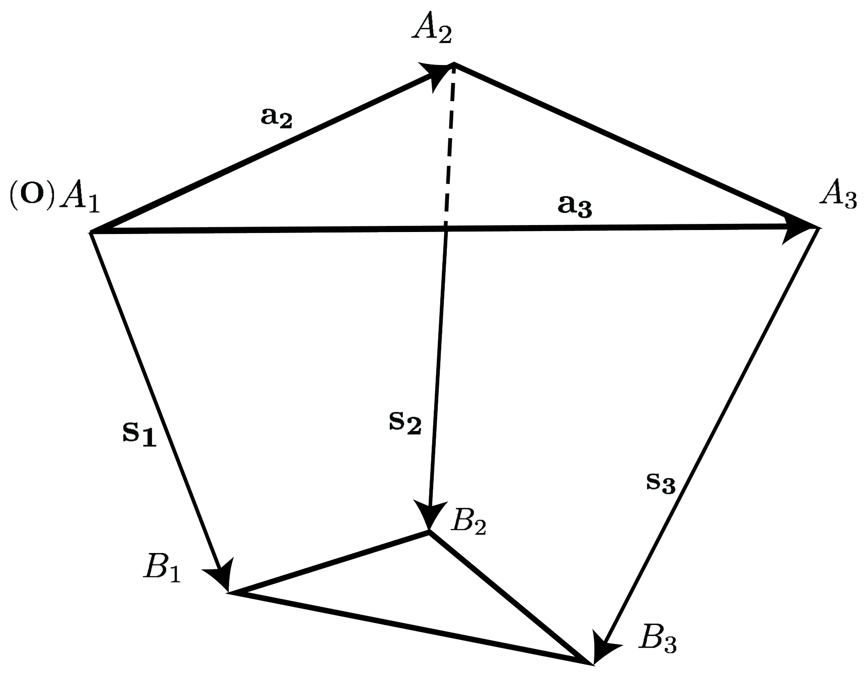

Figure 5: Description of the 3-3 CDPR [38]

position and orientation of the end-effector where $\boldsymbol{X}$ is,

$$\boldsymbol{X} = [x, y, z, e_1, e_2, e_3]^\top. \tag{8}$$

As mentioned in [38](carricato2010geometrico), the obtained set of 12 equations are linearly independent, and hence the selection of any three equations and the three loop-closure constraints form a valid system of 6 equations in 6 unknowns which would lead to the same set of solutions. As mentioned in the [39](abbasnejad2012real), there will be spurious roots introduced in solving the FK problem with the choice of the equations, but this is not true for tracking the roots since we start with a known initial guess. Since the choice of the equations does not change the solution set, the equations with the smallest degree in $\boldsymbol{X}$ are selected. Further, it should be noted that irrespective of the equations selected, the obtained roots would also satisfy all the 15 constraint equations mentioned above.

4.2.2. Root-tracking problem

The manipulator is assumed to follow the given path quasi-statically, i.e. move in such a way that any dynamic effects can be ignored. The FK problem of the 3-3 CDPR yields 156 solutions or branches. Unilaterality constraint of the solutions are checked, and only the poses which admit tension in the cables are selected. Hence, checking the feasibility of the FK solutions for every iteration adds to the computational burden. Due to the advantage of control over the accuracy of the computed solutions, the Newton-Raphson method is selected. The physical parameters of the manipulator used can be found in Appendix B.

4.2.3. Newton-Raphson method based root-tracking

The work [38](carricato2010geometrico) reports that there are 156 possible solutions and only 10 of them are real of which only 6 admit positive tensions in the cables. One of these solutions is taken as a starting pose for the root-tracking.

$$\boldsymbol{X} = [-3.3554,\ 0.542536,\ 1.71102,\ 2.93133,\ 4.07689,\ 6.04519]^{\top}. \tag{9}$$

The path to be followed quasi-statically by the moving platform is obtained by interpolating the cable lengths between the initial and the final configurations.

$$\boldsymbol{\rho}_i = \frac{1}{2}[15,\ 20,\ 19]^{\top}, \tag{10}$$

$$\boldsymbol{\rho}_f = [10,\ 11,\ 8]^{\top}, \tag{11}$$

where the $\boldsymbol{\rho}_i$ and $\boldsymbol{\rho}_f$ denote the vectors of the initial and final lengths of the cables. Following the method described in Method 0(NRalgo), the six constraint equations and its corresponding Jacobian matrix is computed. The error e_1 for the solutions obtained from root-tracking using Newton-Raphson method were of the order of 10^{-11} which is comparable to the reported roots obtained from solving the FK problem. For the discretisation of 1000 steps, the time taken to track is 0.0619 seconds, for required precision set to 10^{-10}. The red curve is the path followed by the manipulator shown in Fig. 6.

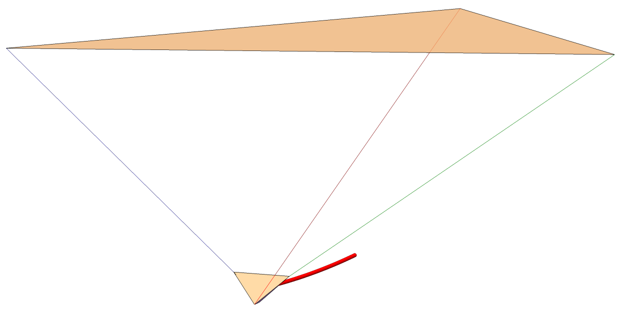
Figure 6: The path followed by the moving platform of the CDPR represented by a red curve

5. Conclusion

Three methods for the application of root-tracking are discussed and compared. The drawbacks and the scope of the methods are studied using their implementation on SRSPM for a path following problem. As the number of solutions for the non-linear equations increase, it is difficult to solve the FK solution iteratively as in the case of CDPRs. A path following application is demonstrated for the 3-3 CDPR. Such methods allow one to simulate the dynamics of the system with any set of variables used to represent the system. Fast computations allow higher control bandwidths and hence better tracking performance of high-frequency motions.

6. Acknowledgement

Appendix A. Physical parameters of the SRSPM

Table A.2: Parameters of the SRSPM used to solve the FK problem

Parameter	Symbol	Value	Unit
Circumradius of the base platform	r_b	1	m
Circumradius of the moving platform	r_t	0.5803	m
Angular spacing between the adjacent pair of legs of the fixed platform	γ_b	0.2985	rad
Angular spacing between the adjacent pair of legs of the moving platform	γ_t	0.6573	rad
Length of the sliding link of the prismatic joint	$l_{bi}, i = (1, \ldots, 6)$	0.5	m
Length of the fixed link of the prismatic joint	$l_{ai}, i = (1, \ldots, 6)$	1.5	m

Appendix B. Details of the 3-3 CDPR

Table B.3: Physical parameters used for the FK problem of the CDPR

Parameter	Symbol	Value (m)
Coordinates of the vertices of the fixed platform	$\boldsymbol{a}_1$	$[0, 0, 0]^\top$
	$\boldsymbol{a}_2$	$[10, 0, 0]^\top$
	$\boldsymbol{a}_3$	$[0, 12, 0]^\top$
Coordinates of the vertices of the moving platform	$\boldsymbol{b}_1$	$[1, 0, 0]^\top$
	$\boldsymbol{b}_2$	$[0, 1, 0]^\top$
	$\boldsymbol{b}_3$	$[0, 0, 1]^\top$

References

[1] A. J. Sommese, C. W. Wampler, The Numerical Solution of Systems of Polynomials Arising in Engineering and Science, WORLD SCIENTIFIC, 2005. arXiv:https://www.worldscientific.com/doi/pdf/10.1142/5763, doi:10.1142/5763.

[2] M. Raghavan, B. Roth, Solving Polynomial Systems for the Kinematic Analysis and Synthesis of Mechanisms and Robot Manipulators, Journal of Mechanical Design 117 (B) (1995) 71–79. arXiv:https://asmedigitalcollection.asme.org/mechanicaldesign/article-pdf/117/B/71/5920678/71_1.pdf, doi:10.1115/1.2836473.

[3] D. Manocha, S. Krishnan, Solving algebraic systems using matrix computations, SIGSAM Bull. 30 (4) (1996) 4–21. doi:10.1145/242961.242965.

[4] J.-P. Merlet, Interval analysis for certified numerical solution of problems in robotics, International Journal of Applied Mathematics and Computer Science 19 (3) (2009) 399–412.

[5] D. Starer, A. Nehorai, High-order polynomial root tracking algorithm, in: ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 4, 1992, pp. 465–468 vol.4. doi:10.1109/ICASSP.1992.226335.

[6] A. Ghosal, Robotics: Fundamental Concepts and Analysis, Oxford University Press, New Delhi, 2006.

[7] R. Stoughton, T. Arai, Optimal sensor placement for forward kinematics evaluation of a 6-DOF parallel link manipulator, in: IEEE/RSJ International Workshop on Intelligent Robots and Systems, IEEE, 1991, pp. 785–790.

[8] T. Dallej, M. Gouttefarde, N. Andreff, R. Dahmouche, P. Martinet, Vision-based modeling and control of large-dimension cable-driven parallel robots, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 1581–1586.

[9] P. Tempel, P. Miermeister, A. Lechler, A. Pott, Modelling of kinematics and dynamics of the IPAnema 3 cable robot for simulative analysis, in: Progress in Production Engineering, Vol. 794 of Applied Mechanics and Materials, Trans Tech Publications Ltd, 2015, pp. 419–426. doi:10.4028/www.scientific.net/AMM.794.419.

[10] P. Miermeister, A. Pott, Modelling and real-time dynamic simulation of the cable-driven parallel robot IPAnema, in: D. Pisla, M. Ceccarelli, M. Husty, B. Corves (Eds.), New Trends in Mechanism Science, Springer Netherlands, Dordrecht, 2010, pp. 353–360.

[11] J.-P. Merlet, Simulation of discrete-time controlled cable-driven parallel robots on a trajectory, IEEE Transactions on Robotics 33 (3) (2017) 675–688.

[12] P. H. Borgstrom, N. P. Borgstrom, M. J. Stealey, B. Jordan, G. Sukhatme, M. A. Batalin, W. J. Kaiser, Discrete trajectory control algorithms for NIMS3D, an autonomous underconstrained three-dimensional cabled robot, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2007, pp. 253–260.

[13] H. Abdellatif, B. Heimann, Computational efficient inverse dynamics of 6-DOF fully parallel manipulators by using the lagrangian formalism, Mechanism and Machine Theory 44 (1) (2009) 192 – 207. doi:https://doi.org/10.1016/j.mechmachtheory.2008.02.003.

[14] D. Bates, D. Brake, M. Niemerg, Paramotopy: Parameter homotopies in parallel, in: J. H. Davenport, M. Kauers, G. Labahn, J. Urban (Eds.), Mathematical Software – ICMS 2018, Springer International Publishing, Cham, 2018, pp. 28–35.

[15] A. Agarwal, C. Nasa, S. Bandyopadhyay, Dynamic singularity avoidance for parallel manipulators using a task-priority based control scheme, Mechanism and Machine Theory 96 (2016) 107 – 126. doi:https://doi.org/10.1016/j.mechmachtheory.2015.07.013.

[16] C. Nasa, S. Bandyopadhyay, Trajectory-tracking control of a planar 3-$\underline{\text{R}}$RR parallel manipulator with singularity avoidance, in: 13th World Congress in Mechanism and Machine Science, 2011, pp. 19–25.

[17] F. E. Udwadia, R. E. Kalaba, Analytical Dynamics: A New Approach, Cambridge University Press, Cambridge, 1996.

[18] K. V. Reddy, M. Kodati, K. Chatra, S. Bandyopadhyay, A comprehensive kinematic analysis of the double wishbone and MacPherson strut suspension systems, Mechanism and Machine Theory 105 (2016) 441–470.

[19] H. A. N. Hejase, On the use of Davidenko's method in complex root search, IEEE Transactions on Microwave Theory and Techniques 41 (1) (1993) 141–143. doi:10.1109/22.210241.

[20] M. K. Karnam, A. Baskar, R. A. Srivatsan, S. Bandyopadhyay, Computation of the safe working zones of planar and spatial parallel manipulators, Robotica (2019) 1–25.

[21] V. Ashtekar, S. Bandyopadhyay, Forward dynamics of the double-wishbone suspension mechanism using the embedded Lagrangian formulation, in: Asian MMS Conference, IFToMM Asian Mechanism and Machine Science, 2018, pp. 1–16.

[22] D. Stewart, A platform with six degrees of freedom, Proceedings of the Institution of Mechanical Engineers 180 (1) (1965) 371–386.

[23] J. S. Freeman, G. Watson, Y. E. Papelis, T. C. Lin, A. Tayyab, R. A. Romano, J. G. Kuhl, The Iowa driving simulator: An implementation and application overview, in: SAE Technical Paper, SAE International, 1995. doi:10.4271/950174.

[24] M. K. Park, M. C. Lee, K. S. Yoo, K. Son, W. S. Yoo, M. C. Han, Development of the PNU vehicle driving simulator and its performance evaluation, in: IEEE International Conference on Robotics and Automation, Vol. 3, IEEE, 2001, pp. 2325–2330.

[25] J. Pradipta, M. Klünder, M. Weickgenannt, O. Sawodny, Development of a pneumatically

driven flight simulator Stewart platform using motion and force control, in: IEEE/ASME International Conference on Advanced Intelligent Mechatronics, IEEE, 2013, pp. 158–163.

[26] G. Lebret, K. Liu, F. L. Lewis, Dynamic analysis and control of a Stewart platform manipulator, Journal of Robotic Systems 10 (5) (1993) 629–655.

[27] T.-Y. Lee, J.-K. Shim, Forward kinematics of the general 6–6 Stewart platform using algebraic elimination, Mechanism and Machine Theory 36 (9) (2001) 1073–1085.

[28] B. Dasgupta, T. Mruthyunjaya, A canonical formulation of the direct position kinematics problem for a general 6-6 Stewart platform, Mechanism and Machine Theory 29 (6) (1994) 819 – 827. doi:https://doi.org/10.1016/0094-114X(94)90081-7.

[29] T.-Y. Lee, J.-K. Shim, Improved dialytic elimination algorithm for the forward kinematics of the general Stewart–Gough platform, Mechanism and Machine Theory 38 (6) (2003) 563–577.

[30] A. Nag, S. Bandyopadhyay, A comparative study of the configuration-space and actuator-space forward dynamics of closed-loop mechanisms using the Lagrangian formalism, in: D. N. Badodkar, T. A. Dwarakanath (Eds.), Machines, Mechanism and Robotics, Springer Singapore, Singapore, 2019, pp. 95–106.

[31] S. Bandyopadhyay, A. Ghosal, Geometric characterization and parametric representation of the singularity manifold of a 6–6 Stewart platform manipulator, Mechanism and Machine Theory 41 (11) (2006) 1377–1400.

[32] R. Bostelman, J. Albus, N. Dagalakis, A. Jacoff, J. Gross, Applications of the NIST RoboCrane, in: Proceedings of the 5th International Symposium on Robotics and Manufacturing, Vol. 5, 1994, pp. 14–18.

[33] P. Garrec, J. P. Friconneau, Y. Measson, Y. Perrot, ABLE, an innovative transparent exoskeleton for the upper-limb, in: 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 1483–1488. doi:10.1109/IROS.2008.4651012.

[34] H. Xiong, X. Diao, A review of cable-driven rehabilitation devices, Disability and Rehabilitation: Assistive Technology 0 (0) (2019) 1–13, pMID: 31287340. arXiv:https://doi.org/10.1080/17483107.2019.1629110,, doi:10.1080/17483107.2019.1629110.

[35] D. Q. Nguyen, M. Gouttefarde, O. Company, F. Pierrot, On the analysis of large-dimension reconfigurable suspended cable-driven parallel robots, in: IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 5728–5735.

[36] M. Yamamoto, N. Yanai, A. Mohri, Inverse dynamics and control of crane-type manipulator, in: Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289), Vol. 2, 1999, pp. 1228–1233 vol.2. doi:10.1109/IROS.1999.812847.

[37] P. Gallina, A. Rossi, R. L. Williams II, Planar cable-direct-driven robots, part II: Dynamics and control, in: ASME Design Engineering Technical Conference, Vol. 2, 2001, pp. 1241–1247.

[38] M. Carricato, J.-P. Merlet, Geometrico-static analysis of under-constrained cable-driven parallel robots, in: Advances in Robot Kinematics: Motion in Man and Machine, Springer, 2010, pp. 309–319.

[39] G. Abbasnejad, M. Carricato, Real solutions of the direct geometrico-static problem of under-constrained cable-driven parallel robots with 3 cables: a numerical investigation, Meccanica 47 (7) (2012) 1761–1773.