Robotica (2016) volume 34, pp. 1–22. © Cambridge University Press 2014
doi:10.1017/S0263574714001362

The forward kinematics of the 6-6 parallel manipulator using an evolutionary algorithm based on generalized generation gap with parent-centric crossover

Luc Rolland†* and Rohitash Chandra‡

†*Memorial University, Faculty of Engineering, St-John's, Canada*
‡*School of Computing, Information and Mathematical Sciences, University of the South Pacific, Fiji*

(Accepted April 30, 2014. First published online: June 19, 2014)

SUMMARY
In this paper, a fast and efficient evolutional algorithm, called the G3-PCX has been implemented to solve the forward kinematics problem (FKP) of the general parallel manipulators being modeled by the **6-6** hexapod, constituted by a fixed and mobile platforms being non planar and non-symmetrical. The two platforms are connected by six linear actuators, each of which is located between one ball joint and one universal joint. Forward kinematics are formulated using Inverse Kinematics applying one position based equation system which is converted into an objective function by expressing the sum of squared error on kinematics chain lengths and mobile platform distances. In less than one second, the 16 unique real solutions are computed with improved accuracy when compared to previous methods.

KEYWORDS: Parallel robot; Forward kinematics; Position based model; Genetic algorithms; G3-PCX.

1. Introduction
Parallel manipulators were introduced by Gwinnett who invented the *Oxymoron* for a cinema motion simulator (USA Patent no. 1,789,680). The first industrial robot patents were filed by William Pollard (USA Patent no. 2,286,571) and his son (USA Patent no. 2,213,108). Based on the *Multi-Axis Simulation Tables*, Gough and Cappel constructed the first parallel hexapods respectively for a tire testing device[13] and a motion simulator (USA Patent no. 3,295,224). Since their successful application as flight simulators,[34] parallel manipulators have attracted academic and industrial interest. Hence, in 1979, MacCallion implemented the first industrial parallel robot in an industrial environment.[20]

The general **6-6** parallel manipulator is defined as an hexapod constituted by a fixed base, a mobile platform with the end-effector and six similar kinematics chains, Fig. 1. The rigid platform joints are not located on a plane and they are not defining a specific geometry. Each kinematics chain is constituted by one prismatic or linear actuator located between one Universal or ball joint and one ball joint. The mobile platform, kinematics chains, fixed base are considered infinitely rigid and the joints provide for relative motions with no friction nor backlash. We restrict our study to the robots with six kinematics chains where the sensor number is not exceeding the number of end-effector *degrees-of-freedom*.

Lazard, Ronga and Mourrain have proven that the general 6-6 hexapod FKP has 40 complex solutions using respectively Gröbner bases, Chern classes of vector bundles and explicit elimination techniques.[18,24,32] From an engineering point-of-view, the significant issue is the one of real solutions since they correspond to effective manipulator postures. The number of real solutions is always equal or less than the number of complex ones. Recently, Dietmaier has proposed an algorithm which modifies a Gough platform configuration into one which features 40 real solutions.[5] Fast

* Corresponding author. Email: lrolland@mun.ca, luc.rolland@gmail.com

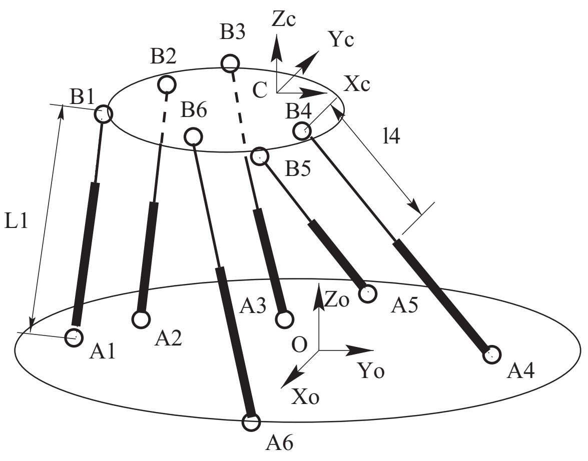

Fig. 1. The general spatial manipulator as the typical **6-6** hexapod.

numerical approaches usually implement Newton's method.[6] However, Newtons method can suffer from Jacobian inversion problems, numeric instabilities related to the application of floating numbers, and convergence problems when not provided with a good initial position in the search space. The Geometric Iteratice Method has shown its prowess in being faster than Newton's method on complex problems but is not yet available as a solving package.[35] Homotopy Continuation methods might then be advocated,[26] but they are prone to miss solutions.[29] Resultant or Dialytic Elimination methods have also been successfully applied on a certain number of problems,[16,36] but they might add spurious solutions.[29] More recently, interval analysis was implemented that could find certified results.[22] However, homotopy and interval analysis require one numerical method for solving an equation system and this is usually Newton's method, implying that, in certain conditions, some solutions may not be found.[22,26]

There exists also an algebraic method applying Gröbner bases from which a rational univariate representation is computed producing exact certified results,[28] however response time can reach several minutes on general hexapod FKPs. This method can be applied as a verification and validation tool for any other faster solving methods such as numerical analysis or evolutionary computation methods.

Evolutionary algorithms (EAs) employ a population of solutions which are evolved and improved. They are also part of the family of metaheuristic search algorithms which optimize a problem by iteratively improving the solution. Genetic algorithms (GAs) are a major type of evolutionary algorithms. They have shown good performance in real world optimization problems.[4,15,37] In parallel robotics, evolutionary algorithms have been initially applied for solving the forward kinematics problem of the 3-RPR.[1] Boudreau and Turkkan proposed a real-coded genetic algorithm (RCGA), also called floating point genetic algorithm for improved approximation integrating crossover and mutation operators inspired by operators used in binary GA. They reported that the RCGA method is more time consuming than Newton-Raphsons method.[1] It was shown that the convergence domain is larger allowing process launch with a more distant initial guess or, in the case of GAs, an initial population with a larger initial population zone. This surely constitutes a reason to favor evolutionary algorithms. Moreover, GAs provide for simpler calculations where Jacobians are not required to be computed and matrices are not inverted. More recently, the FKPs of the spherical and spatial manipulators have also been solved with genetic algorithms, the **3-RRR** by[25] and the ideal Gough platform by[7,11] being the **SSM**, still not the general **6-6** hexapod. However, in ref. [11], a binary coded genetic algorithm is implemented to find one solution. In ref. [38], it has been reported that the differential evolution algorithm converges to a solution relatively faster than traditional GA on a **SSM** platform. The FKP of the general 6-6 hexapod was solved for the first time with GAs by the authors[30] using Wrights heuristic crossover and non-uniform mutation where several solutions could be properly validated for the first time. The combination of GAs with simulated annealing into a hybrid meta-heuristic method further improved the performance.[30] The hybrid meta-heuristic method was further improved for solving the FKP of the 3-RPR.[3]

Evolutionary Algorithms are good alternatives to numerical methods if they could be adapted to calculate all solutions, offering a compromise between response time and the solution with very high accuracy. However, for the moment, being of heuristic nature, they cannot be certified and they do provide for solution approximate estimates. Our research goal is to produce a method which can find all solutions either in one long run or in multiple runs. However, the proposed method should be faster then the exact method and should give relatively accurate results. To be applicable in one of the areas of robotics, such as material handling, task repetitivity of 0.5 mm and accuracies of 3 mm are required.

The G3-PCX genetic algorithm consists of the generalized generalization gap model for selection and the parent-centric operators for reproduction.[17] It has shown improved performance when compared to other evolutionary algorithms for a set of global optimization problems. This paper implements the G3-PCX genetic algorithm for solving the FKP of the general form hexapod, namely the general 6-6 parallel manipulator. The selected FKP has 16 distinct real solutions confirmed by an exact algebraic method based on proven Groebner bases.[28] The G3-PCX will be used to find all the 16 solution and the accuracy and response time of the results will be compiled and compared with the literature. Note that this paper is a substantial extension of the results presented in [31].

Parallel robot FKPs are associated with solving a system of non-linear equations and are rarely treated as a direct optimization problem. Therefore, the non-linear equation solving problem has to be converted into an optimization problem where the objective function describes the entire robot system kinematics.[30]

As we have just reviewed, several studies have shown the capability of genetic algorithms to find one or more solutions to the FKP, without going into further details. In this paper, we provide in depth analysis for the specific 6-6 FKP case on one difficult configuration for the first time. In prior trials reported in ref. [30], we have shown that genetic algorithms can produce results which do not necessarily correspond to the equation system solutions. On the specific 6-6 FKP case, the proposed G3-PCX genetic algorithm will produce hopefully all solutions which should be verified justifying that the exact solutions will be calculated with an exact algebraic method based on proven Groebner bases.[28] The difference between the G3-PCX results and the exact ones will allow to determine the absolute accuracy. The main aim of this research work is the efficiency demonstration of the G3-PCX method on the difficult problems of solving the kinematics problems of the 6-6 parallel manipulator.

We underline the advantages and limitations in terms of response time, solution error and capacity to find one specific and ultimately all real solutions. Moreover, the same objective function can be expressed with its parameters being either floating numbers or rational numbers. The G3-PCX algorithm will be tried on both formulations and results and performance will be compared. This work will also verify if the proposed algorithm has a tendency to jump to a distant solution even in the case where the initial population is selected in the vicinity of each exact solution.

The paper is organized as follows: In Section 2, the FKP formulation is reviewed and the conversion of the non-linear equation system into an optimization function is given. In Section 3, an overview of the metaheuristic techniques with emphasis on genetic algorithms and their variants is presented. Section 4 details the (G3-PCX) algorithm implemented in this study. Section 5 presents the experiments, the results and their analysis. Section 6 concludes the paper with directions for future research.

2. Forward Kinematics of the 6-6 Parallel Manipulator

Any manipulator is characterized by its mechanical configuration parameters and the posture variables. The configuration parameters are thus $\mathbf{OA}_{|R_f}$, the base joint attachment point coordinates in R_f (the base reference frame, located at $\mathbf{O}$), and $\mathbf{CB}_{|R_m}$, the mobile platform joint attachment point coordinates in R_m (the mobile platform reference frame, located at $\mathbf{C}$). The base joint coordinates are located at fixed positions on the base platform. The mobile platform joint coordinates are located at fixed positions on the mobile platform. These joint positions are determined in the configuration parameter file. The kinematics model variables are the joint coordinates and end-effector generalized coordinates. The joint variables are described as l_i, the prismatic joint or linear actuator positions. The generalized coordinates are expressed as $\overrightarrow{X}$ comprising the end-effector position and orientation.

The generalization of the hexapod, namely Gough platform,[13] often called Stewart platform,[34] is selecting configurations without any specific geometric properties such platform symmetries or

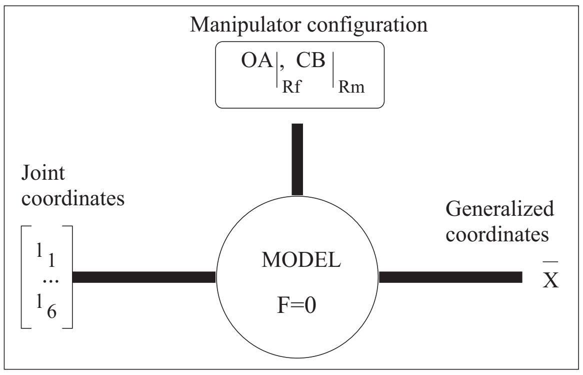

Fig. 2. Kinematics model.

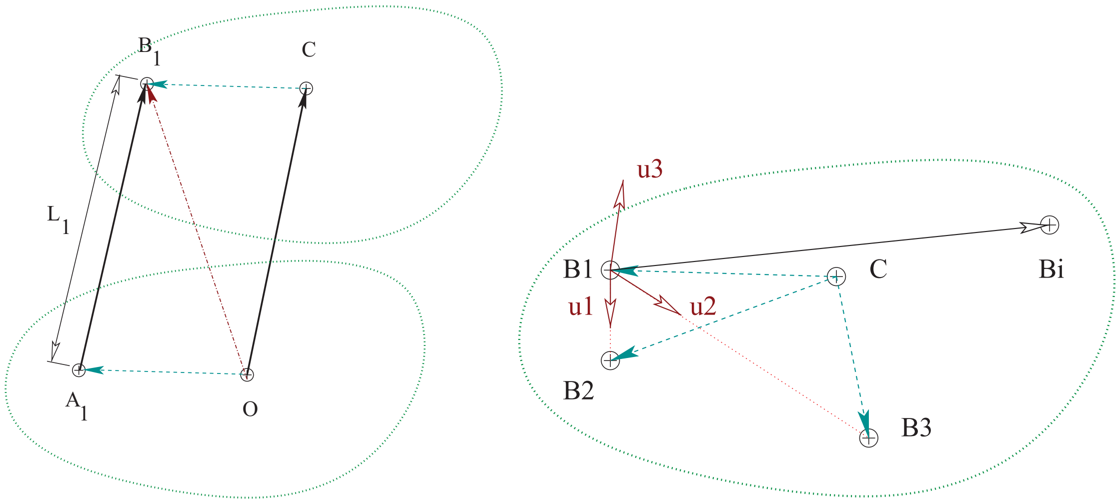

Fig. 3. Kinematics chain and mobile platform vectors.

joint position coincidences.[21] In numerical methods, specific geometric properties usually lead to computation simplifications and smaller mathematical model expression and in many instances, the methods cannot solve the general configurations.[29]

For the sake of clarity and simplicity, $\mathbf{OA}_{|_{R_f}}$ will be replaced by $\mathbf{OA}_O$ and $\mathbf{CB}_{|_{R_m}}$ by $\mathbf{CB}_O$.

This article shall only concentrate on the forward kinematics problem (**FKP**), Fig. 2, identified as a difficult problem.[27] Usually the *inverse kinematics problem* is required to model the **FKP** and is defined as:[26] *given the generalized coordinates of the manipulator end-effector, find the joint positions.*

Accordingly, the *forward kinematics problem* is defined as:[26] *given the joint positions, find the generalized coordinates of the manipulator end-effector.*

2.1. *Vectorial formulation of the implicit kinematics model*

Containing as many equations as variables, vectorial formulation constructs an equation system for each kinematics chain,[6] as a closed vector cycle between the A_i and B_i kinematics chain attachment points, the fixed base reference frame O and the mobile platform reference frame C. For each kinematics chain, an implicit function $\overrightarrow{A_i B_i} = U_1(X)$ can be written between joint positions A_i and B_i. Each vector $\overrightarrow{A_i B_i}$ is expressed knowing the joint coordinates $\overline{L}$ and X giving function $U_2(X, \overline{L})$. The following equality has to be solved: $U_1(X) = U_2(X, \overline{L})$. The distance between A_i and B_i is set to l_i. Thus, the end-effector position X or C can be derived by one platform displacement $\overrightarrow{OC}$ and then one platform general rotation expressed by the rotation matrix $\mathcal{R}$. For each distinct platform point $\overrightarrow{B_i}_O$ with $i = 1, \ldots, 6$, see Fig. 3, the position can be calculated in terms of the base reference frame:[21]

$$\overrightarrow{OB_i}_O = \overrightarrow{OC} + \mathcal{R}\overrightarrow{CB_i} \tag{1}$$

The vectorial formulation, evolves as a displacement based equation system using the following relation :

$$\overrightarrow{A_i B_i} = \overrightarrow{OC} + \mathcal{R}\overrightarrow{CB_i} - \overrightarrow{OA_i} \tag{2}$$

As such, these six equations cannot be applied as such. Hence, each kinematics chain can be expressed using the distance norm constraint:[21]

$$l_i^2 = ||A_i B_i||^2 \tag{3}$$

The rotation matrix $\mathcal{R}$ can be expressed utilizing various orientation models with their specific rotation variable sets such as navigation angles (yaw, pitch and roll), Euler angles, quaternions or even taking the nine rotation matrix components as variables.[29] Implementing the equation 2 directly, various displacement based equation models can be derived depending on the selected orientation variables.[29]

In our experience, several different evolutionary algorithms and optimization techniques could not isolate the orientation variables to acceptable accuracy levels less then 1 degree. Even quaternion and double quaternion based models led to variable isolation with poor result accuracy in the order of 25 percent. Preliminary genetic algorithm experiments have shown that displacement based equations where positions and orientations are represented differently with their own variables produce the same absolute error levels making orientations less likely to be determined.

For example, if we typically apply simple sensitivity analysis using a classic displacement based model where the variables are the end-effector position $\overrightarrow{OC} = \{x_c, y_c, z_c\}$ and the Euler angles for rotations, we can obtain an error of 0,5 on a position over a position range of 1000 giving a sensitivity of 0,00005 on position while the same value of 0,5 on radian measured angles with a range of 2π produce a sensitivity of 0,07958, thereby leading to a ratio of 1591,5. This means that any optimization technique will produce results being 1591 times more precise for position then orientation. If the orientations are expressed in terms of trigonometric functions instead of angles, then angle sensitivity would reach 0,25 increasing ratio to 5000.

Any rigid object can be positioned into three dimensional space by three distinct points, Fig. 3. Selig demonstrated how this principle can be applied to describe the position and displacement of any robotic end-effector constituted by one rigid body.[33] Any rigid body three points are actually characterized by distance constraints which remain constant. This principle was then applied to the forward kinematics model of parallel manipulators by Lazard[19] resulting in the so-called position based equation models. It is easy to choose three distinct points which are not collinear on most mobile platforms. These three points are usually selected to coincide with three joint centers connecting the mobile platform to the kinematics chains allowing to utilize the vectorial model, 3 and to rewrite of $\overrightarrow{A_i B_i}$, 2 as it will be explained in the next section.

Two reasons justify the choice of the position based model. Every variable yield the same units and their ranges are equivalent leading to the same weight in the equation system. Hence, the rotation impact is included into the point parameters and made equivalent to the translation impact. Preliminary genetic algorithm experiments have shown that angles could be more precisely isolated using position-based equation models then with displacement based models.

The main disadvantage could be the unknown number exceeding the end-effector DOF number,[28] but our preliminary experiments showed that this does not necessarily lead to significantly slower computation times. The second disadvantage pertains to its inability to model specific mobile platforms leading to collinear joints, but in reality, this is rarely occurring.

2.2. *The inverse kinematics problem*

The 3 is actually the general form of the IKP.

The coordinates of the three distinct joint center points become the nine variables from which constraints equation can be written. The three platform distinct points are usually selected as the three first joint centers, namely B_1, B_2 and B_3. Each coordinate of the selected joint centers becomes a variable. The nine end-effector variables are set to : $\overrightarrow{OB}_{i|O} = [x_i, y_i, z_i]$ for $i = 1 \dots 3$. To simplify computations, we choose one non-Cartesian reference frame R_{b_1} to be located at B_1 joint center. Then, we define u_1, u_2 and u_3 as R_{b_1} reference frame axes which are calculated by:

$$u_1 = \frac{\overrightarrow{B_1B_2}}{||\overrightarrow{B_1B_2}||}, \; u_2 = \frac{\overrightarrow{B_1B_3}}{||\overrightarrow{B_1B_3}||}, \; u_3 = u_1 \wedge u_2 \tag{4}$$

This new reference frame R_{b_1} is applied instead of R_m as the mobile platform Cartesian reference frame and has its origin located at B_1 and the reference frame axes u_1 and u_2 point towards B_2 and B_3 respectively. The third reference frame u_3 points perpendicular to the plane determined by B_1, B_2 and B_3. It becomes the mobile platform pointing axis. This transformation is achieved to produce a simpler equation system.

Knowing that the mobile platform is supposed infinitely rigid, any platform point M can be expressed in the reference frame R_{b_1} by calculating the following linear composition:

$$\overrightarrow{B_1M} = a_M u_1 + b_M u_2 + c_M u_3 \tag{5}$$

where a_M, b_M, c_M are constants in terms of these three points. Hence, in the case of the **IKP**, the constants are noted $a_{B_i}, b_{B_i}, c_{B_i}$, $i = i \dots 6$ and can explicitly be deduced from the mobile platform fixed distances $\mathbf{CB}_{|C}$ by solving the following linear system of equations :

$$\overrightarrow{B_1B}_{i|R_{b_1}} = a_{B_i} u_1 + b_{B_i} u_2 + c_{B_i} u_3 \;,\; i = 1 \dots 6. \tag{6}$$

where $\overrightarrow{B_1B}_{i|R_{b_1}} = \overrightarrow{B_1B}_{i|C}$.

Note that the mobile platform fixed distances $\mathbf{CB}_{|C}$ are given by the configuration which is obtained from the design values or deduced from a calibration procedure after the Gough platform manipulator construction. The configuration file is providing the position of all six joints of the mobile platform relative to the mobile platform reference frame and this ensure that the points belong to the same rigid body which is the mobile platform.

Equation 7 requires that we calculate the configuration distances with:

$$\overrightarrow{B_1B}_{i|C} = \overrightarrow{CB}_i - \overrightarrow{CB}_1 \;,\; i = 1 \dots 6. \tag{7}$$

Hence, the remaining three mobile platform joint centers B_4, B_5 and B_6 are expressed in terms of the nine end-effector variables.

Using the relations Eq. (6), the distance constraint equations $l_i^2 = ||\overrightarrow{A_iB}_{i|O}||^2$, $i = 1 \dots 6$ can be expressed Thus, for $i = 1 \dots 6$, the **IKP** is obtained by isolating the l_i actuator variables in the six following equations:

$$l_i^2 = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2 \;,\; i = 1 \dots 3 \tag{8}$$

$$l_i^2 = ||\overrightarrow{B}_{i\,|R_{b_1}} - \overrightarrow{OA}_{i\,O}||^2 \;,\; i = 4 \dots 6 \tag{9}$$

2.3. *The forward kinematics problem*

For the general Gough platform parallel manipulator, it is actually not possible to express the FKP directly or explicitly.[21] We have to revert to the **IKP** expression which gives an algebraic system comprising six equations in terms of three point variables : $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$, Eq. (9).

This system contains algebraic (polynomial) functions which can be handled by the numerical solvers implemented in all genetic algorithms.

The usual method advocated for writing the FKP equation system starts by rewriting the IKP as functions. This produces an algebraic system of three leg equations and three functions in terms of the nine variables: $x_i, y_i, z_i\ ,\ i = 1, 2, 3$.

$$F_i = (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2 - l_i^2\ ,\ i = 1 \dots 3 \tag{10}$$

$$F_i = ||\overrightarrow{B_i}_{|R_{b_1}} - \overrightarrow{OA}_{i\,O}||^2 - l_i^2\ ,\ i = 4 \dots 6 \tag{11}$$

When solving the FKP with numeric or algebraic methods, it is necessary to provide a zero-dimensional system, meaning an equation system which contains as many equations as their are variables,[28] and.[29] In this case, this means that to the six equations provided by the IKP, three more shall be selected to close the system.

Moreover, the actual FKP is derived directly from the IKP model, Eq. (10, 11), and it does not provide for any information to constrain the position of the mobile platform joint positions which are necessary to describe the FKP.

Hence, to complete the algebraic system and to constrain the mobile platform joint positions, three constraints are derived from the following three functions. Two functions can be written using two characteristic platform distances, expressed as norms between the B_1, B_2 distinct points and the B_1, B_3 ones. The computations will select the variables which are only at the right distance from the B_1 reference joint point. These constraint equations require one last equation. The points are known relative to each other in terms of distance but the mobile platform alignment is left undetermined. To alleviate this problem, the third constraint equation will determine where the mobile platform is pointing. The pointing vector is selected as the one perpendicular to the three points $B_i\ ,\ i = 1, 2, 3$ by calculating the vectorial multiplication of the two vectors separating B_2 and B_3 from B_1:

$$F_7 = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 - ||\overrightarrow{B_2 B_1}_{|R_{b_1}}||^2 \tag{12}$$

$$F_8 = (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 - ||\overrightarrow{B_3 B_1}_{|R_{b_1}}||^2 \tag{13}$$

$$F_9 = (x_3 - x_1)(x_2 - x_1) + (y_3 - y_1)(y_2 - y_1) + (z_3 - z_1)(z_2 - z_1) - ||\overrightarrow{B_3 B_1}_{|R_{b_1}}|| \wedge ||\overrightarrow{B_2 B_1}_{|R_{b_1}}|| \tag{14}$$

In Eq. (14), as the choice of F_9, the last function, provides an important mobile platform constraint related to the pointing axis. For F_9, it would be possible to write a function related to the distance between B_2 and B_3 but our experience shows us that it does not lead to better results then the platform pointing function.

The result constitutes then an algebraic system with nine equations in the former nine unknowns.

In the case of unconstrained optimization methods such as the G3-PCX method, the system to optimize usually contains a large number of variables in one function to be either maximized or minimized. It is just necessary to write a minimum number of constraints sufficient to model the equation system. Genetic algorithms handle a certain number of repeated runs which should be kept minimal in order to obtain reasonable response times. This justifies limiting the constraint equations to their minimum.

3. The Conversion to an Optimization Problem

The most important consideration in creating a genetic algorithm is defining a problem representation, meaning the careful selection of the physical model which, in this case, should represent the Gough platform FKP.[23]

In solving an equation system, the goal is to find all real roots of a function system which then can be applied for assembly mode analysis and singularity analysis or a specific root like in trajectory planning or even control.[21] In the case of the Gough platform, the FKP model comprise an equation system $(F(X) = 0)$ with $F = \{f_1(X), \dots . f_9(X)\}$ where $X = \{x_1, \dots, x_9\}$.

Genetic algorithms like all evolutionary computations are tailored to solve optimization problems where an objective function is either maximized or minimized and therefore cannot solve any system of equation directly.[23] Solving an equation system ($F(X) = 0$) is different from solving an optimization problem where we wish to minimize or optimize a function, ($min(G(x_1, \dots, x_n)$ or $max(G(x_1, \dots, x_n))$.

The first difference is that equation solving finds the roots to a function system constituted of several functions concurrently, thereby equating all functions to zero, whereas the optimization of one function finds either one maximal value or one minimal value, either local or maximal.

Secondly, the optimization process usually finds only one answer or so-called one optimum.

Thirdly, one cannot simply take the equation system and apply one optimization technique directly onto them.

The non-linear equation solving problem system has to be converted to an optimization problem, meaning to transform a robot kinematical model system of nine non-linear functions into a model comprising only one function describing the same system.

Knowing that each equation system root is actually cancelling the nine functions, then it possible to combine these nine functions together:

$$G(X)_{opti} = \sum_{1}^{9}(f_i(X)). \tag{15}$$

The equation solving problem corresponds to finding when the combined function cancels. For each solution set $\{x_1, \dots, x_9\}_s$ which cancels all $f_i(X)$ functions of the $F(X) = 0$ equation system, the same solution also cancels the $G(X)_{opti}$ function since this a sum of the former, meaning $G(X)_{opti} = 0$.

Since, no actual equation solver method can solve one multivariate equation, meaning find the roots to one multivariate function, the function can be minimized by any optimization method. The G3-PCX algorithm can play that role.

In the FKP case, the problem becomes the isolation of the same solution with the same initial population range. In real-time control or robot trajectory simulation, the initial population ranges are determined from the last trajectory position. The selected zone will be closer to one specific solution and further away from the alternate solutions. In some instances, it is well known that Newton's method does not always find the closest solution to an initial guess depending on the Jacobian behavior.

The following step in the subsection 3.1 consists in determining one objective function from the equation system.

3.1. Objective function

Evolutionary Algorithms generally require one or several objective functions, often called fitness, performance or cost functions, to be maximized or minimized.[23] Therefore, we need to convert the non-linear equation solving problem into an optimization one. The fitness value is actually defined as the result of the instantiations of the objective function with the variable estimates.

In Eq. (9), the chosen kinematics model intrinsic nature pertains to distance constraints evaluated by the calculation of distance squared norm between known points which are the base and mobile platform joint positions. We have chosen the six prismatic actuators determining the six kinematics chain lengths. It is also possible to add three lengths between the three joint centers OB_1, OB_2, OB_3 or even more.

The simplest form of an optimization problem involves single function minimization. We set the nine distances as open values which will be sought. The algorithms will be giving values to the end-effector variables x_i, y_i, z_i , $i = 1, 2, 3$ and the corresponding distances will be calculated. These estimated distances will be subtracted from the known distances producing the error on each distance. The distances are either the kinematics lengths $l_i, i = 1 \dots 6$ given as input to the FKP or measured mobile platform configuration values. Each error can be added and this produces the objective function. The exact solution is found when the objective function is zero, meaning that all distance errors are also zero. There will be only one objective function being the sum of six to nine distance errors. Finding the exact solution is very unlikely, therefore, the process will be stopped when the error is sufficiently small and this value will be applied as a process stopping criteria.

The distance errors cannot be solved independently since this would only return any point located on a sphere around a known fixed point determined by the related base joint position. This justifies the solving of all distance errors at the same time be solving of the resulting global error function.

The optimization problem will be written as an unconstrained problem where the variables can reach any value.

Hence, from the IKP, we can easily derive an objective function. This function will be calculated on each FKP estimation representing the total error on each kinematics chain lengths. Let lg_i be the length of kinematics chain i which are given as problem inputs. Let $H_i = l_i^2$, from Eq. (9), the fitness function F_F is set to:

$$F_F = \sum_{i=1}^{6} (\sqrt{H_i} - lg_i)^2 \tag{16}$$

Equation 16 contains six individual objectives being the kinematics chain length difference constraints which will be minimized.

The expanded result for Eq. (16) becomes:

$$F_F = \sum_{i=1}^{3} (((x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2)^{1/2} - lg_i)^2$$
$$+ \sum_{i=4}^{6} (||\overrightarrow{B_i}_{|_{R_{b_1}}} - \overrightarrow{OA}_{i\,O}|| - lg_i)^2 \tag{17}$$
$$\tag{18}$$

This objective function includes the six single objectives obtained from the 6 kinematics chain lengths and its detailed expression easily spans over several lines and cannot be shown here.

3.2. *Extended objective function*

Preliminary tests led to several solutions which were absolutely not in correspondence with the exact proven ones.[30] In other words, the selected fitness function was incorrect since it did find other solutions which did not correspond to the FKP solutions.

To alleviate this problem, the selected fitness function is then augmented by the 3 constant distances between the 3 distinct platform points: B_1, B_2 and B_3.

Knowing that $||\overrightarrow{B_2B_1}_C||$, $||\overrightarrow{B_3B_1}_C||$ and $||\overrightarrow{B_3B_2}_C||$ are determined from the measured configuration of the mobile platform, the mobile platform distances can be converted to distance errors adding the following functions to the objective.

$$\begin{aligned} G_1 &= ||\overrightarrow{B_2B_1}_C||^2 - (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \\ G_2 &= ||\overrightarrow{B_3B_1}_C||^2 - (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 \\ G_3 &= ||\overrightarrow{B_3B_2}_C||^2 - (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2 \end{aligned} \tag{19}$$

Hence, the fitness function becomes :

$$F_F = \sum_{k=1}^{3} (\sqrt{H_k} - lg_k)^2 + \sum_{n=1}^{3} (G_n)^2 \tag{20}$$

The expanded result for equation 20 becomes:

$$\begin{aligned}
F_F = &\sum_{i=1}^{3}(((x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 + (z_i - OA_{iz})^2)^{1/2} - lg_i)^2 \\
&+ \sum_{i=4}^{6}(||\overrightarrow{B_i}_{|R_{b_1}} - \overrightarrow{OA}_{iO}|| - lg_i)^2 \\
&+ ||\overrightarrow{B_2B_1}_C||^2 - (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \\
&+ ||\overrightarrow{B_3B_1}_C||^2 - (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 \\
&+ ||\overrightarrow{B_3B_2}_C||^2 - (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2
\end{aligned} \tag{21}$$

This objective function is unique and includes the 9 single distance objectives obtained from the 6 kinematics chain lengths and 3 platform distance constraints and its detailed expression easily spans over several lines.

4. Background on Evolutionary Algorithms

4.1. Genetic algorithms

Evolutionary algorithms are nature inspired computational paradigms which use a population of candidate solutions (called individuals or chromosomes) that are improved over time with evolutionary operators. Examples of evolutionary algorithms are genetic algorithms,[2,15] evolution strategies,[25] evolutionary programming and genetic programming.[7] Most evolutionary algorithms utilize a solution population and the differences mainly lie in how new solutions are created. Evolutionary algorithms are considered to be global search methods, can be used for problems that are noisy and change over time and are problem independent. They can be applied to solve optimization problems and can be used for problems that are non-differentiable. The term generation is mainly used for an iteration in which a population of new individuals is created in the evolutionary process.

The canonical genetic algorithm introduced by Holland used binary encoding to represent solutions.[15] Real coded genetic algorithms (RCGA) implemented real numbers rather than bits.[9]

Initially, a number of possible random solutions constitute the initial population. In our cases, the candidate solutions are seeded in the area of search space known as the robot workspace where the desired solution is likely to be found. After each generation, the algorithm computes the objective function on each individual to evaluate its fitness and employs genetic operators to produce offspring from selected best parents. The fitness measures solution quality which is problem dependent. The offspring are added into the population while least fit individuals are discarded. The process is repeated until the algorithm obtains a sufficiently good solution. In the genetic algorithm, the evolutionary process continues until the termination condition is satisfied. The search is terminated when one of the following is true: 1) maximum number of generations or function evaluations is reached, 2) a solution with a fitness function being sufficiently minimized. This goal is obtained when an arbitrary error threshold is reached. 3) the highest ranking solutions come to a point where there is no change of fitness indicating no improvement in candidate solutions.

The choice of the appropriate genetic operator is important as it directly influences the convergence of the genetic algorithm. However, different forms of the main genetic operators are needed according to the type of the genetic algorithm and the nature of the optimization problem.[4]

1. **Selection:** The selection operator plays an important role as it ensures that qualities from the fittest individuals always survive in future generations.
2. **Reproduction using Crossover:** The crossover operator exchanges genetic material from selected parents and forms either a single or multiple offspring.
3. **Reproduction using Mutation:** The mutation operator provides random diversity in the population. This is important when the algorithm gets trapped in a local minimum, but finding all

local minimum is what is required from solving non-linear equation system. Hence, they shall not be utilized.

4.2. G3-PCX: generalized generation gap with parent-centric crossover

The generalized generation gap with parent centric crossover operator (G3-PCX) in genetic algorithms has been shown to outperform evolutionary strategies[14] and other real-coded genetic algorithms for some function optimization problems.[17]

The usual standard algorithm only produces one solution corresponding to a local minimum. In order to calculate all FKP solution, the main algorithm will launch a certain number of runs being selected larger then the maximum number of solutions. Through various experiments, the value is set to 100 which insures finding all solutions.

The standard algorithm has been adapted to solve the 6-6 FKP and is given in Algorithm 1. Its modularity allows any other Evolutionary algorithm implementation in a black box structure taking advantages of object oriented programming included in C++. By extension, this algorithm could even integrate any unconstrained single target optimization method.

Algorithm 1 The Basic GA Algorithm for solving the FKP

```
$optitype \leftarrow minimize$
$accuracy \leftarrow 1e - 80$
$MaxNumberFunctions \leftarrow 50000000$
$NumberKids \leftarrow 2$
$NumberParentsinCrossover \leftarrow 1$
$NumberParentsReplaced \leftarrow 2$
$sigmazeta \leftarrow 0.1$
$sigmaeta \leftarrow 0.1$
$MaxPopulationSize \leftarrow 1000$
$NumberVariables \leftarrow 9$
$MaxNumberRuns \leftarrow 100$
$NumberParents \leftarrow 2$
$gen \leftarrow MaxNumberFunctions/NumberKids$
Initialise population randomly
for $NumberRun \rightarrow NumberMaxRuns$ do
    Call the genetic algorithm subprogram (G3-PCX)
    Calculate average
    Calculate average error
end for
Return the best solution
```

In G3-PCX, the whole population is randomly initialized and evaluated similarly to the standard genetic algorithm. The difference lies in the optimization phase where a small sub-population is made of just a few chosen parents and children. At each generation, only the subset of the entire population is evaluated rather than evaluating the whole population, thereby saving on computation time. The children with their new fitness become part of the entire population. The proposed G3-PCX algorithm is shown in Algorithm 2.

Algorithm 2 begins by initializing the population and, in the first iteration, all the parents are chosen randomly. It continues by evaluating all the individuals in the population where the best fitness is retained.

One sub-population is then randomly created which has the size of number of individuals. The number of parents and children must be designated beforehand. The best parent is selected from the population and the rest of the parents are selected randomly. The selection of the best parent ensures elitism in the procedure. The children are created from the parents using the PCX crossover operator given in ref. [17]. The parents and the children are combined in the sub-population. Afterwards, the best two individuals are chosen from the combined sub-population which are further replaced in the population.

Algorithm 2 G3-PCX Evolutionary Algorithm

```
$tempfit \leftarrow 0$
$count \leftarrow 0$
Start clock
Initialise *population*
Evaluate fitness for all individuals in the *population*
$tempfit \leftarrow$ best fitness of the *population*
**for** ($i = 1 \rightarrow kids$ **do**
    Random setup of the *subpopulation*
    Find the weakest parents to be replaced by children
    Sort all parents according to fitness
    Choose the 2 best parents
    **for** $i \rightarrow NumberKids$ **do**
        Generate a child using the PCX reproduction operator
    **end for**
    Replace the chosen individuals in the *subpopulation*
    $tempfit \leftarrow$ best fitness of the *subpopulation*
**end for**
Stop clock
Compute number of cycles
Calculate error of the best individual
Return the best solution
```

The best individual in each sub-population is retained at each generation. The parent centric crossover operator is used in creating an offspring based on orthogonal distance between the parents. The parents are made of female and male components. The *female* parent points to search areas and the *male* parent is used to determine the extent of search of the areas pointed by the female. The genes of the offspring extract values from intervals associated in the neighborhood of the female and male using probability distribution. The range of this probability distribution depends on the distance among the genes of the male and the female parent. The parent-centric crossover operator assigns more probability to create and offspring near the female than anywhere else in the space. The genetic algorithm subprogram comprise the PCX reproduction operator and it is given in Algorithm 3.

An advantage of the parent-centric crossover operator is that it behaves like a mutation operator and at the same time retains diversity and is self-adaptive. They have been used as a hill-climbing local search procedure in a memetic based algorithm.[10] There is no mutation operator in the original G3-PCX algorithm. Amendments to the original algorithm have been done by proposing a mutation operator[12] which has shown good performance in multi-modal problems. Further amendments have been done in the parent-centric crossover operator by introducing a female and male differentiation process which determines the male and female individuals chosen from the population and by further using parent selection mechanisms shown in ref. [8].

5. Experiments, Results and Analysis

In this section, the G3-PCX algorithm is implemented for solving the FKP of the general 6-6 hexapod manipulator for the first time with in depth analysis. We have chosen one one difficult configuration producing 16 solutions. The fitness function is derived from the augmented IKP of the 6-6 parallel manipulator as shown in Eq. (20).

We aim to obtain a solving method giving the highest result accuracy which implies the least fitness values and the largest success rates, hopefully in all instances.

The following steps are taken in this section.

1. A difficult FKP example on a typical 6-6 hexapod is selected among a large number of trials which led to results being considered hard to obtain by other methods.

Algorithm 3 PCX reproduction operator algorithm

```
Initialize each variable centroid to 0.0
for $i = 1 \rightarrow NumVariable$ do
    for $j = 1 \rightarrow RandParent$ do
        Centroid[j] = Centroid[j] + Population(TempIndex[j])
        Centroid[j] = Centroid[j]/RandParent
    end for
end for
for $j = 1 \rightarrow RandParent$ do

    for $i = 1 \rightarrow NumVariable$ do
        Distance vector calculation of each solution from centroid
        Population difference vector calculation
    end for
end for
Distance vector magnitude calculation
for $i = 1 \rightarrow RandParent$ do
    Orthogonal direction calculations
    Orthogonal distance calculations $\leftarrow D[i]$
end for
Average of the perpendicular distances
for $j = 1 \rightarrow NumVariable$ do
    random vector calculation
    child calculation
    child inclusion in the new population
end for
Return the new child
```

2. In prior trials reported in ref. [30], we have shown that genetic algorithms can produce results which do not necessarily correspond to the equation system solutions. In order to properly verify the solutions from the G3-PCX algorithm, the FKP solutions are first calculated applying the exact certified algebraic method, based on proven Groebner bases, giving the certified results and described in ref. [28].
3. The best population size for the G3-PCX algorithm is evaluated.
4. The initial population is specifically analyzed for the G3-PCX algorithm. We find how each unique solution can be obtained by configuring the initial population within a specific range. At the same time, we also investigate if the G3-PCX can guarantee isolation of all specific real solutions.
5. Precision analysis will be performed comparing the solutions obtained by the G3-PCX algorithm with the exact solutions from the algebraic method.
6. We try a rational number model for the fitness function and compare its performance with the previous fitness function model.

The main performance measurements will be the following parameters:

1. The duration of the optimization in terms of number of function evaluations and success rate;
2. The solution quality in terms of error defined by the fitness function;
3. The solution accuracy calculated by the difference between the G3-PCX results and the exact ones.
4. The success rate.

The success rate determines how well the particular algorithm can guarantee a solution within a specified time. A run is considered successful if a desired solution defined by an error is found before the maximum time is reached in a predetermined number of independent experimental runs.

We use the Maple computer algebra environment for fitness function calculation and conversion into the C++ language. We produce two different expressions of the same fitness function having their coefficients respectively with floating numbers as *fitfloat* and with rational numbers as *fitrat*. The

Table I. Parallel manipulator configuration table.

Joint	Coordinates
OA_1	(464.141, 389.512, −178.804)
OA_2	(569.471, 207.131, −178.791)
OA_3	(529.050, −597.151, −178.741)
CB_1	(68.410, 393.588, 236.459)
CB_2	(375.094, −137.623, 236.456)
CB_3	(306.664, −256.012, 236.461)

Table II. FKP Exact Solutions with floating numbers, first 8 solutions.

No.	x_1, y_1, z_1	x_2, y_2, z_2	x_3, y_3, z_3
1	−127.706,505.931,−1273.639	−499.778,366.940,−806.217	−568.185,248.537,−806.229
2	112.871,418.003,1020.485	420.172,−112.850,1020.300	503.133,−143.725,916.075
3	−125.525,−426.751,561.805	−432.557,104.258,561.380	−560.748,109.088,608.733
4	185.128,−539.915,−966.722	760.175,−517.885,−1179.031	828.559,−399.469,−1178.894
5	−68.937,917.451,−1178.608	−375.587,430.711,−965.794	−307.150,322.201,−918.459
6	−68.194,917.281,821.484	−374.945,430.433,609.062	−306.530,321.9503336,561.6336491
7	68.521,−616.353,449.061	375.522,−363.774,916.174	307.163,−307.440,1020.347
8	−68.004,−393.741,637.180	−375.007,137.284,637.039	−306.649,255.714,636.994

Table III. FKP Exact Solutions with floating numbers, last 8 solutions.

No.	x_1, y_1, z_1	x_2, y_2, z_2	x_3, y_3, z_3
9	68.089,−616.271,−806.530	374.548,−363.341,−1273.809	306.068,−307.030,−1377.916
10	−68.754,−393.741,−994.300	−375.424,137.477,−994.155	- 306.991,255.864,−994.148
11	68.709,393.333,1006.993	375.311,−137.925,1006.846	306.862,−256.304,1006.876
12	67.619,393.333,−1364.239	374.705,−137.645,−1364.410	306.364,−256.086,−1364.382
13	−126.015,−426.719,−919.060	−432.927,104.359,−918.475	- 561.139,109.219,−965.768
14	185.713,−540.218,608.962	760.812,−518.160,821.128	829.173,−399.731,821.0175248
15	−126.884,505.339,916.535	−499.392,366.719,449.349	−567.849,248.346,449.363
16	112.014,418.519,−1377.831	419.187,−112.408,−1377.877	502.220,−143.302,−1273.715

fitness function can have its real coefficients approximated by either floating or rational numbers. In either case, no proof exist pointing towards the most performant approach.

The G3-PCX algorithm is programmed in C++ and the experiments are performed on an IBM compatible PC with Intel i3 core processors and Linux as the operating system. The performance and error calculations are performed with Mupad Pro.

The G3-PCX algorithm employs the mating pool size of 2 offspring and 2 parents with the generation gap model for selection for NSP and CoSyNE. The termination condition of the G3-PCX is either when

- the maximum number of function evaluation is reached which is fixed to 200000,
- the solution in the populations are almost the same with distance value of 1×10^{-80},
- the fitness function error has reached 1×10^{-20}.

5.1. Configuration for the FKP of 6-6 General Parallel Manipulator

We try one difficult FKP example on a typical 6-6 hexapod with 40 complex solutions out of which 16 real solutions can be extracted. Its configuration is given on Table I where the fixed base and mobile platform joint coordinates, OA_O, CB_C are given with units being the millimeter.

The prismatic actuator variables are set respectively to $l_i = 1250$ for $i = 1, \ldots, 6$. This example has been deliberately selected since this problem yields 16 real solutions as it was confirmed by an algebraic method giving the exact results.[28]

Table IV. Performance according to population size.

PopSize	No.Func.Eval	Fitness	CPU Time(s)	Success
100	144022	$8.79X10^{-18}$	0.74	100
	± 5894	$\pm 4.25X10^{-18}$	± 0.023	
150	74479	$3.67X10^{-19}$	0.39	100
	± 4123	$\pm 2.19X10^{-19}$	± 0.010	
200	63059	$5.58X10^{-20}$	0.34	100
	± 5354	$\pm 2.77X10^{-20}$	± 0.009	
250	56790	$2.18X10^{-20}$	0.31	100
	± 6251	$\pm 1.91X10^{-20}$	± 0.006	
300	57046	$6.98X10^{-20}$	0.32	100
	± 5789	$\pm 1.05X10^{-20}$	± 0.006	

5.2. *Exact solutions*

The exact algebraic method is implemented as a tool to verify and troubleshoot the proposed G3-PCX method. In Tables II and III, the 16 exact solutions are shown with floating numbers up to the third digit.

5.3. *The number of runs for isolating all solutions*

After many trials with run numbers ranging from 20 to 1000, we chose 100 runs which insured to obtain 100 percent confidence to find the 16 solutions and even up to 40 real solutions.

5.4. *An evaluation for the population size*

The results show that population size impacts computation performance. In this subsection, the impact of initial population sizes on the G3-PCX algorithm is investigated. Various population sizes are compared on hundreds of simulation runs. The selected population sizes are 100, 150, 200, 250 and 300. The goal of this work is to establish the population sizes which insure that the method reaches the smallest fitness function values, meaning the best convergence which is indirect way to determine accuracy. The success rate and response time are also performance criterions which will appear in the results.

Table IV shows G3-PCX method performance results. It contains the population size, the number of function evaluation for 100 simulation runs, the fitness value, the response time and the success rate. These parameters are also compiled in graphic format where Fig. 4 shows the number of function evaluations and CPU times according to population number and Fig. 5 gives the different fitness values according to population number.

The simulation method is heuristic in nature, meaning that performance will vary statistically, thereby justifying that the performance results are shown with the mean value and 95 percent confidence interval for the selected 100 experimental runs.

In all instances, very consistently over a large number of trials, the success rate is 100 %, meaning that for population ranging from 100 to 300, we can expect to isolate all solutions. The final fitness values and simulation run response times are diminishing (improving) when the population size increases. The simulation run response times stabilize when population is around 250. The number of function evaluations is a also a good performance indicator which will affect response time. The results show a number of function calculation minimum around a population of 150.

5.5. *Initial population analysis*

In robotic kinematics, solving an equation system has the goal to find all real roots.[21]

In trajectory planning and simulation, the problem becomes the isolation of the same single solution with the same initial population range. In real-time control or robot trajectory simulation, the initial population ranges are determined from the last trajectory position. The selected zone will be closer to one specific solution and further away from the alternate solutions. In some instances, it is well known that Newton's method does not always find the closest solution to an initial guess.

In this subsection, two questions are investigated:

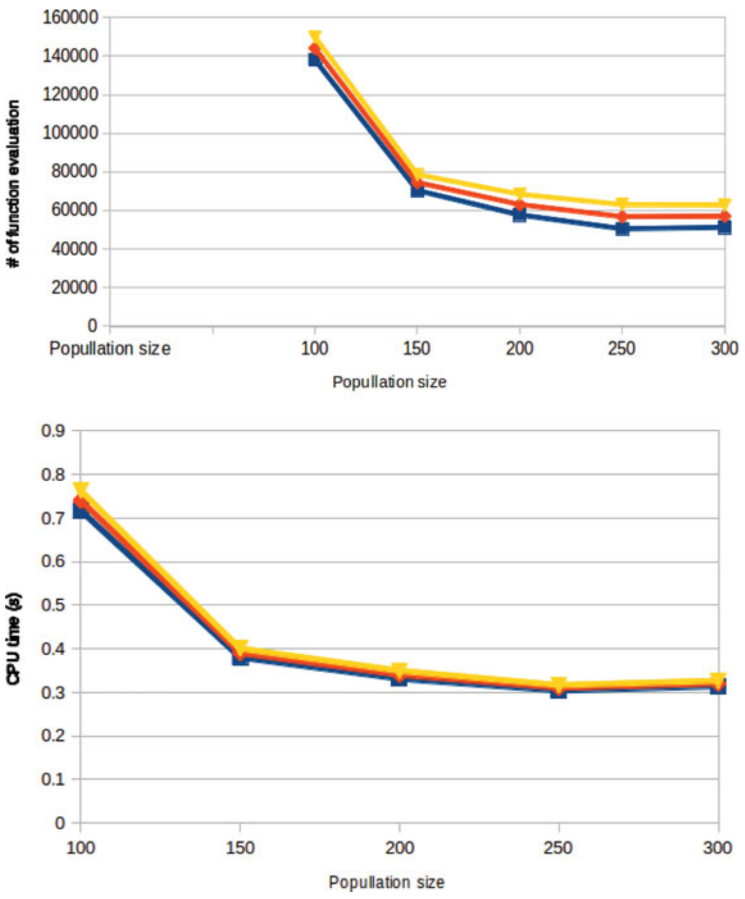

Fig. 4. The number of function evaluations and CPU times according to population number.

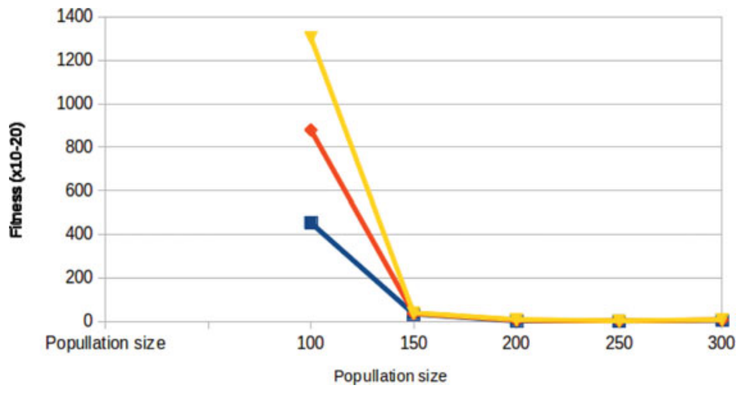

Fig. 5. The different fitness values according to population number.

Firstly, the impact of initial population ranges on the G3-PCX algorithm is investigated. We wish to determine if all specific solutions can be isolated from an initial population range in the vicinity of the root.

Secondly, the initial population range is increased to encompass all exact solutions and we wish to determine the number of runs necessary to find all solutions.

Numerical methods require an initial solution, corresponding to the initial guess encountered in numerical methods. Similarly, EAs need a population of initial solutions which are randomly generated. In order to verify all solution isolation capability of the G3-PCX algorithm, we have prepared several trial runs with initial populations which were set in specific variable ranges to isolate one specific solution. These range values are shown in Tables V and VI.

Table V. Ranges for initial populations, for solution 1 to 8.

No.	x_1, y_1, z_1	x_2, y_2, z_2	x_3, y_3, z_3
1	−150,505.7,−1300	−600,366.9,−850	−700,248.5,−850
	−127.3,600,−1250	−499.7,380,−400	−568.1,250,−300
2	112.5,410,1015	420,−120,1015	503,−170,880
	130,418.2,1200	500,−112.8,1500	600,−143.6,950
3	−125.7,−450,520	−432.7,50,500	−560.9,50,580
	−100,−426.73,575	−410,104.27,570	−500,109.1,620
4	160,−540,−970	700,−517.9,−1200	700,−399.5,−1200
	185.3,−500,−950	760.3,−500,−1100	828.7,−360,−1150
5	−80,917.4,−1200	−390,430.6,−970	−400,322.1,−930
	−68.9,1000,−1150	−375.51,500,−950	−307.1,400,−900
6	−68.4,800,800	−374.96,410,590	−306.58,300,520
	−68.1,917.3,850	−300,430.5,615	−305.4,322,570
7	68.2,−700,200	374.5,−400,880	307.1,−340,1015
	68.6,−616.3,480	380,−363.7,950	350,−307.3,1500
8	−68.1,−393.7417,620	−375.2,130,625	−306.7,252,630
	−50,−100,700	−375,137.3,700	−306.6,255.75,700

Table VI. Ranges for initial populations, for solution 9 to 16.

No.	x_1, y_1, z_1	x_2, y_2, z_2	x_3, y_3, z_3
9	67.8,−616.3,−850	300,−363.5,−1300	200,−307.1,−1500
	68.2,−600,−400	374.6,−300,−1250	306.1,−290,1372
10	−68.8,−393.7417,−1050	−375.49,137.4,−1000	−350,255.8,−1000
	−68.5,−100,−980	−375.3,200,−985	−306.9,270,−990
11	68.6,100,980	374.2,−200,980	306.7,−270,970
	100,393.3331,1010	375.35,−137.8,1010	306.9,−256.2,1010
12	50,393.3333,−1369	374.65,−137.7,−1370	306.3,−256.1,−1369
	67.8,400,−1350	374.9,−130,−1350	306.5,−220,−1330
13	−126.4,−426.72,−930	−450,104.32,−930	−563,109.2,−980
	−125.8,−410,−900	−432.8,110,−900	−561,150,−950
14	185.5,−550,590	760.6,−600,800	829,−500,800
	200,−540.15,610	900,−518.1,850	900,−399.7,850
15	−127,500,900	−499.5,300,200	−567.9,200,100
	−126.5,505.5,950	−480,366.8,470	−566,248.4,470
16	100,418.3,−1500	400,−112.5,−1500	450,−143.5,−1300
	112.5,450,−1375	419.5,−50,−1372	502.5,−100,−1250

Each line of Tables V and VI comprises one specific solution indicated by the solution number in the first column corresponding to the solution number on the exact solution Tables II and III. All nine output variables x_i, y_i, z_i for $i = 1, 2, 3$ are related to Eq. (20) with floating point numbers.

At the same time, these tests will allow to examine the following question: can the G3-PCX algorithm find all real solutions? Finally, the underlying question concerns extraneous solutions which are known to be found by some numerical methods. In a sense, this process leads to determine the circumstances and conditions which ensure that the proposed G3-PCX method is applicable.

5.5.1. Solution isolation with restricted ranges. It is essential to determine if the G3-PCX algorithm can isolate each solution. The selected verification procedure involves preparing as many as 16 tests, corresponding to one test for each distinct solution. In this test procedure, the initial population zones are then manually and carefuly selected to encompass one and only one solution. From the knowledge

Table VII. FKP Solutions with the G3-PCX algorithm.

No.	x_1, y_1, z_1	x_2, y_2, z_2	x_3, y_3, z_3
1	−127.162,505.756,−1273.17	−499.804,366.797,−806.191	−568.166,248.369,−806.251
2	112.655,417.05,1021.28	420.437,−113.526,1021.04	503.509,−144.266,916.862
3	−125.705,−427.868,561.915	−431.976,103.579,563.519	−560.131,108.428,610.969
4	185.472,−539.987,−966.61	761.004,−518.48,−1177.66	829.468,−400.111,−1177.44
5	−68.1685,917.387,−1178.05	−374.843,430.066,−966.607	−306.411,321.543,−919.296
6	−68.3187,917.379,821.976	−375.049,430.181,610.328	−306.629,321.626,563.072
7	67.7181,−616.597,449.881	373.604,−363.006,917.177	304.996,−306.896,1021.31
8	−68.2931,−394.147,637.993	−375.098,136.994,638.259	−306.695,255.398,638.348
9	68.116,−616.726,−805.925	374.52,−363.503,−1273.08	306.027,−307.284,−1377.23
10	−68.3917,−394.265,−993.816	−375.024,136.976,−994.343	−306.582,255.358,−994.447
11	68.3473,392.975,1007.72	375.18,−138.15,1007.67	306.783,−256.559,1007.7
12	68.204,392.804,−1363.68	375.287,−138.177,−1363.68	306.946,−256.617,−1363.69
13	−125.675,−427.508,−918.336	−432.246,103.767,−919.18	−560.425,108.599,−966.568
14	185.446,−539.781,610.843	761.051,−518.405,821.702	829.584,−400.076,821.349
15	−127.172,506.525,917.082	−499.754,367.052,450.21	−567.942,248.523,450.283
16	112.496,418.835,−1377.19	419.065,−112.442,−1377.16	502.1,−143.394,−1273.01

of the exact solutions which we did compute beforehand with the algebraic method, selection of the zone intervals is achieved by identifying the midpoints between the solutions and by keeping the zones to be mutually exclusive.

Each test includes 100 independant runs per distinct solution. In one test, each run starts with a different initial population randomly determined from the same initial zones. The solution results are shown in Table VII.

For several trials utilizing the initial exclusive zones, the G3-PCX algorithm worked properly, meaning that, for each zone interval, the solution herein contained has be isolated in all instances.

These test results show that the 16 solutions can be isolated with 100 % success.

5.5.2. Solution isolation with the global range. In another series of tests, we wish to determine the number of runs necessary for obtaining all solutions. In those cases, the initial population zone or range is set to a large interval range being [−2000, 2000] for each of the nine variables. This interval range should be selected to encompass the robot workspace which is limited by prismatic actuator motion stops. Again, with the knowledge of the exact solution location, this range is calculated to exceed the solution space containing all solutions.

The results show that in one hundred G3-PCX runs, all 16 solutions are determined at a 100 % rate, however this value is an upper limit since the run number is often inferior and recent tests showed good results on 40 runs. Several trial repetitions confirm the calculation of these solutions within that number of runs. The heuristic nature of EAs mean that often all solutions will be isolated with a lesser number runs. The consequence of that observation is that the same solutions can be found through several runs. A final consequence of this observation is that the run number value indirectly indicates the guaranteed maximum response time.

5.6. *Precision analysis*

One important performance criteria is solution precision. Convergence techniques such as EAs and numerical methods are converging towards a solution but rarely reaching it. EAs and numerical analysis offer no direct way to evaluate precision since exact solutions may only be located by theorems inside some convergence zones which actually become an upper bound. In our tests, for each solution, we calculate the exact precision by subtracting the solution isolated with the G3-PCX from the exact solutions.

In Table VIII, the first series of solutions were calculated from the implementation of a model computed with floating numbers as parameters and function coefficients.

With $i = 1, 2, 3$, we give eB_i, the error given in millimeter on OB_i mobile platform joint positions and $\overline{eB}$, which respectively correspond to variables, meaning $eB_i = [x_i, y_i, z_i]$ for $i = 1, 2, 3$. Then, the table last column presents $\overline{eB}$, the error average on the three mobile platform joint positions.

Table VIII. Error for each platform point with floating parameters.

No.	eB_1	eB_2	eB_3	$\overline{eB}$
1	0.73992	0.14787	0.17096	0.35292
2	1.26015	1.03570	1.02528	1.10704
3	1.13614	2.31790	2.41161	1.95522
4	0.36882	1.70918	1.83115	1.30305
6	0.95263	1.27788	1.29625	1.75588
7	0.51660	1.29483	1.47778	1.09640
8	1.17322	2.29729	2.43306	1.96786
9	0.95292	1.25709	1.39113	1.20038
10	0.75775	0.74757	0.73255	0.74596
11	0.80032	0.66853	0.71643	0.72843
12	0.88708	0.86333	0.86535	0.87192
13	0.96686	1.07463	1.04848	1.02999
14	1.12369	1.14564	1.23890	1.16941
15	1.94914	0.66718	0.62974	1.08202
16	1.33658	0.99072	0.94103	1.08944

Table IX. Error for each platform point with rational parameters.

No.	eB_1	eB_2	eB_3	$\overline{eB}$
1	0.7375629108	0.1386186354	0.1690542995	0.3484119486
2	1.252548519	1.028132266	1.020792232	1.100491006
3	1.139760648	2.321422796	2.418990832	1.960058092
4	0.3702850312	1.704354739	1.816770184	1.297136651
5	0.9503243866	1.272803737	1.293239371	1.172122498
6	0.5172606852	1.290747698	1.478021555	1.095343313
7	1.170887266	2.266442008	2.39724913	1.944859468
8	0.9506282254	1.257050363	1.39603738	1.201238656
9	0.7554266616	0.7506488831	7359806518	0.7473520655
10	0.8055514798	0.6706625755	0.7189587033	0.7317242529
11	0.8877348993	0.8637904751	0.8647654186	0.872096931
12	0.9658938612	1.072517076	1.047978414	1.02879645
13	1.130421895	1.150749328	1.246737325	1.175969516
14	1.936249033	0.6650298119	0.626748433	1.076009093
15	1.375549276	0.9963190726	0.9478332275	1.106567192
16	0.8639565828	0.7305174016	0.7142587355	0.7695775733

The index numbers correspond to the former tables of solutions. Joint errors do range between 0.15 and 2.5 mm with an average of 1.1644 mm. These accuracies would be sufficient for most material handling applications.

The same FKP can be prepared with the exact model implementing parameters selected as rational numbers instead. The fitness function is then converted to C++ by the specific Maple functions including the numerators and denominators as integers without any form of truncation. In Table IX, the second series of solutions are compared with the exact ones to calculate precision.

The error levels are showing little precision improvement or worstening.

5.7. *Performance comparison between parameter formats*

For each of the 16 solutions found by the G3-PCX algorithm, performance can be evaluated for the two cases where we apply the fitness model with parameters being either floating numbers or rational numbers. Table X is divided into two parts where the results are compiled as the population size, the number of function evaluation for 200 simulation runs, the fitness value, the response time and the success rate. The first two lines show the results for the fitness function written with floating numbers and the two last lines with rational numbers.

Table X. An evaluation of the best method for the FKP.

Parameters	PopSize	No.Func.Eval	Fitness	CPU Time(s)	Success
fit-float	200	63059	5.58e-20	0.34	100
		± 5354	± 2.77e-20	± 0.009	
fit-rat	200	86505	1.80749e-19	0.48	100
		± 4685	± 1.4717e-19	± 0.02	100

The success rates are always 100 % in all instances. The solution isolation completion times are included in the interval [0, 33, 0, 50]. The CPU response time result indicates that the application of rational parametering in the model does lead to slightly longer response times by about 0.15 s, being the result of a larger number of function evaluations without doubt. With a population size of 200, the application of rational numbers bring fitness functions being twice larger.

5.8. Discussion

Any solution can be isolated from an initial population randomly selected in the vicinity of that root. The G3-PCX algorithm has confirmed its ability to always find the closest solution if the problem is not singular. The 16 solutions were computed with 100 % success. For the first time in kinematics, one genetic algorithm has now shown its prowess to isolate any distinct FKP solution.

The G3-PCX produces response times of around 0,34 s for one solution, slower than numerical methods, but faster then algebraic methods or interval analysis.

Over a large number of trials, the success rate is 100 %, meaning that for population ranging from 100 to 300, we can expect to isolate all solutions. The final fitness values and simulation run response times reach a minimum at 150.

The main advantage of optimization techniques such EAs and especially the G3-PCX algorithm is that it allows handling equation systems with more variables then the number of equations, thereby opening to the perspectives of introducing error, calibration or even tolerance parameters on each variables and configuration parameters. In equation solving, all numerical and algebraic methods require that the number of equation be equal to the number of variables. That explains why optimization methods are implemented to solve calibration problems since a larger number of variables does not prevent solving the optimization problem.

It is actually not possible to solve equation systems which do contain more variables then equations with the numerical and algebraic solving methods described in the introduction.

As a significant advantage, the G3-PCX method does isolate all solutions of the general 6-6 parallel robot FKP in all instances that were tested. One hundred runs will obtain all FKP solutions with a total response time below 35 s. This response time allows to perform assembly mode analysis when dealing with single FKP, but also with trajectory simulation in robotic path planning. In the author's knowledge, it is the first time that an EA has been adapted to find all the solutions to a non-linear equation system as complex as the FKP of an hexapod parallel robot. This method is then the fastest method to find all solutions in all instances since the Gröbner based methods and interval analysis methods implementing the Newton method easily reach response times exceeding the minute.

Practical evidence of the effectiveness of the G3-PCX algorithm can be confirmed by solution of the various FKP examples that were attempted. Another important proof of the effectiveness and reliability of the method is that it does not find extraneous solutions.

This method allows to solve kinematics without matrix inversions nor Jacobian computations as it is encountered in some numerical methods. Unlike numerical methods, the computation errors are only cumulative over one isolation iteration.

As limitations, with small fitness functions, we can observe that the resulting solution precision is sufficient for material handling but would surely not allow application to high precision tasks such milling or surgery. According to the solution, the average platform position error ranges from 0.352 to 1.968 with an average of 1,164 meaning a certain variety of precision depending on the solution. Improving solution precision is remaining an open question for further research. In order to improve precision, the G3-PCX algorithm could be complemented by another method to refine the solution.

As a clear disadvantage, the G3-PCX algorithm, like any genetic algorithm, remains not as fast as most numerical methods such as the Newton method, even when one single solution is isolated.

There exists no formal proof to guarantee that the G3-PCX will always find all solutions in any case.

Computing other parallel manipulators where knowledge solution location may be difficult to obtain, could become a challenge.

Solving with the rational number model does not really affect solution accuracy but results in slightly slower response times.

The reduction of the number of runs could result is smaller cycle times. Further research should investigate this issue.

As perspectives, the G3-PCX algorithm could be studied in path planning and simulation procedures where only one solution is isolated in order to follow the selected path.

6. Conclusion

The use of the parent-centric crossover operator with the generalized generation gap is effective for solving the forward kinematics of the general parallel manipulator. The method calculated all distinct real solutions to a level of accuracy suitable for material handling over several trials. Utilizing an appropriate position based model with a set of carefully selected constraints implementing kinematics chain distances and mobile platform configuration distances, the G3-PCX algorithm can be applied to solve the non-linear equation systems encountered in parallel manipulator kinematics. The method can also isolate one specific solution from an appropriately selected initial population range.

Satisfactory response times of the optimization process motivate method implementation into simulation packages with off-line path planning. Future work should investigate means to augment precision in order to allow for trajectory pursuit through hybridation with other solving or optimization techniques.

Future work could also investigate the application of complete kinematics model with calibration errors and even tolerance synthesis.

Moreover, multi-objective evolutionary algorithms can also be envisioned to solve optimization problems where the FKP has to be solved in calibration or trajectory analysis where task feasibility is studied.

References

1. R. Boudreau and N. Turkkan, "Solving the forward kinematics of parallel manipulators with a genetic algorithm," *J. Robotic Systems* **13**(2), 111–125 (1996).
2. G. Capi and K. Doya, "Evolution of recurrent neural controllers using an extended parallel genetic algorithm," *Robot. Auton. Syst.* **52**(2-3), 148–159 (2005).
3. R. Chandra and L. Rolland, "Hybrid meta-heuristics for the forward kinematics of 3-rpr planar parallel manipulator," *Appl. Math. Comput.* (2011).
4. R. Chandra and L. Rolland, "Hybrid meta-heuristics for the forward kinematics of 3-RPR planar parallel manipulator," *Appl. Math. Comput.* **2**17(22):8997–9008, 2011.
5. P. Dietmaier, "The stewart-gough platform of general geometry can have 40 real postures," *Adv. Robot Kinematics* **1**, 7–16 (1998).
6. R. Parrish, E. Dieudonné and R. Bardusch, An actuator extension transormation for a motion simulator and an inverse transformation applying newton-raphson's method. *Technical Report* D-7067 (NASA, Washington, 1972).
7. A. Omran, *et al.* "Genetic algorithm based optimal control for a 6-dof non redundant stewart manipulator," *Int. J. Mech. Syst. Sci. Eng.* **2**(2), 73–79 (2008).
8. C. Garcia-Martinez, *et al.* "Global and local real-coded genetic algorithms based on parent-centric crossover operators," *Eur. J. Oper. Res.* **185**(3), 1088–1113 (2008).
9. J. D. Schaffer, *et al.* "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization," *Proceedings of the Third International Conference on Genetic Algorithms*, San Francisco, CA, USA (Morgan Kaufmann Publishers Inc., 1989) pp. 51–60.
10. M. Lozano, *et al.* Real-coded memetic algorithms with crossover hill-climbing. *Evolutionary Computation* **12**(3), 273–302 (2004).
11. Sheng, *et al.* "Forward Kinematics of the Stewart Platform Using Hybrid Immune Genetic Algorithm," *Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation* (Jun. 2006) pp. 2330–2335.
12. T. Jason, *et al.* "Harnessing Mutational Diversity at Multiple Levels for Improving Optimization Accuracy in g3-pcx," *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE (2007) pp. 4502–4507.

13. V. E. Gough and S. G. Whitehall, “Universal Tyre Test Machine,” *Proceedings of 9th International Technical Congress FISITA*, Mai (1962), pp. 117–135.
14. N. Hansen and A. Ostermeier, “Completely derandomized self-adaptation in evolution strategies,” *Evol. Comput.* **9**(2), 159–195 (2001).
15. J. H. Holland, *Adaptation in Natural and Artificial Systems* (MIT Press, Cambridge, MA, USA, 1992).
16. M. Husty, “An algorithm for solving the direct kinematic of stewart-gough-type platforms,” *J. Mech. Mach. Theory* **31**(4), 365–379 (1996).
17. A. Anand, K. Deb and D. Joshi, “A computationally efficient evolutionary algorithm for real-parameter optimization,” *Evol. Comput.* **10**(4), 371–395 (2002).
18. D. Lazard, “On the representation of rigid-body motions and its application to generalized platform manipulators,” *Comput. Kinematics* **1**, 175–181 (1993).
19. D. Lazard, “On the representation of rigid-body motions and its application to generalized platform manipulators,” *J. Comput. Kinematics* **1**(1), 175–182 (1993).
20. H. MacCallion and D. T. Pham, “The Analysis of a Six Dof Workstation for Mechanized Assembly,” *Proceedings of the 5th World Congress on Theory of Machines and Mechanisms* (1979) pp. 611–616. Montréal.
21. J. P. Merlet. *Les Robots parallèles*, 2nd ed. (Robotique. Hermès, Paris, traité des nouvelles technologies edition, 1997).
22. J. P. Merlet, “Solving the forward kinematics of a gough-type parallel manipulator with interval analysis,” *Int. J. Robot. Res.* **23**(3), 221–235 (Mar. 2004).
23. M. Mitchell, *An Introduction to Genetic Algorithms*.
24. B. Mourrain, “The 40 Generic Positions of a Parallel Robot,” *Proceedings of the ISSAC’93* (1993) pp. 173–182.
25. S. Darenfed, R. Boudreau and N. Turkkan, “étude comparative de trois nouvelles approches pour la résolution du problème géométrique direct des manipulateurs parallèles,” *J. Mech. Mach. Theory* **33**(5), 463–477 (1998).
26. M. Raghavan, “The stewart platform of general geometry has 40 configurations,” *J. Mech. Design, Trans ASME* **115**(2), 277–282 (1993).
27. M. Raghavan and B. Roth, “Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators,” *Trans. ASME* **117**, 71–79 (June 1995).
28. L. Rolland, “Certified solving of the forward kinematics problem with an exact method for the general parallel manipulator,” *Adv. Robot.* **19**(9), 995–1025 (2005).
29. L. Rolland, “Certified Solving and Synthesis on Modeling of the Kinematics Problems of Gough-Type Parallel Manipulators with an Exact Algebraic Method,” **In**: *Parallel Manipulators, Towards New Applications* (I-Tech Education and Publishing, 2008).
30. L. Rolland and R. Chandra, “Forward Kinematics of the 6-6 General Parallel Manipulator Using Real Coded Genetic Algorithms,” *Proceedings of the IEEE/ASME Conference on Advanced Intelligent Mechatronics (AIM 2009)*, page In Press (2009).
31. L. Rolland and R. Chandra, *The Application of an Efficient Evolutionary Algorithm in Solving the Forward Kinematics of 6-6 General Parallel Manipulator.*, volume 524 of *CISM International Centre for Mechanical Sciences*, (Springer, Berlin, 2010) pp. 117–124. ROMANSY 18 - Robot Design, Dynamics and Control.
32. F. Ronga and T. Vust, “Stewart Platforms Without Computer?,” *Proceedings of the International Conference on Real, Analytic and Algebraic Geometry* (1992) pp. 197–212.
33. J. M. Selig, *Introductory Robotics* (Prentice-Hall, Hemel Hemstead, 1992) pp. 157.
34. D. Stewart, “A platform with 6 degrees of freedom,” *Proceedings of the Institution of Mechanical Engineers* **180**, 371–386 (1965).
35. O. Altazurra, V. Petuya, A. Alonso and A. Hernandez, “Resolution of the Direct Position Problem of the Parallel Kinematic Platforms Using the Geometric-Iterative Method,” *Proceedings of the IEEE International Conference on Robotics and Automation, Barcelona* (2005) pp. 3255–3260.
36. C. W. Wampler, “Forward displacement analysis of general six-in-parallel sps (stewart) platform manipulators using soma coordinates,” *Mech. Mach. Theory* **31**(3), 331–337 (1996).
37. A. H. Wright, “Genetic Algorithms for Real Parameter Optimization,” **In**: *Foundations of Genetic Algorithms* (Morgan Kaufmann, 1991) pp. 205–218.
38. M. Hao, X. Wang and Y. Cheng, “On the use of differential evolution for forward kinematics of parallel manipulators,” *Appl. Math. Comput.* **205**(2), 760–769 (2008).