2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics
Suntec Convention and Exhibition Center
Singapore, July 14-17, 2009

Forward Kinematics of the 6-6 general Parallel Manipulator Using Real Coded Genetic Algorithms

Luc Rolland
Mechanical Engineering dept.
Middle East Technical University, North Cyprus Campus
Kalkanli, Turkish Republic of North Cyprus
Luc.Rolland@gmail.com or luc@metu.edu.tr

Rohitash Chandra
Information Technology
Middle East Technical University, North Cyprus Campus
Kalkanli, Turkish Republic of North Cyprus
Rohitash_c@yahoo.com

Abstract—**This article examines an optimization method to solve the forward kinematics problem (FKP) applied to parallel manipulators. Based on Genetic Algorithms (GA), a non-linear equation system solving problem is converted into an optimization one. The majority of truly parallel manipulators can be modeled by the 6-6 which is an hexapod constituted by a fixed base and a mobile platform attached to six kinematics chains with linear (prismatic) actuators located between two ball joints. Parallel manipulator kinematics are formulated using the explicit Inverse Kinematics Model (IKM). The position based equation system is implemented. In order to implement GA, the objective function is formulated specifically for the FKP using one squared error performance criteria applied on the leg lengths augmented by three platform joint distances. The proposed approach implements an elitist selection process where a new mutation operator for Real-Coded GA is analyzed. These experiments are the first to converge towards several exact solutions on a general Gough platform manipulator with fast convergence.**

Index Terms—**Parallel robot, Gough platform, spatial parallel manipulator, forward kinematics, position based model, genetic algorithms, mutation-based operators, elitist selection, real root isolation.**

I. INTRODUCTION

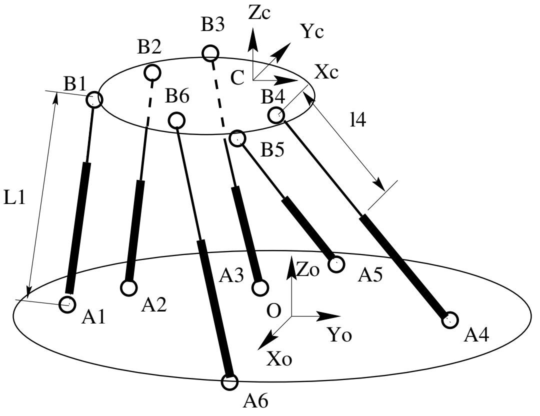

Figure 1. The general spatial manipulator as the typical **6-6** hexapod

The truly parallel manipulator is defined as an hexapod constituted of a fixed base, a mobile plateform where the end-effector is mounted and six kinematics chains, fig. (1). The platforms are attached through universal or ball joints, respectively with 2 or 3 DOFs. Each kinematics chain contains one 1 DOF prismatic joint.

Ronga, Lazard 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, [1]–[3]. From en 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. Fast numerical approaches usually implement Newton's method, however it is sometimes plagued by Jacobian inversion problems and numerical instabilities. Resultant or dialytic elimination might add spurious solutions, [4]. Homothopy methods are prone to miss some solutions, [4]. In the majority of parallel manipulator cases, the **FKP** is a difficult problem, [13]. Therefore, this justifies the implementation of another method which could find solutions numerically and rapidly.

The genetic algorithms introduced by Holland have evolved significantly in order to suit real-world optimization challenges faced by engineers, [5]. Evolutionary algorithms have been applied for solving the **FKP** of simple parallel manipulators. However, the later is a classical problem of finding all solutions to a non-linear equation system, whereas GAs solve optimization problems. Hence, there exists a difficulty to derive an optimization problem from a root finding one. In the first, we are only interested by one maximum in the later, we calculate all solutions. A real-coded genetic algorithm (RCGA) was proposed, [7] integrating crossover and mutation operators inspired by operators used in binary GA. It was reported that the GA method is more time consuming than Newton-Raphson's method. However, it was shown that the domain in which the GA will converge to a solution is larger allowing process launch with a more distant initial guess. Recently, genetic algorithms have succeeded to solve the **3-RRR**, [8], and the **SSM** Gough platform, [9], [10]. However, in [9], one binary coded genetic algorithm is implemented to find one solution of the **SSM** manipulator. Solving the **FKP** for the general Gough platform has never been yet attempted.

In this work, the RCGA uses Wright's heuristic crossover operator with different mutation operators, [11]. The Pivot Mutation operator is introduced. The RCGA uses roulette

978-1-4244-2853-3/09/$25.00 ©2009 IEEE

wheel selection combined with the elitist strategy in order to avoid oscillation during the search. Furthermore, as a first, the general Gough platform **FKP** results obtained by the RCGA are verified with the exact ones obtained from a proven Gröbner based method implemented on computer algebra [12].

This article is presented as follows: Section I addresses the issues involved in kinematics modeling of the general Gough platform and the equation solving problem conversion to an optimization problem. Section II reviews prominent and recently proposed real-coded genetic operators and furthermore gives details of the proposed Pivot Mutation operator along with other operators used in this study. Section 3 presents the results obtained on a **6-6** hexapod with 16 real solutions with conclusion and directions for future work.

II. The Forward Kinematics Problem Formulation

A. *The kinematics of parallel manipulators*

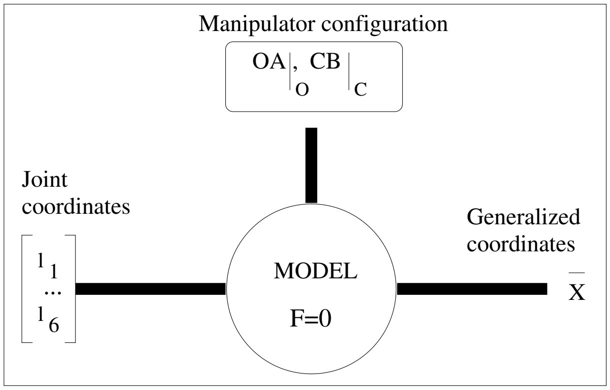

Figure 2. Kinematics model

Any manipulator is characterized by its mechanical configuration parameters and the posture variables. The configuration parameters are thus $\mathbf{OA}_O$, the base attachment point coordinates in O (the base reference frame), and $\mathbf{CB}_{|C}$, the mobile platform attachment point coordinates in C (the mobile platform reference frame). 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 $\overrightarrow{X}$, the end-effector position and orientation.

The kinematics model is an implicit relation between the configuration parameters and the posture variables, $F(\overrightarrow{X}, \overline{L}, \mathbf{OA}_{|O}, \mathbf{CB}_{|C}) = 0$ where $\overline{L} = \{l_1, \ldots, l_6\}$.

This article shall only investigate the forward kinematics problem (**FKP**), Fig. 2. Usually the *inverse kinematics problem* is required to model the **FKP** and is defined as: *given the generalized coordinates of the manipulator end-effector, find the joint positions.*

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

B. *Vectorial formulation of the basic kinematics model*

Containing as many equations as variables, vectorial formulation constructs an equation system, [14], as a closed vector cycle between the following points: A_i and B_i, kinematics chain attachment points, O the fixed base reference frame and C the mobile platform reference frame. For each kinematics chain, an implicit function $\overrightarrow{A_iB_i} = U_1(X)$ can be written between joint positions A_i and B_i. Each vector $\overrightarrow{A_iB_i}$ is expressed knowing the joint coordinates $\overrightarrow{l_i}$ and X giving function $U_2(X, \overrightarrow{L})$. The following equality has to be solved: $U_1(X) = U_2(X, \overrightarrow{L})$. 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 $\overrightarrow{OC}_{|O}$ and then one platform general rotation expressed by the rotation matrix $\mathcal{R}$. Vectorial formulation 2 evolves as a displacement based equation system using the following relation :

$$\overrightarrow{A_iB_i}_{|O} = \overrightarrow{OC}_{|O} + \mathcal{R}\overrightarrow{CB_i}_{|C} - \overrightarrow{OA_i}_{|O} \tag{1}$$

For each distinct platform point $\overrightarrow{OB_i}_{|O}$ with $i = 1, ..., 6$, each kinematics chain can be expressed using the distance norm constraint, [15]:

$$L_i^2 = ||A_iB_i||^2 \tag{2}$$

C. *The inverse kinematics problem*

In 3D space, any rigid boby can be positioned by three distinct points. Every variable have then the same units and their range is equivalent leading to same weight. Hence, the rotation impact is included into the point parameters and made equivalent to the translation impact. The main disadvantage is the unknown number exceeding the end-effector DOF number, [16].

The three platform distinct points are usually selected as the three joint centers B_1, B_2, B_3. The nine variables are set as : $\overrightarrow{OB_i}_{|O} = [x_i, y_i, z_i]$ for $i = 1 \ldots 3$. To simplify calculations, one reference frame R_{b_1} is precisely located on B_1. The unit vectors u_1, u_2 and u_3 represent the new frame axes and are defined by:

$$u_1 = \frac{\overrightarrow{CB_1CB_2}_{|O}}{||\overrightarrow{CB_1CB_2}_{|O}||}, \; u_2 = \frac{\overrightarrow{CB_1CB_3}_{|O}}{||\overrightarrow{CB_1CB_3}_{|O}||}, \; u_3 = u_1 \wedge u_2 \tag{3}$$

Knowing that the platform is supposed infinitely rigid, any platform point M can be expressed

$$\overrightarrow{B_1M} = a_M u_1 + b_M u_2 + c_M u_3 \tag{4}$$

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 $a_{B_i}, b_{B_i}, c_{B_i}$, $i = i \ldots 6$ and can explicitly be deduced from $\mathbf{CB}_C$ by solving the following linear system of equations :

$$\overrightarrow{B_1B_i}_{|R_{b_1}} = a_{B_i}u_1 + b_{B_i}u_2 + c_{B_i}u_3 \; , \; i = 1 \ldots 6. \tag{5}$$

Using the relations, equ. (5), the distance constraint equations $l_i^2 = ||\overrightarrow{A_iB_{iO}}||^2$, $i = 1 \ldots 6$ 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:

$$\begin{aligned} l_i^2 &= (x_i - OA_{ix})^2 + (y_i - OA_{iy})^2 \ , \ i = 1 \dots 3 \quad (6) \\ l_i^2 &= ||\overrightarrow{B_k}_{|R_{b_1}} - \overrightarrow{OA}_{k|R_f}||^2 \ , \ i = 4 \dots 6 \quad (7) \end{aligned}$$

D. The Forward Kinematics Problem

The **IKP** expression gives an algebraic system comprising the first six equations in terms of three point variables : $x_1, y_1, z_1, x_2, y_2, z_2, x_3, y_3, z_3$, equation (7). This system contains trigonometric functions which can be handled by the numerical solvers implemented in GA.

E. The conversion to an optimization problem

GAs being only applicable to optimization problems, the root finding problem is converted into an optimization problem. The inverse kinematic model is implemented from which is derived an objective function, also called *fitness* function which is calculated on each **FKP** estimation representing the total error on each leg lengths. Let lg_i be the leg length of kinematics chain i which is given as input of the problem. If we set $H_i = l_i^2$ coming from equation (7), the fitness function is set to :

$$\sum_{i=1}^{6} (sqrt(H_i) - lg_i)^2 \quad (8)$$

This last objective function includes the combination of six individual objectives being the kinematics chain lengths. Preliminary tests with equ. 8 led to several solutions which were NOT in correspondence with the exact proven ones. Hence, this function needed to be augmented by one constraint set : the platform fixed distances between the three selected joint points : B_1, B_2 and B_3 distinct points provide for three functions.

$$\begin{aligned} G_1 &= ||\overrightarrow{B_2 B_1}_C||^2 - (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 \\ G_2 &= ||\overrightarrow{B_3 B_1}_C||^2 - (x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2 \\ G_3 &= ||\overrightarrow{B_3 B_2}_C||^2 - (x_3 - x_2)^2 + (y_3 - y_2)^2 + (z_3 - z_2)^2 \end{aligned} \quad (9)$$

Three distances can be calculated using two characteristic platform vector norms between the B_1, B_2 distinct points and the B_1, B_3 ones. The last value comes from these vector multiplication.

$$\begin{aligned} d_1 &= ||\overrightarrow{B_2 B_1}_C||^2 \\ d_2 &= ||\overrightarrow{B_3 B_1}_C||^2 \\ d_3 &= ||\overrightarrow{B_3 B_2}_C||^2 \end{aligned} \quad (10)$$

Hence, the objective function becomes :

$$\sum_{i=1}^{3} (sqrt(H_i) - lg_i)^2 + \sum_{k=1}^{3} (G_i - d_i)^2 \quad (11)$$

The resulting objective function includes then the nine single objectives obtained from the norm of kinematics chain lengths and platform distance constraints. Each distance objective results in similar impact on the objective function result, since the position-based equations allow to keep the same sensitivity to all objectives.

III. SOLVING WITH GENETIC ALGORITHMS

A. Real Coded Genetic Algorithms

Initial guesses constitute a number of possible solutions, namely chromosomes, to build a population. The objective function is evaluated for each chromosome where we implement genetic operators like selection, crossover and mutation for producing new solutions, called offsprings, which are added to the population. The process is repeated until the algorithm obtains the optimal solution. Careful selection of parent chromosomes is done using one selection operator. The optimization procedure of a standard RCGA is shown in Algorithm 1.

Algorithm 1 Real Coded Genetic Algorithm

```
Initialize Population
Evaluate fitness
while $!termination$ do
  while $i < PopulationSize$ do
    1) selection
    2) crossover
    3) mutation
    $i++$
  end while
  Update population
end while
Get the best solution
```

The computational and optimization power of RCGA has been demonstrated in several theoretical studies [6], [17], [18]. The major advantages of real-coded GA over standard binary-coded GA are their precision maintainability, the chromosome size reduction directly improving computation time and real-valued encoding giving a close conceptual approximation of real weight values. The genetic operator choice directly influences convergence. However, different forms of the genetic operators i.e. selection, crossover and mutation are needed according to the GA type and the optimization problem nature. An overview of each implemented genetic operator is discussed below:

Selection: It ensures that the fittest chromosome qualities survive over generations. Common selection strategies are rank [20], roulette wheel, [21] and the elitist one, [19]. In rank, selection is done according to the individual fitness rank where the fittest individuals have priority. In roulette wheel, selection is done according to the wheel which gives priority to those individuals with greater fitness. However, lesser fit chromosomes are also chosen as they may contain useful genetic material. In elitism, some of the fittest individuals are always retained to insure survival of the best performing chromosomes.

Crossover: It exchanges genetic material from selected parents forming either one or multiple offsprings. Common operators are flat crossover, [17], simple crossover, [6], [22], arithmetic crossover, [22], blend crossover, [24], linear breeder crossover, [25], and Wright's heuristic crossover. The latter was implemented since it is leading to superior performance compared with binary GA, [11].

Mutation: It provides for random diversity in the population helping when the GA converges towards a local minima. The effect of mutation rates, the strength of mutation and its impact in building better solutions has been studied in [26]–[28]. Common mutation operators are random uniform mutation and Michalewicz non-uniform mutation [22], [23]. In uniform mutation, a random number in the range of $[a, b]$ is added to a selected gene where a and b are the highest and lowest values in the chromosome, respectively. In non-uniform mutation, the strength of mutation is decreased as the number of generations increases.

B. Pivot Mutation

The Pivot Mutation operator selects a pivot point in the chromosome and all the genes after the selected pivot are mutated by adding a small real-random numbers, respectively. For instance, given a chromosome $x = (x_1, x_2, x_3, ...x_n)$, the resulting pivoted chromosome becomes $y = (x_1, x_2, y_3, ...y_n)$, where ,$y_i = x_i + r$ given that r is a small real random number in the interval $[a, b]$, where a and b are small negative and positive real numbers chosen by the user, respectively.

C. Genetic operators implemented in this study

The Wright's heuristic crossover operator is selected for its superior performance, [11]. From a pair of parents $x^1 = (x_1^1, x_2^1, x_3^1, ...x_n^1)$ and $x^2 = (x_1^2, x_2^2, x_3^2, ...x_n^2)$, where n is the number of unknowns, an offspring is produced as follows:

$$y_i = r(x_i^1 - x_i^2) + x_i^1$$

where $r \in [0, 1]$ is a real random number and x^1 is the parent with the best fitness. A new offspring is produced with a new r value until a chromosome with a a better fitness is created.

The Michalewicz's uniform mutation operator adds a small random number $[a, b]$ to a selected gene, where a and b are the greatest and the least values found in the chromosomes, respectively. The non-uniform mutation operator is one of the widely used operators in real-coded GA, [23]. In this method, from a point $x^t = (x_1^t, x_2^t, x_3^t, ...x_n^t)$, the muted point $x^{t+1} = (x_1^{t+1}, x_2^{t+1}, x_3^{t+1}, ...x_n^{t+1})$ is created as follows:

$$x^{t+1} = \begin{cases} x_i^t + \Delta(t, x_i^u - x_i^t), & \text{if } r \leq 0.5 \\ x_i^t - \Delta(t, x_i^u - x_i^t), & \text{otherwise} \end{cases}$$

where t is the current generation number and r is a uniformly distributed random number in the interval $[0, 1]$. x_i^l and x_i^u are the lower and upper bounds of the selected chromosome, respectively. The function $\Delta(t, y)$ given below takes value in the interval $[0, y]$.

$$\Delta(t, y) = y\left(1 - u^{\left(1-\frac{t}{T}\right)^b}\right)$$

where u is a uniformly distributed random number in the interval $[0, 1]$, T is the maximum number of generations and b is a number which determines the strength of the mutation operator. This mutation operator performs global search during the initial search and local search in the later generations.

IV. RESULTS AND ANALYSIS

A. Modeling the hexapod **6-6**

In this section, we shall examine one example of the **FKP** resolution on a typical **6-6** manipulator configuration which has been carefully selected with 16 real solutions.

The manipulator configuration includes the manipulator base coordinates of the joint center positions $\mathbf{OA}_C$ in the base reference frame O and the mobile platform coordinates of the joint center positions $\mathbf{CB}_O$ in the platform reference frame C, the minimum bar lengths if applicable. The corresponding configuration example is shown in the following :

$OA_1(x)$ $OA_1(y)$ $OA_1(z)$	464.141 389.512 -178.804
$OA_2(x)$ $OA_2(y)$ $OA_2(z)$	569.471 207.131 -178.791
$0A_3(x)$ $OA_3(y)$ $OA_3(z)$	529.050 -597.151 -178.741
$CB_1(x)$ $CB_1(y)$ $CB_1(z)$	68.410 393.588 236.459
$CB_2(x)$ $CB_2(y)$ $CB_2(z)$	375.094 -137.623 236.456
$CB_3(x)$ $CB_3(y)$ $CB_3(z)$	306.664 -256.012 236.461

Table I
PARALLEL MANIPULATOR CONFIGURATION TABLE

The joint variables are set respectively to $L := [1250, 1250, 1250, 1250, 1250, 1250]$. For the aforementioned hexapod parallel manipulator, we have deliberately chosen one case with sixteen exact results in order to increase difficulty.

B. Algebraic system exact solution

In order to verify the GA method, we revert to an exact algebraic method which has computed the certified exact results, [16], which are shown on table II where only the six first variables are displayed with two exact digits after the point due to space limitations.

x_1 y_1 z_1	x_2 y_2 z_2
-125.52 -426.75 561.80	-432.55 104.25 561.38
68.70 393.33 1006.99	375.31 -137.92 1006.87
68.52 -616.35 449.06	375.52 -363.77 916.17
-68.19 917.28 821.48	-374.94 430.43 609.06
-68.75 -393.74 -994.30	-375.42 137.47 -994.15
-68.00 -393.74, 637.18	-375.00 137.28 637.03
-126.015 -426.71 -919.06	-432.92 104.35 -918.47
68.08 -616.27 -806.53	374.54 -363.34 -1273.80
-126.88 505.33 916.53	-499.39 366.71 449.34

Table II
FKP ROOT EXACT RESULTS

Table II gives only the exact solutions found with the three operators.

C. Results obtained using real-coded genetic algorithm

The performance of 3 different combinations of genetic operators in RCGA is evaluated. In all experiments, the roulette wheel selection is used in conjunction with the elitist strategy method. This ensures that the fittest chromosome is retained in future generations. Note that if the population size is P, then P selections are done in order to make P offsprings for the new population. The following combinations were used in order to get the best combination of genetic operators in order to evaluate the performance of the proposed 'Pivot Mutation' operator. Note that the respective mutation and crossover rates were determined in trial experimental runs.

- 1. Hx Non-uniform: The Wright's heuristic crossover combined with Michaelwicz's non-uniform mutation. The crossover and mutation rate are 0.8 and 0.05, respectively.
- 2. Hx Crossover: In this strategy, the Wright's heuristic crossover operator dominates the GA. There is no mutation. The crossover rate is 0.8.
- 3. Hx Pivot: Wright's heuristic crossover operator is combined with Pivot Mutation. The crossover and mutation rates are 0.8 and 0.1, respectively.

For the selected **6-6** hexapod **FKP** , the implemented objective function is given in equation 8. All experiments initialize the population with real numbers in the range of $[-700, 700]$. The population size of 40 and chromosome size of 9, representing all variables were used. Fig. 3 shows the convergence of a typical experimental run for each GA optimization technique. For each method, a total of 50 experimental runs were done. The results are summarized in table III. The optimization results for coordinates x_1, y_1, z_1, x_2, y_2 and z_2 with their respective fitness are given in table IV.

	No. Generations	Fitness	CPU Time
Hx Crossover	3000± 0	247206 ± 46864	226.06±4.29
Hx Non-Uniform	3000± 0	5878 ±1638	251.38±0.74
Hx Pivot	2806±147	20.47±9.03	252.66±13.21

Table III
THE MEAN AND 95 PERCENT CONFIDENCE INTERVAL FOR 50 EXPERIMENTAL RUNS

$x_1\ y_1\ z_1$	$x_2\ y_2\ z_2$	Fitness
Hx Crossover		
-65.82 -722 -474.34	-568.01 -613.72 -735.03	155724
Hx Non-Uniform		
37.45 393.95 1027.2	411.20 -101.64 1005.62	1313
77.25 -625.32 451.71	276.67 -310.62 944.35	1392
-44.90 934.99 826.51	-392.94 510.87 556.63	941
-146.79 -447.49 529.58	-448.58 91.90 539.57	231.5
Hx Pivot		
-125.15 -437.51 552.16	-424.36 97.05 572.55	0.999628
65.5314 400.848 1007.03	365.17 -132.92 1007.46	3.471
62.43 -615.52 448.77	354.80 -352.34 918.99	0.9999
-68.36 918.16 821.64	-378.36 434.43 604.99	0.9999
-66.30 -407.11 -982.29	-366.356 126.21 -1003.56	3.0124
-70.65 -383.52 646.93	-382.85 145.46 629.73	2.0849
-124.57 -435.38 -910.28	-427.11 97.61 -925.52	0.9914
86.35 -620.16 -809.44	435.91 -398.82 - 1264.04	12.20
-133.64 312.21 916.68	-515.18 328.42 433.56	20.53

Table IV
OPTIMAL SOLUTION VALUES FROM THE BEST EXPERIMENTAL RUN FOR EACH METHOD.

Note that the experiments were done on a 2 Giga Hertz Linux dual core machine and the CPU time is given in milliseconds. The CPU time was calculated using the number of clock ticks taken for the GA in converging to a solution. The CPU executes 2^9 clock ticks in 1 second (2 Giga-Hz CPU). Therefore, the time in milliseconds was calculated by $T(ms) = (ClockTicks * (1/2^9)) * 1000$.

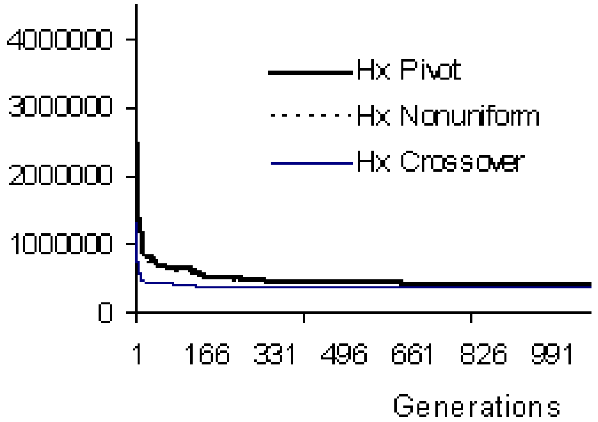

Figure 3. A typical experimental run showing the convergence of the three methods

D. Discussion on model solving

According to table III results, Pivot mutation has outperformed other methods with the smallest fitness values and also the smallest deviations. For each method, the best solutions are given in table IV. The smallest fitness values are obtained with the *Hx Pivot operator*. Moreover, this method could easily converge toward more exact real solutions. Note that in the method where no mutation operator is present (Hx Crossover), poor convergence is indicated by the large fitness final value.

This method has approached nine certified solutions. It obtained these results with a certain level of error. The overall accuracy of the optimization method can vary significantly and is still an issue which will be addressed in further investigation. The fitness function value gives an account of the solution precision and the *Hx Pivot* algorithm surely outperforms the other operators. Hence, the Pivot mutation algorithm has shown its potential to find all solution for the problem and therefore calls for further research. Paper space limitations have forced us to leave thorough accuracy analysis and initial guess impact for further work.

V. CONCLUSION

To solve the difficult **FKP** , an innovative Genetic Algorithm based method has been examined for the general spatial **6-6** parallel manipulator. For the first time, the inverse kinematics equation system using the position-based model served as a basis for the objective function expression to write the error sum. Then, the objective function needed to be augmented with some mobile platform dimension constraints. The proposed *Pivot Mutation* operator used in conjunction with Wright's heuristic crossover was successful in converging to nine solutions confirmed by the exact solutions given by a proven algebraic method.

In future work, we shall investigate the reduction of the GA optimization time and improving its accuracy by using different genetic operators. This work will aim to prepare implementation of a **FKP** solver first in a trajectory simulator and later in a real-time robot controller applied in material handling. Further works on accuracy and initial guess are suggested as well.

ACKNOWLEDGMENT

The authors would like to thank Information Technology for their valuable support by providing access to computers for the required calculations.

REFERENCES

[1] F. Ronga and T. Vust. Stewart platforms without computer? In Proc. Int. Conf. on Real, Analytic and Algebraic Geometry, Trento, pp. 197-212, 1992.

[2] D. Lazard. On the representation of rigid-body motions and its application to generalized platform manipulators. Computational Kinematics, vol. 1, pp 175-181, 1993.

[3] B. Mourrain. The 40 generic positions of a parallel robot. In Proc. ISSAC'93, Kiev, pp. 173-182, 1993.

[4] 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, edited by H. Wu, I-Tech Education and Publishing, 2008.

[5] J. H. Holland. Adaptation in natural and artificial systems. The University of Michigan Press, 1975.

[6] D. E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Systems, vol. 5, pp. 139-167, 1991.

[7] R. Boudreau and N. Turkkan. Solving the Forward Kinematics of Parallel Manipulators with a Genetic Algorithmn. Journal of Robotics Systems, vol. 13, no. 2, pp. 111-125, 1995.

[8] R. Boudreau, S. Darenfeld S. and N. Turkkan. Etude comparative de trois nouvelles approches pour la solution du problème géométrique direct des manipulateurs parallèles. Mechanism and Machine Theory, vol. 33, no. 5, pp. 463-477, 1998.

[9] L. Sheng et al. Forward Kinematics of the Stewart platform using hybrid immune genetic algorithm. In Proc. of the 2006 IEEE International Conference on Mechatronics and Automation. Luoyang, June 25-28th 2006.

[10] A. Omran, G. El-Bayiumi, M. Bayoumi, and A. Kassem. Genetic Algorithm Based Optimal Control for a 6-DOF Non Redundant Stewart Manipulator. International Journal of Mechanical Systems Science and Engineering, vol. 2, no. 2, pp. 73-79, 2008.

[11] A.H. Wright. Genetic algorithms for real parameter optimization. Foundations of Genetic Algorithms. Morgan Kaufman ed., 1991.

[12] L. Rolland. Synthesis on the forward kinematics problem algebraic modeling for the spatial parallel manipulator, displacement-based equation systems. Advanced Robotics, Vol. 21, No. 9, pp. 1071-1092, 2007.

[13] M. Raghavan and B. Roth. Solving polynomial systems for the kinematic analysis and synthesis of mechanisms and robot manipulators. Transac. of ASME, vol. 117, pp. 71-79, 1995.

[14] R. Parrish, E. Dieudonné and R. Bardusch. An actuator extension transformation for a motion simulator and an inverse transformation applying newton-raphson's method. Technical report D-7067, NASA, Washington DC, 1972.

[15] J. P. Merlet. Les Robots Parallèles. Hermès, Paris, 1997.

[16] L. Rolland. Certified solving of the forward kinematics problem with an exact method for the general parallel manipulator. Advanced Robotics, vol. 19, no. 9, pp. 995-1025, 2005.

[17] N. J. Radcliffe. Equivalence class analysis of genetic algorithms. Complex Syst. vol. 2, no. 5, pp. 183-205, 1991.

[18] J. Schaffer, R. Caruana, L. Eshelman and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J. Schaffer ed. 3rd International Conference on Genetic Algorithms, pp. 51-60, Morgan Kaufmann, San Mateo, 1989.

[19] K. A. DeJong. An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation. University of Michigan, 1975.

[20] D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. Proceedings of the Third International Conference on Genetic Algorithms, pp. 116-121, 1989.

[21] J. E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey, Lawrence Erlbaum Associates, pp. 14-21, 1987.

[22] Z. Michalewicz. Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York, 1992.

[23] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation, vol. 4, no. 1, pp. 1-32, 1996.

[24] L. J. Eshelman and J. D. Scahffer. Real-coded genetic algorithms and interval-schemata. Foundations of Genetic Algorithms, vol. 2, pp. 187-202, Morgan Kaufmann ed., San Mateo, 1993.

[25] D. Schlierkamp-Voosen. Strategy adaptation by competition. Proceedings of the Second European Congress on Intelligent Techniques and Soft Computing, pp. 1270-1274, 1994.

[26] J.E. Smith. Modelling GAs with self adapting mutation rates. Proceedings of the Genetic and Evolutionary Computation Conference, 2001.

[27] J.E. Smith and T.C. Fogarty. Operators and parameter adaptation in genetic algorithms. Soft Computing, vol. 1, no. 2, pp. 81-87, 1997.

[28] N. Krasnogor and J.E. Smith. Emergence of profitable search strategies based on a simple inheritance mechanism. Proceedings of the International Conference on Genetic and Evolutionary Computations, pp. 432-439, Morgan Kaufmann, San Mateo, CA, 2001.

[29] K. Deep and M. Thakur. A new mutation operator for real coded genetic algorithms. Applied Mathematics and Computation, vol. 193, no. 1, pp. 211-230, 2007.