Abstract—Applicability of tuning the controller gains for Stewart manipulator using genetic algorithm as an efficient search technique is investigated. Kinematics and dynamics models were introduced in detail for simulation purpose. A PD task space control scheme was used. For demonstrating technique feasibility, a Stewart manipulator numerical-model was built. A genetic algorithm was then employed to search for optimal controller gains. The controller was tested onsite a generic circular mission. The simulation results show that the technique is highly convergent with superior performance operating for different payloads.

Keywords—Stewart Kinematics, Stewart Dynamics, Task Space Control, Genetic Algorithm.

I. INTRODUCTION

Flight simulators imitate the physical feeling of piloting an aircraft by providing graphical windows, sound, and motion platform. One of the most popular flight simulator platforms is Stewart manipulator, where a moving plate is connected to a base plate by six legs. Each leg has an upper part sliding inside a lower part simulating the three translational motions (surge, sway, and heave) and the three rotational motions (pitch, roll, and yaw) as shown in Fig. 1.

Two schemes are commonly used in control of the Stewart manipulator: task space control and joint space control. The task space control scheme has been investigated by [1-2]. In this scheme, the frame work is multi-inputs multi-outputs (MIMO). Thus the forward kinematics model is imbedded in the control loop to estimate the task space displacements \( \mathbf{X} \) from the measured joint displacements \( \mathbf{q} \) as shown in Fig. 2.

On the other hand, the joint space scheme is developed by the information of joint displacements only, since each leg of the manipulator is controlled as a single-input single-output (SISO) system. The error between the actual and desired joint displacement is used as a feedback signal to the controller. The inverse kinematics of the Stewart manipulator has a closed form and it is easy to be implemented. In this way, the sophisticated computations of the forward kinematics are omitted from the control loop. This scheme has been widely used by many research reports, especially for experimental application. Pasquale [10] used a robust control scheme with acceleration feedback. Li [11] designed a proportional gain controller. Fang [12] implemented a fuzzy control. Su [13]
proposed a new technique of robust auto disturbance rejection controller (ADRC). However all these studies overlook the manipulator dynamics, because the controller is designed based on actuator model such as hydraulic system [11, 13], or servo motors [10, 12], which restricts the ability of designing a controller with high performance tracking.

The contribution of this paper is to show the merit of using genetic algorithm in tuning the controller gains for Stewart manipulator based on manipulator dynamics instead of actuator dynamics as in the previous studies [10-13]. For simplicity, a PD controller in joint space is considered. An actuator dynamics as in the previous studies [10, 12], which restricts the ability of designing servo motors [10, 12], which restricts the ability of designing a controller with high performance tracking.

The position of the joint \( J_i \) in the inertia frame is defined as:

\[
P^R_{J_i} = \begin{bmatrix} x_{ui}^B \\ y_{ui}^B \\ z_{ui}^B \end{bmatrix} \quad i = 1,2,...,6
\]

\[
= \begin{bmatrix} R_i \cos(\beta_i) & R_i \sin(\beta_i) & 0 \end{bmatrix}^T
\]

In the same manner, an angle \( \alpha_i \) is defined between the inertia \( X \)-axis and the line of the joint \( J_i \). The position of the joint \( J_i \) in the inertia frame is defined as:

\[
P^R_{J_i} = \begin{bmatrix} x_{ui}^B \\ y_{ui}^B \\ z_{ui}^B \end{bmatrix} \quad i = 1,2,...,6
\]

\[
= \begin{bmatrix} R_i \cos(\alpha_i) & R_i \sin(\alpha_i) & 0 \end{bmatrix}^T
\]

The upper plate has a capability for 6-DOF motion (three rotational motions and three translational motions). The rotational motions of the plate are defined by Euler angles in sequence 1-2-3. Thus the transformation from the body frame \((x_b, y_b, z_b)\) to the inertia frame \((X, Y, Z)\) is given by the Matrix \( R_{plate} \):

\[
R_{plate} = \begin{bmatrix} C\psi C\theta & S\phi S\psi C\theta - C\phi S\theta S\psi + S\phi S\psi & C\phi S\psi S\theta + C\phi S\psi & \end{bmatrix}
\]

\[
C\phi S\psi C\theta S\phi S\psi \theta + C\phi S\psi & C\phi S\psi S\theta - C\phi S\psi & S\phi C\theta
\]

In addition to the rotation, one should consider the translation vector \( T^l_{plate} \) as:

\[
T^l_{plate} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) + h \end{bmatrix}
\]

where \( h \) is the initial height of the upper plate’s center. The trajectory of the upper plate’s center is defined by \( x(t), y(t), \) and \( z(t) \). The position of the joint \( J_{ui} \) in inertial frame \((X, Y, Z)\) is then calculated as:

\[
P^R_{J_{ui}} = \begin{bmatrix} x_{ui}^B \\ y_{ui}^B \\ z_{ui}^B \end{bmatrix} = R_{plate}P^B_{J_{ui}} + T^l_{plate}
\]

The length vector of the \( i^{th} \) leg \( L^i_i \) can then be computed from (2) and (6) as:

\[
L^i_i = P^R_{J_{ui}} - P^R_{J_{ui}} \quad i = 1,2,...,6
\]

By substituting from (3) and (5) into (6), and considering the square value of vector \( L^i_i \) in (7), the relationship between the joint space variables and task space variables can be summarized as:
\[ \Gamma_i^2 = X_i^2 + Y_i^2 + Z_i^2 \]

where

\[
\begin{align*}
X_i &= C_o C_p R_o \cos(\beta_i) + (S_o S_p C_p - C_o S_p) R_e \sin(\beta_i) \\
Y_i &= C_o S_p R_o \cos(\beta_i) + (S_o S_p S_p + C_o C_p) R_e \sin(\beta_i) \\
Z_i &= -S_o R_e \cos(\beta_i) + S_p C_o R_e \sin(\beta_i) + z(t) + h
\end{align*}
\]  

and \(i = 1, 2, ..., 6\). Based on (8), the inverse kinematics has a closed form. On the other hand, it is “impossible” to develop any closed form for the forward kinematics.

Each leg has three degrees of freedom: two rotational and one translational motion. The leg can rotate around the universal joint, while the upper part of the leg is sliding inside the lower part by an actuating force \(F\) as shown in Fig. 4. Thus a spherical joint is employed to connect the upper part of each leg by the movable plate while the lower part is connected to the base plate by a universal joint.

The motion of the each leg is considered by two frames: a leg fixed frame \((X_{\text{leg}}, Y_{\text{leg}}, Z_{\text{leg}})\) located at the joint \(J_i\) parallel to the inertia frame and the leg body frame \((x_{ib}, y_{ib}, z_{ib})\) located at the same point with \(x_{ib}\)-axis pointing in upward. The rotation sequence of the leg starts from rotating around \(Z_{\text{leg}}\)-axis with an angle \(\Gamma\), followed by a rotation about the \(y_{ib}\)-axis with an angle \(\epsilon\). The rotational angle of the leg can be specified by the position of the upper joint \(P_{Ja}^I\) and the position of the lower joint \(P_{Jb}^I\) as

\[
\Gamma_i = \tan^{-1}\left( \frac{Y_{ib}^I - Y_{il}^I}{X_{il}^I - X_{ib}^I} \right)
\]

\[
e_i = \tan^{-1}\left( \frac{Z_{il}^I - Z_{iL}^I}{\sqrt{(X_{il}^I - X_{iL}^I)^2 + (Y_{il}^I - Y_{iL}^I)^2}} \right)
\]

where \(i = 1, 2, ..., 6\). The transformation matrix \(R_{\text{Leg}}\) from \((x_{ib}, y_{ib}, z_{ib})\) frame to \((X_{\text{leg}}, Y_{\text{leg}}, Z_{\text{leg}})\) frame is given as

\[
R_{\text{Leg},i} = \begin{bmatrix}
C_{ei} & C_{ri} & -S_{ei} & C_{ri} \\
S_{ri} & C_{ei} & C_{ri} & S_{ei} \\
0 & 0 & 1 & 0
\end{bmatrix}
\]

The angular velocity \(\omega_i\) of the \(i^{th}\) leg with respect to the leg body frame is defined by

\[
\omega_i = \begin{bmatrix}
\hat{\Gamma} \sin(\epsilon) \\
\hat{\epsilon} \\
\hat{\Gamma} \cos(\epsilon)
\end{bmatrix}
\]

### III. DYNAMICS MODEL

The dynamics model of Stewart manipulator has been addressed by many methods such as the Lagrange equation [1, 11], Newton-Euler equation [13-14], and the principle of virtual work [16-17]. Based on the results that have been shown by Khalil [18], Newton-Euler method emerged as the most effective way to model Stewart manipulator dynamics. However this method has been highlighted by some common errors in previous research reports. These errors were listed by Shaowen [19], and corrected in the current research.

In Newton–Euler method, the dynamics model of Stewart manipulator is described through 24 governing equations, six equations for the movable plate, and the others for the legs. For completeness, the dynamic equations will be list here. More details of the derivation are given by many references [13, 14, and 18]. Thus the moment equation around the universal joint is

\[
m_{ij} R_{ij} \times (-\ddot{g} + \ddot{a}_i) + m_{u_j} R_{u_j} \times (-\ddot{g} + \ddot{a}_{u_j}) + \left[ I_{u_i} + I_{L_i} \right] \ddot{L}_i + \ddot{\omega}_j \times \left[ I_{u_i} + I_{L_i} \right] \ddot{L}_i - \dot{L}_i \times \dot{L}_i = 0
\]  

where \(m_{ij}\) is the lower leg mass, \(m_{u_j}\) is the upper leg mass, \([ I_{L_i} \] is the “invariant” inertia matrix of the lower leg , \([ I_{u_i} \] is the “variant” inertia matrix [17] of the upper sliding leg, \(\ddot{a}_i\) is the acceleration of the lower leg, \(\ddot{a}_{u_j}\) is the acceleration of the upper leg, \(\ddot{g}\) is the gravitational acceleration, \(\dot{\omega}_j\) is the angular velocity of the leg , \(\ddot{a}_{u_j}\) is the angular acceleration, \(\ddot{L}_i\) is the length of the whole prismatic
leg, \( \mathbf{r}_u \) is the position vector of the lower leg from universal joint, \( \mathbf{r}_u' \) is the position vector of the upper leg measured from the universal joint, and \( \mathbf{f}_i \) is the reaction force between the spherical joint and the upper plate. The reaction force \( \mathbf{f}_i \) is decomposed into three components in the leg’s body frame as shown in Fig. 4. The force equation in \( x_{hi} \)-direction of the sliding mechanism is given as

\[
m_{pi} \mathbf{r}_{u'} \left( \ddot{\mathbf{r}} + \dddot{\mathbf{r}}_{u'} \right) = \mathbf{F}_i - f_x \tag{13}
\]

The dynamics of the upper plate have three moment equations and three force equations as

\[
m_p \mathbf{a}_p = m_p \dot{\mathbf{g}} + \sum_{i=1}^{6} \mathbf{R}_{plate}^T \mathbf{R}_{leg}^T \mathbf{r}_i \mathbf{f}_i
\]

\[
\left[ l_p \right] \dddot{\mathbf{r}} + \mathbf{a}_p + \sum_{i=1}^{6} \mathbf{R}_{plate}^T \mathbf{R}_{leg}^T \mathbf{r}_i \mathbf{f}_i
\]

where \( i = 1, 2, \ldots, 6 \), \( m_p \) is the mass of plate plus the external payload, \( \left[ I_p \right] \) is the inertia matrix of the upper plate, \( \mathbf{a}_p \) is the acceleration for the upper plate’s center of mass, \( \mathbf{a}_p \) and \( \dot{\mathbf{a}}_p \) are the angular velocity and acceleration of plate, and \( \mathbf{r}_b \) is the position vector from the center of plate to the joint \( J_h \).

Solving the dynamic equations of Stewart manipulator has two models. The first model is the inverse dynamics computing the equilibrium forces. The second model is the forward dynamics building the simulation tool. The input of the inverse dynamics is the desired trajectory of the movable platform as a function of the time and the outputs are the actuator forces. On the other hand, the inputs of the forward dynamics models are the actuator forces applied at cylindrical joints, and the outputs are the movable upper plate positions and orientations. The algorithm of the inverse dynamics model can be summarized in the following steps:

- **Step 1:** Specify the desired task space displacements as \([\phi(t), \theta(t), \psi(t), x(t), y(t), z(t)]^T\) and their derivatives with time.
- **Step 2:** Obtain the transformation matrix \( \mathbf{R}_{plate} \) of the moving plate from (3).
- **Step 4:** Obtain the angular velocity of the moving plate \( \omega_p \) from (4).
- **Step 5:** Use numerical differentiation to compute the angular acceleration of the moving plate \( \mathbf{a}_p \).
- **Step 6:** Compute the positions of joints \( J_{ai} \) and \( J_{bi} \) from (2) and (6).
- **Step 7:** Obtain the value of angles \( \epsilon \) and \( \Gamma \) for each leg from (9).
- **Step 8:** Obtain the transformation matrix for each leg \( \mathbf{R}_{leg} \) from (10).
- **Step 9:** Use numerical differentiation to evaluate the time derivatives of angles \( \epsilon \) and \( \Gamma \).
- **Step 10:** Obtain the angular velocity \( \mathbf{a}_p \) of each leg from (11).
- **Step 11:** Use numerical differentiation to compute the angular acceleration \( \mathbf{a}_p \) for each leg.
- **Step 12:** Solve (12) to compute \( f_x \) and \( f_y \) for each leg.
- **Step 13:** Solve (14) as a set of linear homogeneous equations in \( f_{ai} \).
- **Step 14:** Calculate the actuating force \( \mathbf{F} \) from (13).

Sequentially, the forward dynamics model is developed in reverse direction of the inverse kinematics algorithm.

### IV. OPTIMIZATION PROCEDURE USING GA

Genetic algorithm is now considered as one of the most popular optimization and search techniques. The first obvious application for the algorithm traced back to 1962 when Holland introduced the algorithm in his work studying adaptive systems [20]. The algorithm then received an enormous exploration by Goldberg [21]. The main advantages of GA are its global optimization performance and the ease of distributing its calculations among several processors or computers as it operates on the population of solutions that can be evaluated concurrently. It is a very simple method, generally applicable, not inclined to local optimization problems that arise in a multimodal search space, and no needs for special mathematical treatment. Moreover, the algorithm is more applicable for the discontinuous problem unlike the conventional gradient-based searching algorithms.

Genetic algorithm basically works based on the mechanism of natural selection and evolutionary genetics. The algorithm starts by coding the variables to binary strings (chromosomes). Every chromosome has \( n \) genes. The gene is a binary bit by value zero or one. Three main operations control the procedure of the GA: reproduction, crossover, and mutation. Reproduction is processing to select the parent from a generation. The process is based on survival of the fittest (highest performance index). In this way, the reproduction process guides the search for the best individuals (high performance index). After the individuals are selected, the crossover process is then used to swap between two chromosomes by specific probabilistic decision. The crossover process generates offspring carrying mixed information from swapped parents (chromosomes). Mutation is the mechanism to prevent the algorithm from local optimal points by adding some degree of randomness. The process is performed by alternation of the gene from zero to one or from one to zero with the mutation point determined uniformly at random. The mutation rate should be consider carefully since the higher mutation rate means more number of generations are required for algorithm convergence and a low mutation rate may lead to a convergence for a local minimum. The algorithm maintains a constant size of generation by selecting the fittest chromosomes from parents and offsprings. The algorithm iteratively operates to converge for schema matches by some tolerance. Roughly, a genetic algorithm works as shown in Fig. 5. Further description of genetic algorithms can be found in Goldberg [21-22].

Fig. 6 shows a joint space PD controller scheme. In this scheme, the inverse kinematics is employed to compute the
desired joint displacements \((L_1, L_2, \ldots, L_6)\) from the desired task space displacements \((x_d, y_d, z_d, \theta_d, \phi_d, \psi_d)\), the desired and measured joint displacements are then compared feeding the control logic.

\[
F = K_p e + K_d \dot{e} = K_p (L_{ref} - L) + K_d (\dot{L}_{ref} - \dot{L})
\]  

(15)

The control law of this PD is given as

The PD controller is commonly designed by analytical methods such as root locus, or state space model [23-25]. If the nonlinearity is significant, then it is difficult to use such analytical methods and weigh up the influence of each gain on the response. In this case, other methods were proposed. Haruhisa [23] has developed a gain tuning technique based on time derivatives. Faa-Jeng [24] used a recurrent fuzzy-neural-network (RFNN) to tune an IP controller. The application of optimal tuning technique to the controller gains has been extensively explored for processes that are difficult to be tuned analytically. Baogang [25] proposed a new methodology for a nonlinear PID control. This controller is based on the theoretical fuzzy analysis and genetic-based optimization. The controllers gave better results than the conventional PID. The controller has not been applied to practical problems; the model is a nonlinear mathematical one. Chris [26] has applied an optimization technique to control a robot in tracking problem as a highly nonlinear problem. The results showed the power of using optimization techniques in tuning controller parameters. This encourages using optimization techniques in this research to tune controller gains of Stewart manipulator.

The performance index given in (16) is selected here to minimize the absolute area under the error curve (difference between desired and measured joint displacements) with time. In addition, another term proportional to the settling time is added avoiding the flatness of the error curve. A 2% criteria is used to define the settling time. Settling time is also considered as the time span for the integration The weighting factors \(w_1\) and \(w_2\) are selected such that the two terms in the cost function being in the same level of the magnitude.

\[
F_{cost} = w_1 \times \sum_{i=1}^{6} \int_{t=0}^{t=span} |e_i(t)| dt + w_2 \times T_{span}
\]

(16)

GA in Fig. 5 propagates searching for optimal controller gains in (15) to minimize the cost function in (16) for unit step inputs as a reference.

V. SIMULATION RESULTS AND DISCUSSION

The proposed optimization technique is applied to the Stewart platform with parameters given in Table I.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
<th>Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>(L_u)</td>
<td>Length of upper leg</td>
<td>0.95</td>
<td>m</td>
</tr>
<tr>
<td>(L_l)</td>
<td>Length of lower leg</td>
<td>0.95</td>
<td>m</td>
</tr>
<tr>
<td>(R_u)</td>
<td>Radius of upper plate</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>(R_l)</td>
<td>Radius of base plate</td>
<td>1</td>
<td>m</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>Joint angles of base plate</td>
<td>[-50, 50, 70, 170, -170, -70] deg</td>
<td></td>
</tr>
<tr>
<td>(\beta)</td>
<td>Joint angles of upper plate</td>
<td>[-2, 2, 118, 122, -122, -18] deg</td>
<td></td>
</tr>
<tr>
<td>(m_{lu})</td>
<td>Mass of each upper leg</td>
<td>37.17</td>
<td>kg</td>
</tr>
<tr>
<td>(m_{ll})</td>
<td>Mass of each lower leg</td>
<td>37.17</td>
<td>kg</td>
</tr>
<tr>
<td>(m_u)</td>
<td>Mass of each upper plate</td>
<td>194.71</td>
<td>kg</td>
</tr>
<tr>
<td>(K_{pl})</td>
<td>Lower value of (K_p)</td>
<td>(10^4)</td>
<td>N/m</td>
</tr>
<tr>
<td>(K_{pu})</td>
<td>Upper value of (K_p)</td>
<td>(10^6)</td>
<td>N/m</td>
</tr>
<tr>
<td>(K_{dl})</td>
<td>Lower value of (K_d)</td>
<td>(10^3)</td>
<td>N/m</td>
</tr>
<tr>
<td>(K_{du})</td>
<td>Upper value of (K_d)</td>
<td>(10^5)</td>
<td>N/m</td>
</tr>
<tr>
<td>(I_{u1} I_{u2} I_{u3})</td>
<td>Moment of inertia of the lower leg</td>
<td>0.0632, 2.8536, 2.8536</td>
<td>kg</td>
</tr>
<tr>
<td>(I_{u4} I_{u5} I_{u6})</td>
<td>Moment of inertia of the upper leg</td>
<td>0.009, 1.5921, 1.5921</td>
<td>kg</td>
</tr>
</tbody>
</table>

For GA optimization, the mutation rate is 10%. Each generation has a fixed population size 100 or no generation overlap. The algorithm is highly convergent. The number of
generations for convergence is 30. The optimization algorithm converges at the values of $K_p$ and $K_d$ as \{9.623 8.055 7.939 6.487 7.755 3.323\} x10^5 N/m and \{1.745 2.461 5.006 3.452 3.264 1.203\} x10^4 N.sec/m respectively. The performance of the controller is tested again by a generic mission. This mission is a horizontal circular track with radius 0.01 m. The parametric equation of this mission is defined as

\[
\zeta = \frac{2\pi}{T} \left( t - \frac{T}{2\pi} \sin \left( \frac{2\pi t}{T} \right) \right)
\]

\[
x_{tr} = 0.01 \sin \zeta
\]

\[
y_{tr} = 0.01 (1 - \cos \zeta)
\]

\[
z = 1
\]

\[
0 \leq \zeta \leq 2\pi \quad \text{and} \quad 0 \leq t \leq T
\]

where the dummy variable $\zeta$ is implemented to pledge that all functions in (17) have zero velocities and accelerations at the beginning and end of the mission. Also the mission has been assigned to be inside the geometric workspace given in Fig. 7. The inverse kinematics was employed to compute the reference joint space displacements ($L_d$) shown in Fig. 8. Now the control model is tested onsite this mission, when ($L_d$) is considered as inputs (see Fig. 6). Fig. 9 shows the generated actuator forces based on the control law given in (15). The time records of the actuator forces look very smooth. This implies that the controller has the capability to capture the relation between the applied forces and the measured joint displacements. Fig. 10 mentions the error between the desired and measured task space displacements. In Fig. 10, the order of maximum error is $10^{-5}$ m, while the order of the desired task displacement is $10^{-2}$ m, which is quite adequate for the flight simulator applications. In addition two different payloads are added to the upper platform. Fig. 12 demonstrates the capability of the controller to perform at different operating conditions with acceptable accuracy levels.
VI. CONCLUSIONS

This paper presents the modeling and control algorithm of a non-redundant 6-DOF Stewart manipulator. It shows that the PD control scheme, using active joints’ degrees of freedom feedback and optimized with GA, is computationally efficient and easy to implement as far as the accuracy of the inverse kinematics model is guaranteed. The control scheme is tested on a three-dimensional circular mission. The results show the efficiency of the algorithm and the robustness of the resulting controller with variable load.

REFERENCES