Use of PSO for Obtaining Solution of the Inverse Robot Dynamic Model

Carlos Alberto Yahir Hervert Cano, Angel Rodríguez-Liñán, Luis M. Torres-Treviño

Universidad Autónoma de Nuevo León, UANL, FIME, San Nicolás de los Garza, Nuevo León, Mexico
hervertkno@gmail.com, angel.rodriguezln@uanl.edu.mx, luis.torres.ciidit@gmail.com

Abstract. The inverse dynamic model allows to compute the torques for each of the degrees of freedom of a robot knowing the positions, velocities and accelerations. In contrast, direct dynamic model is used to calculate the positions, velocities and accelerations knowing the torques for each of the degrees of freedom. Most of the times is difficult to obtain the direct dynamic model analytically because it is given by solution of the differential equations for each variable. For that reason, we propose to compute the positions, velocities and accelerations from the inverse dynamic model using a simple and powerful intelligent algorithm called Particle Swarm Optimization (PSO) without needing to obtain the direct dynamic model.

Keywords: PSO, inverse robot dynamic model.

1 Introduction

Modelling the robot dynamics allows to understand the relationship between the movement of the robot links and its forces. This relationship is obtained by dynamic models which relate mathematically:

1. The location of the robot defined by their joint variables: Position, velocity and acceleration.
2. The forces and torques applied to the joints.
3. Robot dimensional parameters, such as length, mass and inertia of their elements.

A method to obtain the robot dynamic model of n degrees of freedom with rigid links, is given by the equations of motion Euler-Lagrange [6, 1]. The Lagrangian Ł is defined as the difference of the kinetic energy K and potential energy U:

$$Ł = K - U$$

Then, the inverse dynamic model is expressed by the n equations of Euler-Lagrange:

$$\frac{d}{dt}\frac{\partial \mathrm{L}}{\partial \dot{q}_i} - \frac{\partial \mathrm{L}}{\partial q_i} = \tau_i \tag{1}$$

where q_i is the joint position, $\dot{q}_i$ is the joint velocity, and τ_i is the force (or torque) of the i-th link and $i = 1, 2, ..., n$. The inverse dynamic model (1) can be rewritten in their compact form and notation most widely used in robotics [2, 9]:

$$\tau = D(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) + F_c sign(\dot{q}) + F_v \dot{q} \tag{2}$$

where $\tau \in \Re^n$, $q \in \Re^n$, $\dot{q} \in \Re^n$, $\ddot{q} \in \Re^n$ denote the vectors of applied torques, position, velocity, and acceleration in the robot joints, respectively. $D(q) \in \Re^{n\times n}$ is the inertia matrix, $C(q, \dot{q}) \in \Re^{n\times n}$ is the Coriolis and centripetal matrix, $G(q) \in \Re^n$ is the gravity vector, $F_c \in \Re^{n\times n}$ is the Coulomb friction and $F_v \in \Re^{n\times n}$ is the viscous friction.

The goal of this article is to get positions, velocities and accelerations satisfying the inverse dynamic model (2) without solving the n differential equations (1) because most of the time is very difficult solve them. We propose in this work the use of Particle Swarm Optimization (PSO). PSO and Evolutionary Algorithms are optimization tools that are inspired by natural phenomena.The PSO, in particular, was motivated by the simulation of bird flocking or fish schooling. PSO was first introduced by Kennedy and Eberhart in 1995 [7].

In recent years, the relationship between the dynamic models and Intelligent Systems as PSO has been an important merger, since the Intelligent Systems have started to get several applications in robotics as:

⇒ In parameter estimation of robot dynamics because most of the times is very difficult to know the exact parameters or even impossible to obtain all the parameters [5].
⇒ For obtaining the positions, velocities and accelerations of the inverse dynamic model [4].
⇒ In the control of swarm robots [3].

The rest of document is organized as follows: In the next section, the PSO is presented. In Section 3, the inverse dynamic model of three study cases are shown. Experimental results from simulations with PSO are illustrated in Section 4. Finally, some conclusions are given.

2 Particle Swarm Optimization

The PSO algorithm [5, 8, 11] assigns a swarm of k particles to search for the optimal solution in a m-dimensional space, where m is the amount of position components in each and every one of particles. The optimal solution to the problem consists in that any particle find a point or trajectory in the m-dimensional space that minimize some fitness function $f(x)$. The starting position of a particle is randomly set within the range of possible solutions to the problem. The range is based on an intuitive guess of the maximum and minimum possible values of each component of the particle positions p. For the i-th particle is valued the fitness function $p_{b_i} = f(p_i)$ of its current position $p_i \in \Re^m$ that determined the current best $p_{b_{i,t}}$ in the t-th iteration, with $t = \{1, 2, ..., id\}$ as the number of iterations, and has memory of its own best experience $P_{best,i} \in \Re^m$, which is compared to $p_{b_{i,t}}$ in t-th iteration, and is replaced by $p_{b_{i,t}}$ if $f(p_i(t)) < f(P_{best,i}(t-1))$. Besides its own best experience, each particle has knowledge of the best experience achieved by the entire swarm, that is the global best experience denoted by $G_{best} \in \Re^m$ such that $f(G_{best}) = \min_i(f(P_{best,i}))$. Based on the data each agent has, its velocity in the t-th iteration is determined by

$$v_i(t) = wv_i(t-1) + c_1r_1(P_{best,i}(t) - p_i(t)) + c_2r_2(G_{best}(t) - p_i(t)) \tag{3}$$

where $v_i \in \Re^m$ is the velocity of i-th particle, $w \in \Re$ is the inertia weight, $c_1 \in \Re$ is a constant positive cognitive learning rate, $c_2 \in \Re$ is a constant positive social learning rate, $r_1 \in [0, 1]$ and $r_2 \in [0, 1]$ are random numbers re-generated at each iteration.

The position of each particle at the $(t+1)$-th iteration is updated by:

$$p_i(t+1) = p_i(t) + v_i(t) \tag{4}$$

After predefined conditions are satisfy, the algorithm stops and the G_{best} at the latest iteration is taken as the optimal solution to the problem.

In this work, the PSO is used to obtain an estimated value of position q, velocity $\dot{q}$ and acceleration $\ddot{q}$ of n joints of a robot for each iteration, where we know the torque τ and inverse dynamic model (2). Then, the estimated torque is given by

$$\hat{\tau} = D(\hat{q})\ddot{\hat{q}} + C(\hat{q}, \dot{\hat{q}})\dot{\hat{q}} + G(\hat{q}) + F_c sign(\dot{\hat{q}}) + F_v\dot{\hat{q}} \tag{5}$$

where $\hat{q} \in \Re^n$, $\dot{\hat{q}} \in \Re^n$ and $\ddot{\hat{q}} \in \Re^n$ are the estimated position, velocity and acceleration by PSO, respectively. Defining the estimation error $e_i(t) \in \Re^n$ for the i-th particle in the t-th iteration

$$e_i(t) = \|\tau - \hat{\tau}_i(t)\|_1 \tag{6}$$

where $\| \cdot \|_1$ is the norm-1. Let $f : \Re^n \to \Re$ be the fitness function to be optimized, which compute the average error [10]

$$f(e_i) = \frac{(e_{i,1} + e_{i,2} + ... + e_{i,n})}{n} \tag{7}$$

for n freedom degrees.

The objective of the PSO algorithm in the estimation task is to find the i-th set of variables q, $\dot{q}$, $\ddot{q}$ that minimizes the function $f(e_i)$.

3 Inverse Robot Dynamic Model

In this section, 3 manipulator models are shown in order of estimate their position, velocity and acceleration by means of the PSO algorithm presented. In the 3 cases, friction effects was not taken into account in the inverse dynamic models.

Case 1: Simple pendulum.

The inverse dynamic model of the simple pendulum without friction shown in Figure 1 is:

$$\tau = ml^2\ddot{q} + mgl\sin(q) \tag{8}$$

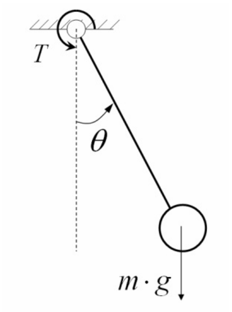

Fig. 1. Simple pendulum without friction

The simple pendulum of Figure 1 has 1 degree of freedom ($n = 1$), $q = \theta$ is the angular position, $\dot{q} = \dot{\theta}$ is the angular velocity, $\ddot{q} = \ddot{\theta}$ is the angular acceleration, the mass value was taken as $m = 1$ Kg and the length as $l = 1$ m, and the inverse model (8) is in form (2).

Case 2: Pendubot.

The inverse dynamic model of the ideal Pendubot shown in Figure 2 is:

$$\begin{aligned}
\tau_1 = {} & [m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos(q_2) + I_1 + I_2]\ddot{q}_1 \\
& + [m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos(q_2) + I_2]\ddot{q}_2 - 2m_2 l_1 l_{c2}\sin(q_2)\dot{q}_1\dot{q}_2 \\
& - m_2 l_1 l_{c2}\sin(q_2)\dot{q}_2^2 + [m_1 l_{c1} + m_2 l_1]g\sin(q_1) + m_2 g l_{c2}\sin(q_1 + q_2) \\
\tau_2 = {} & [m_2 l_{c2}^2 + 2m_2 l_1 l_{c2}\cos(q_2) + I_2]\ddot{q}_2 + [m_2 l_{c2}^2 + I_2]\ddot{q}_2 \\
& + m_2 l_1 l_{c2}\sin(q_2)\dot{q}_1^2 + m_2 g l_{c2}\sin(q_1 + q_2)
\end{aligned} \tag{9}$$

The ideal pendubot of Figure 2 has 2 degrees of freedom ($n = 2$), q_i is the angular position, $\dot{q}_i$ is the angular velocity, $\ddot{q}_i$ is the angular acceleration for each

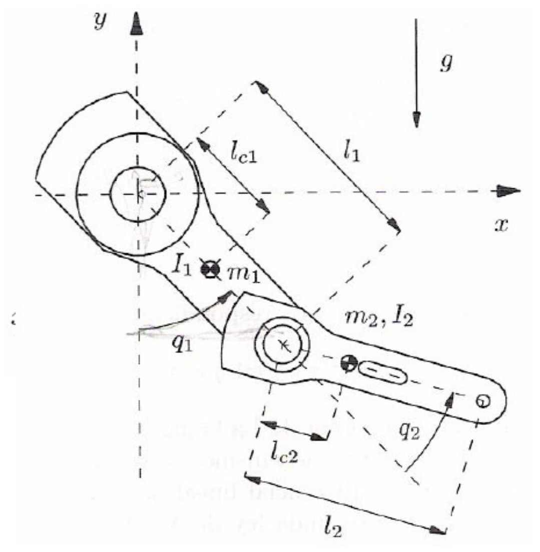

Fig. 2. Pendubot

degree of freedom respectively, the parameter values were taken as $m_1 = 5\text{Kg}$, $m_2 = 3\text{Kg}$, $l_1 = 1\text{m}$, $l_2 = 1\text{m}$, $I_1 = 0.004\text{Kg/m}^2$, $I_2 = 0.006\text{Kg/m}^2$, $l_{c1} = 0.5\text{m}$ y $l_{c2} = 0.5\text{m}$, and the equations of inverse model (9) are the components of Compact Form (2).

Case 3: 3D Cartesian Manipulator.

The inverse dynamic model of a 3D cartesian manipulator shown in Figure 3 is:

$$\begin{aligned} \tau_1 &= [m_1 + m_2 + m_3]\ddot{q}_1 + [m_1 + m_2 + m_3]g \\ \tau_2 &= [m_1 + m_2]\ddot{q}_2 \\ \tau_3 &= m_1\ddot{q}_3 \end{aligned} \tag{10}$$

The 3D cartesian manipulator of Figure 3 has 3 degrees of freedom ($n = 3$), q_i is the articular position, $\dot{q}_i$ is the articular velocity, $\ddot{q}_i$ is the articular acceleration for each degree of freedom respectively, the parameter values were taken as $m_1 = 1.5\text{Kg}$, $m_2 = 1\text{Kg}$, $m_3 = 3\text{Kg}$, and the equations of inverse model (10) are the components of Compact Form (2).

4 Simulation Results

In this section, simulation results from using the PSO algorithm to estimate the torque or force of the 3 cases presented in the section above are illustrated.

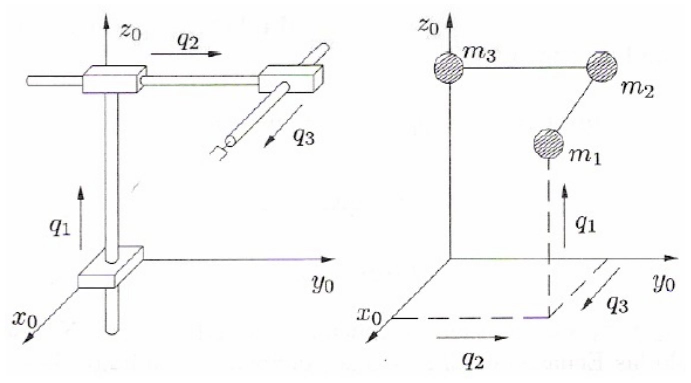

Fig. 3. 3D Cartesian manipulator

The computational experiments with the PSO algorithm for the 3 cases was done in Matlab©. The behaviour parameters of the programed PSO are the inertia weight $w = 0.5$, the cognitive learning rate $c_1 = 0.35$ and the social learning rate $c_2 = 0.35$.

Case 1: Simple pendulum.

The PSO algorithm was run with a population of 200 particles and 100 cycles. In Table 1 the torques τ corresponding to position q and acceleration $\ddot{q}$ values of inverse dynamic model (8) are shown. In Table 2 the torque $\hat{\tau}$, position $\hat{q}$ and acceleration $\ddot{\hat{q}}$ values computed from PSO algorithm minimizing the fitness (7) are shown.

Table 1. Position, acceleration and torque values from model (8)

q	$\ddot{q}$	τ
30	.15	15.165
60	.15	25.9371
90	.15	29.88
120	.15	25.9371
150	.15	15.165
180	.15	0.45
210	.15	-14.265
240	.15	-25.0371
270	.15	-28.98
300	.15	-25.0371
330	.15	-14.265
360	.15	0.45

Table 2. Position, acceleration and torque estimated values from PSO

$\hat{q}$	$\ddot{\hat{q}}$	$\hat{\tau}$
24.4735	1	15.192
128.7704	1	25.9454
66.0566	1	29.8975
64.953	-0.2441	25.9302
149	0	15.1576
185	1	0.435
209	0	-14.2679
252.2604	1	-25.0306
256	0	-28.5558
237.0218	-0.1149	-25.0377
209	0	-14.2679
185	1	0.435

In Figure 4, the torque values τ and their estimated $\hat{\tau}$ are shown for angular positions q from $0°$ to $360°$. The blue line represents the actual torque τ for the different positions of the pendulum with the known position q, velocity $\dot{q}$ and acceleration $\ddot{q}$. The red line is the torque estimated $\hat{\tau}$ with the estimated position $\hat{q}$, velocity $\dot{\hat{q}}$ and acceleration $\ddot{\hat{q}}$. It can be seen that 90 and 270 degrees are values such that more torque is required because the pendulum is in horizontal position.

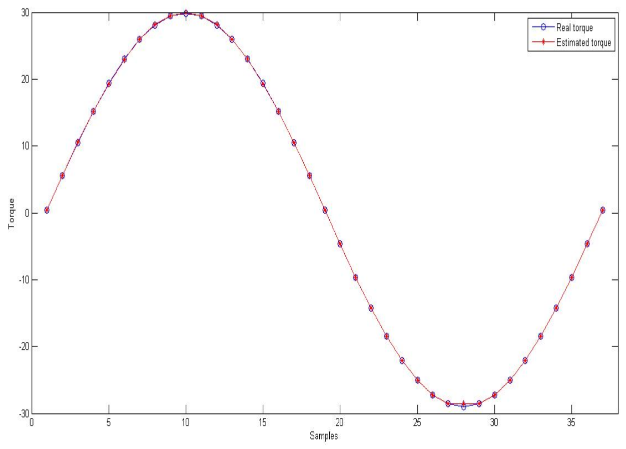

Fig. 4. Torque comparison of pendulum

Case 2: Pendubot.

The PSO algorithm was run with a population of 600 particles and 100 cycles. In Table 3 the torques τ_1, τ_2 corresponding to position q_1, q_2, velocity $\dot{q}_1, \dot{q}_2$ and acceleration $\ddot{q}_1, \ddot{q}_2$ values of inverse dynamic model (9) are shown. In Table 4 the torque $\hat{\tau}_1, \hat{\tau}_2$, position $\hat{q}_1, \hat{q}_2$ and acceleration $\ddot{\hat{q}}_1, \ddot{\hat{q}}_2$ values computed from PSO algorithm minimizing the fitness (7) are shown.

In Figure 5, the torque values τ_1, τ_2 and their estimated $\hat{\tau}_1, \hat{\tau}_2$ are shown for angular positions q_1, q_2 from $0°$ to $180°$. The blue line represents the actual torque τ_i for the different positions of the pendubot with the known position q_1, q_2, velocity $\dot{q}_1, \dot{q}_2$ and acceleration $\ddot{q}_1, \ddot{q}_2$. The red line is the torque estimated $\hat{\tau}_1, \hat{\tau}_2$ with the estimated position $\hat{q}_1, \hat{q}_2$, velocity $\dot{\hat{q}}_1, \dot{\hat{q}}_2$ and acceleration $\ddot{\hat{q}}_1, \ddot{\hat{q}}_2$. In the Figure 5 can be seen that desired torque and the torque estimated are almost identical. The joint 1 requires more torque because it is the joint where the weight of both links rests.

Table 3. Position, velocity, acceleration and torque values from model (9)

q_1	q_2	$\dot{q}_1$	$\dot{q}_2$	$\ddot{q}_1$	$\ddot{q}_2$	τ_1	τ_2
0	0	0.2	0.15	0	0	0	0
0	180	0.2	0.15	0	0	0	0
20	0	0.2	0.15	0	0	5.0328	5.0328
40	0	0.2	0.15	0	0	9.4586	9.4586
60	20	0.2	0.15	0	0	32.9028	14.512
80	40	0.2	0.15	0	0	47.3456	12.7821
100	60	0.2	0.15	0	0	51.6521	5.0848
120	80	0.2	0.15	0	0	47.9806	-4.9737
140	100	0.2	0.15	0	0	40.2699	-12.6845
160	120	0.2	0.15	0	0	32.1278	-14.4395
160	140	0.2	0.15	0	0	21.8585	-12.705
180	120	0.2	0.15	0	0	33.8757	-12.6916
180	140	0.2	0.15	0	0	25.1434	-9.4201
180	160	0.2	0.15	0	0	13.3785	-5.0123
180	180	0.2	0.15	0	0	0	0

Case 3: 3D Cartesian Manipulator

The PSO algorithm was run with a population of 800 particles and 100 cycles. In Table 5 the forces τ_1, τ_2, τ_3 corresponding to acceleration $\ddot{q}_1, \ddot{q}_2, \ddot{q}_3$ values of inverse dynamic model (10) are shown. In Table 6 the force $\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3$ and acceleration $\ddot{\hat{q}}_1, \ddot{\hat{q}}_2, \ddot{\hat{q}}_3$ values computed from PSO algorithm minimizing the fitness (7) are shown.

In Figure 6, the force values τ_1, τ_2, τ_3 and their estimated $\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3$ are shown for different articular acceleration values $\ddot{q}_1, \ddot{q}_2, \ddot{q}_3$. The blue line represents the actual force τ_i for the different known accelerations $\ddot{q}_1, \ddot{q}_2, \ddot{q}_3$ of the cartesian

Table 4. Position, velocity, acceleration and torque estimated values from PSO

$\hat{q}_1$	$\hat{q}_2$	$\dot{\hat{q}}_1$	$\dot{\hat{q}}_2$	$\ddot{\hat{q}}_1$	$\ddot{\hat{q}}_2$	$\hat{\tau}_1$	$\hat{\tau}_2$
0	0	1	0	0	0	-0.03	-0.01
0	0	1	0	0	0	-0.03	-0.01
16	0	1	0	-0.1	1	5.	4.59
320	180	0	0	0	0	9.45	9.45
310	160	0	0	0	0	32.28	13.82
76.95	46.3	1	1	-0.3	1	47.39	13.5
89	74	1	1	0	0	51.84	5.74
236	91	1	1	1	1	47.12	-5.02
194	82	1	1	1	0	39.76	-12.18
147	125	0	0	1	0	32.78	-14.81
238	33	0	0	1	0	22.19	-12.69
159	95.18	1	1	-0.3	1	34.04	-12.17
182	41.72	1	0	0	0	24.74	-9.59
55	154	1	1	0	1	13.95	-5.72
360	360	1	0	0	0	-0.03	-0.01

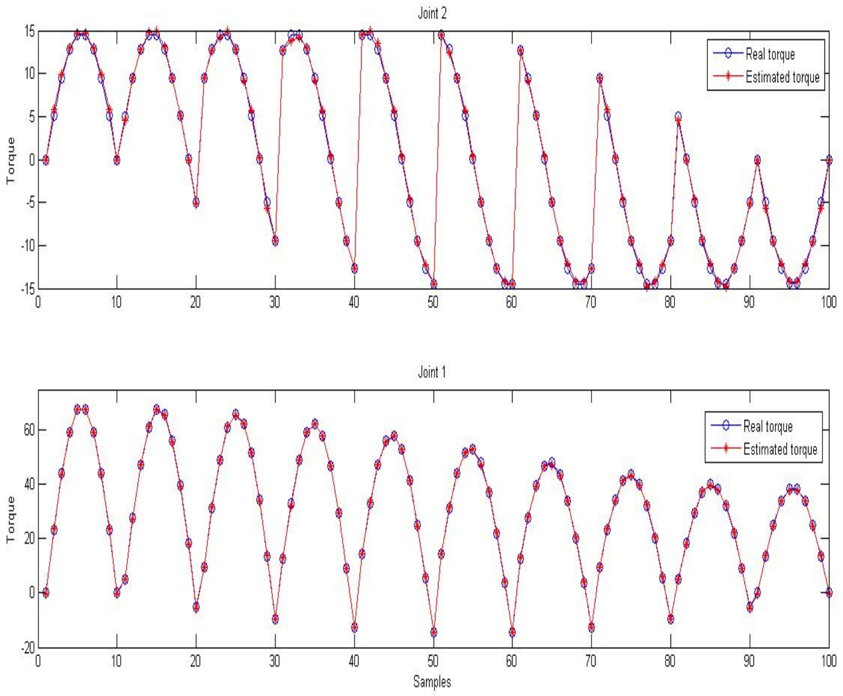

Fig. 5. Torque comparison of Pendubot

manipulator. The red line is the force estimated $\hat{\tau}_1, \hat{\tau}_2, \hat{\tau}_3$ with the estimated accelerations $\hat{\ddot{q}}_1, \hat{\ddot{q}}_2, \hat{\ddot{q}}_3$. It can be seen that desired force and the force estimated are almost identical. The joint 1 requires more force because it is the joint where supports the weight of entire robotic manipulator. The force in joint i-th only depends on the acceleration i-th to perform the motion.

Table 5. Acceleration and force values from model (10)

$\ddot{q}_1$	$\ddot{q}_2$	$\ddot{q}_3$	τ_1	τ_2	τ_3
0	10	20	0.5396	25	30
0	20	0	0.5396	50	0
0	20	10	0.5396	50	15
10	0	10	60.5395	0	15
10	0	20	60.5395	0	30
10	10	0	60.5395	25	0
20	0	20	120.5396	0	30
20	10	0	120.5396	25	0
20	10	10	120.5396	25	15
20	10	20	120.5396	25	30

Table 6. Acceleration and force estimated values from PSO

$\hat{\ddot{q}}_1$	$\hat{\ddot{q}}_2$	$\hat{\ddot{q}}_3$	$\hat{\tau}_1$	$\hat{\tau}_2$	$\hat{\tau}_3$
0	11	23	0.5396	27.5	33.5
0	20	1	0.5396	50	1.5
0	19	7	0.5396	47.5	13.5
10.3864	0	9.8226	62.8579	0	14.7338
10	0	21	60.5395	0	31.5
10	10	0	60.5395	25	0
20	0	18	120.5396	0	28
20	10	2	120.5396	25	2
20	10	11	120.5396	25	16
20	10	20	120.5396	25	30

5 Conclusions

Due to Forward Dynamic Model is difficult to obtain; a time-efficient, easily implemented, and flexible method based on Particle Swarm Optimization was applied to estimation of robot dynamics. As was shown through of simulations for a simple pendulum, pendubot and 3D cartesian manipulator, the estimation of dynamic variables (position, velocity, acceleration and torque) is very effective

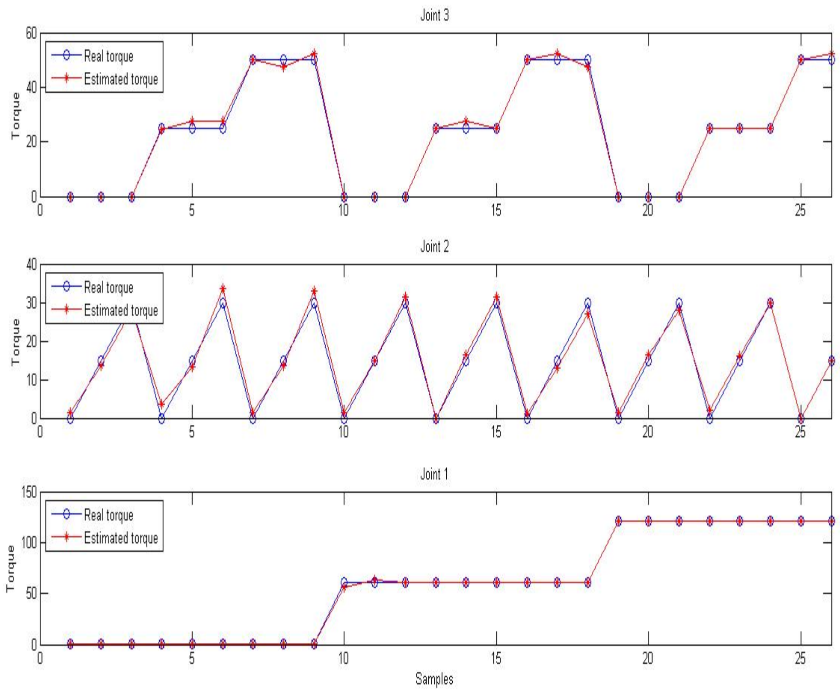

Fig. 6. Force comparison of 3D Cartesian manipulator

and precise. This method is easily executable for any user without no great robot dynamics knowledge for any manipulator with an arbitrary number of degrees of freedom. As future works are considered the parameter estimation (mass, length, Coulomb friction, viscous friction) when dynamic variables are known and position control of manipulator using PSO.

Acknowledgments Author Carlos Hervert thanks to Mexican CONACyT (Consejo Nacional de Ciencia y Tecnología) for scholarship grants. The work of Angel Rodríguez-Liñán was supported by PAICYT (Programa de Apoyo a la Investigación Científica y Tecnológica) de la UANL under grant IT562-10.

References

1. Barrientos, A., nin, F.P., Balaguer, C., Aracil, R.: Fundamentos de Robótica. McGraw Hill
2. Cortéz, F.R.: Robótica Control de Robots Manipuladores. Alfaomega, México, D.F. (2012)

3. Estévez-Carreón, J., García-Ramírez, R.: Necessary conditions for the application of the shooting method to an optimal control of position in a manipulating robot (14), 72–80 (2006)
4. Grigoriadis, G., Mertzios, B., Tourassis, V.: Fast implementation of robot inverse dynamics with distributed arithmetic via a simd architecture. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS 29(2), 209–216 (March 1999)
5. Jahandideh, H., Namvar, M.: Use of pso in parameter estimation of robot dynamics part one: No need for parameterization (2012)
6. Kelly, R., nes, V.S.: Control de movimiento de robots manipuladores. Automática Robótica, Prentice Hall, México, D.F.
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceedings of the IEEE International Conference on Neural Networks pp. 1942–1948 (1995)
8. Lu, L., Luo, Q., yong Liu, J., Long, C.: An improved particle swarm optimization algorithm. IEEE (2008)
9. Swevers, J., W. Verdonck, J.S.: Dynamic model identification for industrial robots. IEEE CONTROL SYSTEMS MAGAZINE (October 2007)
10. Weidong-Ji, Keqi-Wang: An improved particle swarm optimization algorithm. IEEE International Conference on Computer Science and Network Technology pp. 585–589 (2011)
11. Zhu, H., Pu, C., Eguchi, K., Gu, J.: Euclidean particle swarm optimization. 2nd International Conference on Intelligent Networks and Intelligent Systems pp. 669–672 (2009)