World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:7, No:7, 2013

Motion Control of a Ball Throwing Robot with a Flexible Robotic Arm

Yizhi Gai, Yukinori Kobayashi, Yohei Hoshino, and Takanori Emaru

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

***Abstract*—**Motion control of flexible arms is more difficult than that of rigid arms, however utilizing its dynamics enables improved performance such as a fast motion in short operation time. This paper investigates a ball throwing robot with one rigid link and one flexible link. This robot throws a ball at a set speed with a proper control torque. A mathematical model of this ball throwing robot is derived through Hamilton's principle. Several patterns of torque input are designed and tested through the proposed simulation models. The parameters of each torque input pattern is optimized and determined by chaos embedded vector evaluated particle swarm optimization (CEVEPSO). Then, the residual vibration of the manipulator after throwing is suppressed with input shaping technique. Finally, a real experiment is set up for the model checking.

***Keywords*—**Motion control, flexible robotic arm, CEVEPSO, ball throwing robot.

I. Introduction

RAPID motion robot is an important research topic in the robotics research field. Robots equipped with excellent rapid motion functions can simulate fast action more humanlike or even exceed the athletic ability of human beings. Such motion functions would also be helpful for disabled person to recover lost movement abilities by the artificial limbs including the rapid movement functions. There are some researches on this aspect, such as a round plate throwing robot using a rigid link [1]. But for a rigid manipulator, there are disadvantages including heavy weight, large input or control energy, and other problems. To reduce the weight and utilize energy more efficiently, a flexible link manipulator would appear to be a better choice than rigid one. There are also studies on flexible robotic manipulators, like golf swing robots [2]. For this kind of robotic manipulator, it is necessary to overcome a number of difficulties. For example, in comparison with rigid counterpart, it is hard for the flexible link manipulator to maintain sufficient trajectory precision as well as difficult to control for its characteristics.

This paper reports a two-link flexible manipulator able to perform the action of throwing a ball. Here, to obtain an accurate torque input, a Chaos Embedded Vector Evaluated Particle Swarm Optimization (CEVEPSO) algorithm is introduced to optimize the controlling parameters of the input.

First in Section II, the system model is illustrated and then the equations of motion of this system are derived. In Section III, the equations of motion are transformed into a state equation. After that, the CEVEPSO algorithm and results with its application are detailed in Section IV. In Section V, the input shaping technique is introduced to control the residual vibration after throwing followed by the real experiment which is used for model accuracy checking in Section VI. Finally, in the last section, conclusions are described.

Y. Gai, Y. Kobayashi, Y. Hoshino, and T. Emaru are with the Graduate School of Engineering, Hokkaido University, Sapporo, Hokkaido, Japan (e-mail: gaiyizhi@mech-hm.eng.hokudai.ac.jp, kobay@eng.hokudai.ac.jp, hoshinoy@eng.hokudai.ac.jp, emaru@eng.hokudai.ac.jp).

II. Model Descriptions and Equations of Motion

Fig. 1 shows the model of the flexible manipulator discussed in the paper. The model consists of a rigid link fixed on a hub with one set of rotational coordinates O_1-x_1y_1 and a flexible beam fixed on another hub with another set of rotational coordinates O_2-x_2y_2. The ball has a ball holder with a ball on extreme end of the flexible part of the beam. The O-XY coordinates denote the inertial reference frame which is in global coordinates. The angle $\theta_1$and $\theta_2$are the rotational angles of the rigid link and flexible link under the torques$\tau_1$and τ_2, respectively. The flexible beam is a homogeneous isotropic beam with uniform cross section. The displacement by vibration at an arbitrary point x of the flexible beam is represented by $v(x,t)$. And the other system parameters are listed in Table I.

Fig. 2 shows the process of the motion, for the ball throwing motion, besides model 1 as in Fig. 1 that in Fig. 2 (a), and one more model shown as Fig. 2 (b) is needed to be considered.

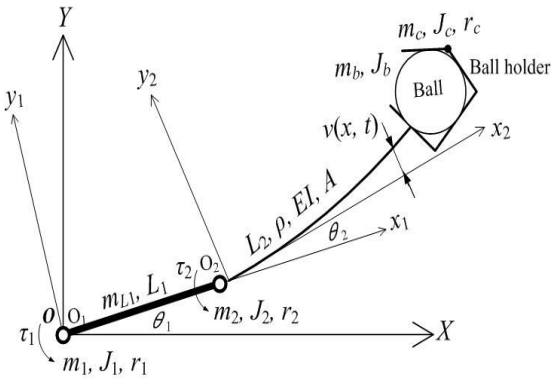


Fig. 1 Model of a ball throwing robot

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

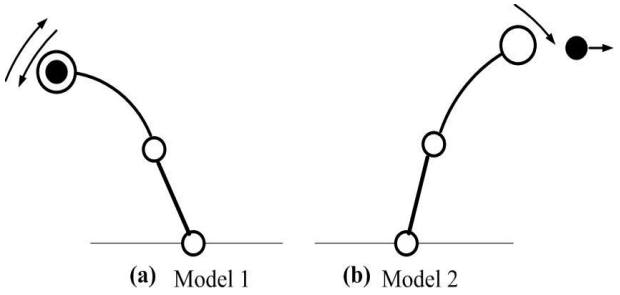


Fig. 2 Throwing motion of manipulator

TABLE I
MODEL PARAMETERS

Symbol	Meaning	Quantity
m_1	Mass of first hub	1.45 kg
J_1	Inertia of first hub	3.45×10^{3} kg·m^2
r_1	Radius of first hub	0.47×10^{-1} m
m_{L1}	Mass of rigid link	0.19 kg
L_1	Length of rigid link	0.26 m
m_2	Mass of second hub	1.04 kg
J_2	Inertia of second hub	1.30×10^{-1} kg·m^2
r_2	Radius of second hub	4.75×10^{-2} m
L_2	Length of flexible link	0.37 m
ρ	Density of flexible link	7.70×10^{3} kg/m^3
E	Young's modulus	4.03×10 GPa
I	Inertia of cross section of beam	5.25×10^{-11} m^4
A	Cross sectional area of beam	8.64×10^{-5} m^2
m_c	Mass of the ball holder	0.10 kg
J_c	Inertia of the ball holder	6.00×10^{-5} kg·m^2
r_c	Radius of the ball holder	0.03 m
m_b	Ball mass	5.50×10^{-2} kg
J_b	Ball inertia	3.75×10^{-5} kg·m^2

For the equation of motion of the system, the kinetic energy of the whole system R can be expressed as

$$R = R_{hub1} + R_{rigid} + R_{hub2} + R_{flexible} + R_{holder} + R_{ball} \tag{1}$$

The kinetic energies for each part can be obtained from:

$$R_{hub1} = \frac{1}{2} J_1 \dot{\theta}_1^2, \tag{2}$$

$$R_{rigid} = \frac{1}{2} m_{L1} \dot{\mathbf{r}}_{1g}^T \dot{\mathbf{r}}_{1g}, \tag{3}$$

$$R_{rigid} = \frac{1}{2} m_{L1} \dot{\mathbf{r}}_{1g}^T \dot{\mathbf{r}}_{1g} + \frac{1}{2} J_2 \left(\dot{\theta}_1 + \dot{\theta}_2\right)^2, \tag{4}$$

$$R_{flexible} = \int_0^{L_2} \frac{1}{2} \rho A \dot{\mathbf{r}}_{3g}^T(x) \dot{\mathbf{r}}_{3g}(x) dx, \tag{5}$$

$$R_{holder} = \frac{1}{2} m_c \dot{\mathbf{r}}_{4g}^T \dot{\mathbf{r}}_{4g} + \frac{1}{2} J_p \left\{\dot{\theta}_1 + \dot{\theta}_2 + \frac{\partial \upsilon(L_2,t)}{\partial x}\right\}^2, \tag{6}$$

$$R_{ball} = \frac{1}{2} m_b \dot{\mathbf{r}}_{4g}^T \dot{\mathbf{r}}_{4g} + \frac{1}{2} J_b \left\{\dot{\theta}_1 + \dot{\theta}_2 + \frac{\partial \upsilon(L_2,t)}{\partial x}\right\}^2, \tag{7}$$

In the above equations, $\mathbf{r}_{1g}$, $\mathbf{r}_{2g}$ and $\mathbf{r}_{4g}$are the position vector of the center of gravity for the rigid link, the second hub and, ball holder, respectively. Similarity, $\mathbf{r}_{4g}$ is the position vector at position x of the flexible link.

The potential energy of the flexible link is

$$V = \frac{1}{2}\int_0^{L_2} EI \left\{\frac{\partial^2 \upsilon(x,t)}{\partial x^2}\right\}^2 dx, \tag{8}$$

The external work W is given as

$$W = \tau_1 \theta_1 + \tau_2 \theta_2, \tag{9}$$

According to Hamilton's principle,

$$\int_{t_1}^{t_2} (R - V + W)\, dt \equiv 0, \tag{10}$$

The governing equation of the system then can be derived as follows:

Equations of motion for $\delta\theta_1$:

$$\begin{aligned}
&\left\{J_1 + J_2 + J_c + J_b + m_{L_1}(\beta L_1 + r_1)^2 + m_2 L_r^2 + (m_c + m_b)\right.\\
&\left(L_f L_r \cos\theta_2 - L_r \upsilon_e \sin\theta_2 + L_{rf}^2 + r_2^2 + L_r^2 + L_{rf} r_2 + \upsilon_e^2\right)\\
&+\rho A \int_0^{L_2} \left[L_r (x + L_2)\cos\theta_2 - L_r \upsilon \sin\theta_2 + L_r^2 + r_2^2 + x^2\right.\\
&\left.+ r_2 x + \upsilon^2\right] dx \Big\} \ddot{\theta}_1 + \left\{J_2 + J_c + J_b + (m_c + m_b)\right.\\
&\left(\tfrac{1}{2} L_f L_r \cos\theta_2 - \tfrac{1}{2} L_r \upsilon_e \sin\theta_2 + L_{rf}^2 + r_2^2 + L_{rf} r_2 + \upsilon_e^2\right)\\
&+\rho A \int_0^{L_2} \left[L_r (x + L_2)\cos\theta_2 - L_r \upsilon \sin\theta_2 + 2\left(r_2^2 + x^2\right.\right.\\
&\left.\left.+ r_2 x + \upsilon^2\right)\right] dx \Big\} \ddot{\theta}_2 + \tfrac{1}{2}(m_c + m_b)\left(L_r \cos\theta_2 + L_f\right)\ddot{\upsilon}_e\\
&+ \tfrac{1}{2}\rho A \int_0^{L_2} \left(L_r \cos\theta_2 + x + r_2\right)\ddot{\upsilon}\, dx + \left(J_c + J_b\right)\ddot{\upsilon}'(L_2)\\
&-(m_c + m_b)\left[L_f \dot{\theta}_1 \dot{\theta}_2 + \tfrac{1}{2} L_f \dot{\theta}_2^2 + \left(\dot{\theta}_1 + \dot{\theta}_2\right)\dot{\upsilon}_e\right] L_r \sin\theta_2\\
&-(m_c + m_b)\left(\dot{\theta}_1 \dot{\theta}_2 \upsilon_e + \tfrac{1}{2}\dot{\theta}_2^2 \upsilon\right) L_r \cos\theta_2 - 2(m_c + m_b)\\
&\left(\dot{\theta}_1 + \dot{\theta}_2\right)\upsilon_e \dot{\upsilon}_e + \rho A \int_0^{L_2} \left\{\left[(x + r_2)\dot{\theta}_1 \dot{\theta}_2 - \tfrac{1}{2}(x + r_2)\dot{\theta}_2^2\right.\right.\\
&\left.\left.+ \left(\dot{\theta}_1 + \dot{\theta}_2\right)\dot{\upsilon}\right] L_r \sin\theta_2 - \left(\dot{\theta}_1 \dot{\theta}_2 \upsilon + \tfrac{1}{2}\dot{\theta}_2^2 \upsilon\right) L_r \cos\theta_2\right.\\
&\left.+ 2\left(\dot{\theta}_1 + \dot{\theta}_2\right)\upsilon \dot{\upsilon}\right\} dx = \tau_1,
\end{aligned} \tag{11}$$

For $\delta\theta_2$:

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

$$\{J_2 + J_c + J_b(m_c + m_b)(\tfrac{1}{2}L_f L_r \cos\theta_2 - \tfrac{1}{2}L_r \upsilon_e \sin\theta_2 + L_{rf}^2 + r_2^2 + L_{rf} r_2 + \upsilon_e^2) + \tfrac{1}{2}\rho A\int_0^{L_2}[L_r(x + L_2)\cos\theta_2 - L_r\upsilon\sin\theta_2 + 2(r_2^2 + x^2 + r_2 x + \upsilon^2)]dx\}\ddot{\theta}_1 + \{J_2 + J_c + J_b + (m_c + m_b)(L_{rf}^2 + r_2^2 + L_{rf} r_2 + \upsilon_e^2) + \rho A\int_0^{L_2}(r_2^2 + x^2 + r_2 x + \upsilon^2)dx\}\ddot{\theta}_2 + \tfrac{1}{2}(m_c + m_b)L_f\ddot{\upsilon}_e + \tfrac{1}{2}\rho A\int_0^{L_2}(x + r_2)\ddot{\upsilon}\,dx + (J_c + J_b)\ddot{\upsilon}'(L_2) + \tfrac{1}{2}L_r\dot{\theta}_1^2(m_c + m_b)(L_f\sin\theta_2 + \upsilon_e\cos\theta_2) + 2(m_c + m_b)(\dot{\theta}_1 + \dot{\theta}_2)\upsilon_e\dot{\upsilon}_e + \tfrac{1}{2}\rho A\int_0^{L_2} L_r\dot{\theta}_1^2[(x + r_2)\sin\theta_2 + \upsilon\cos\theta_2] + 4(\dot{\theta}_1 + \dot{\theta}_2)\upsilon\dot{\upsilon}\}dx = \tau_2, \quad (12)$$

For $\delta\upsilon$:

$$\tfrac{1}{2}\rho A(L_r\cos\theta_2 + r_2 + x)\ddot{\theta}_2 + \tfrac{1}{2}\rho A(r_2 + x)\ddot{\theta}_2 + \rho A\ddot{\upsilon} + EI\upsilon'''' + \tfrac{1}{2}\rho A L_r\dot{\theta}_1^2\sin\theta_2 - \rho A\upsilon(\dot{\theta}_1 + \dot{\theta}_2)^2 = 0, \quad (13)$$

with the boundary conditions:

$$\tfrac{1}{2}(m_c + m_b)(L_f + L_r\cos\theta_2)\ddot{\theta}_1 + \tfrac{1}{2}(m_c + m_b)L_f\ddot{\theta}_2 + (m_c + m_b)\ddot{\upsilon}_e - (m_c + m_b)(\dot{\theta}_1 + \dot{\theta}_2)^2\upsilon_e + \tfrac{1}{2}(m_c + m_b)L_r\dot{\theta}_1^2\sin\theta_2 - EI\upsilon'''(L_2) = 0, \quad (14)$$

$$(J_c + J_b)(\ddot{\theta}_1 + \ddot{\theta}_2) + (J_c + J_b)\ddot{\upsilon}'(L_2) + EI\upsilon'''(L_2) + r_c EI\upsilon''(L_2) = 0, \quad (15)$$

where

$$L_r = L_1 + r_1 + r_2$$
$$L_{rf} = L_2 + r_c$$
$$L_f = L_2 + r_2 + r_c$$
$$\upsilon_e = \upsilon(L_2) + r_c\upsilon'(L_2).$$

III. State Equation

The displacement of the vibration υ can be expressed by an eigen function $\phi_i(x)$ obtained from the ith order vibration mode of a cantilevered beam and a time function $q_i(t)$,

$$\upsilon(x,t) = \sum_{i=1}^{\infty}\phi_i(x)q_i(t). \quad (16)$$

Substituting (16) into (11)~(15) and applying the orthogonality relationship of the vibration mode, the equations of motion can be expressed,

$$\mathbf{M}\ddot{\mathbf{z}} + \mathbf{C}\dot{\mathbf{z}} + \mathbf{K}\mathbf{z} + \mathbf{H} = \mathbf{D}\mathbf{u} \quad (17)$$

where

$$\mathbf{z} = \{\theta_1, \theta_2, q_1, q_2, \cdots q_n\}^T,$$
$$\mathbf{u}(t) = \boldsymbol{\tau}(t).$$

Here **M**, **C**, **K**, and **D**are the mass, damping, stiffness, and input matrices of the system. In the state vector **z**, θ is the rotational angle of the manipulator, and$q_i$is the time function which is combined with the eigen function $\phi_i(x)$ to express the displacement of the flexible beam $\upsilon(x,t)$.

Equation (17) can be rewritten as a state variable equation,

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{W}, \quad (18)$$

where

$$\mathbf{x} = \{\mathbf{z}^T, \dot{\mathbf{z}}^T\},$$
$$\mathbf{A} = \begin{bmatrix} \mathbf{0}_{n+2\times n+2} & \mathbf{I}_{n+2\times n+2} \\ -\mathbf{M}^{-1}\mathbf{K} & -\mathbf{M}^{-1}\mathbf{C} \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} \mathbf{0}_{n+2\times 2} \\ \mathbf{M}^{-1}\mathbf{D} \end{bmatrix},$$
$$\mathbf{W} = \begin{bmatrix} \mathbf{0}_{n+2\times 2} \\ \mathbf{M}^{-1}\mathbf{H} \end{bmatrix}.$$

For Model 2 (Fig. 2(b)) of the system, m_b and J_b can be assumed to be zero.

IV. Torque Input Optimization Based On the CEVEPSO Technique

A. The PSO Algorithm

Particle Swarm Optimization (PSO) [3] is an evolutionary computation technique developed by Kenney and Eberhart in 1995. In our former research [4], the PSO algorithm was used as an optimal algorithm for input design.

Fig. 3 shows a flowchart for implementing this basic PSO. Each particle keeps track of its coordinates in the problem space and the program searches the problem space with the best solution which has been achieved so far. This value is called "pbest". Another "best" value that is tracked by the global version of the particle swarm optimizer is the overall best value, and its location, obtained so far by a particle in the population. This location is called "gbest".

The basic procedure for the global version of PSO is,

- Step one: Initialize the population of particles with random positions and velocities in the search space of the problem.
- Step two: For each particle, evaluate the desired optimal cost function.
- Step three: Compare each particle's cost evaluation with its pbest. If current value is better than pbest, set the pbest value equal to the current value and the pbest location equal to the current location in the search space.
- Step four: Compare cost evaluation with the population's overall previous best. If the current value is better than gbest, reset gbest to the current particle's array index and value.

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

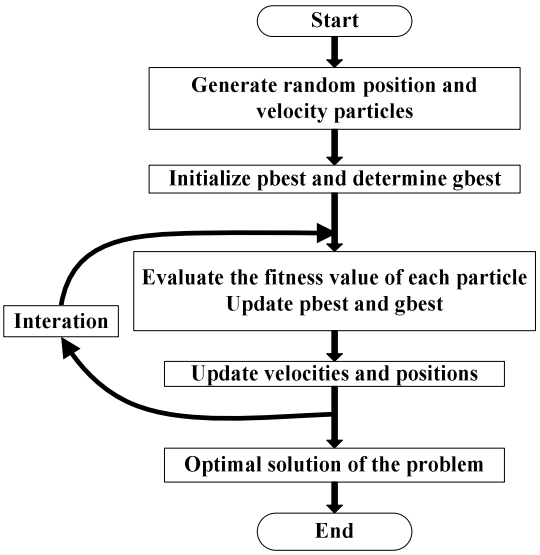


Fig. 3 Flowchart of the basic PSO

- Step five: Update the velocity and position of the particle.
- Step six: Loop to Step two until a criterion is met.

The basic PSO were used for solving some problems such as the optimization of the feedback controller for the flexible manipulator [5] and seismic control optimization of the building [6]. For the problem with one objective function, commonly the basic PSO algorithm can work well. But considering the problem with multiple objective functions, one of the most common methodologies is to aggregate all the objective functions with appropriate weight factors into a single comprehensive objective function. This requires a well knowledge of the problem domain to assign the weight factors appropriately. The nature and appropriate way for this problem tends to handle the multiple objectives separately with multi-objective PSO.

B. Multi-Objective PSO

In Multi-Objective Particle Swarm Optimization (MOPSO) [7], there are many fitness functions. In this way, it is possible to obtain results with specific properties by exploring Pareto's dominance concept. Based on this concept, each particle of the swarm could have different leaders, but only one may be selected to update the velocity. This set of leaders is stored in a repository, which contains the best non-dominated solutions found. At each generation, the velocity of the particles is also updated similar like the basic PSO.

For the common MOPSO, it needs an external archive to store the non-dominated particles. And moreover the selecting rules and size of this archive are hard to be selected and decided. So in this paper, we use VEPSO for the multi-objective optimization and crowding distance for ruling the external archive. If $p \in \text{external archive}$ with respect to cost function cf_i, $i = 1, 2, \cdots, n$. Then for each cost function, the crowding distance of member p is as

$$cd_p(cf_i) = cf_i(q) - cf_i(r), \tag{19}$$

where q is the member of external archive following after $p$in a sorting order which is determined through cf_i values, and r is the member which precedes p in the same order. And the total crowding distance of p can be calculated as

$$cd_{total}(p) = \sum_{i=1}^{n} cd_p(cf_i). \tag{20}$$

C. Vector Evaluated PSO

Vector Evaluated Particle Swarm Optimization (VEPSO) [8]-[10] was introduced by Parsopoulos and Vrahatis as a multi-swarm PSO variant for multi-objective optimization (MO) problems, and it was extended to parallel implementation. The main goal in MO problems is the detection of Pareto optimal points.

In VEPSO, it assumes that n swarms, S_1, S_2, …,S_n of size L, point to optimize n functions simultaneously. And each swarm is evaluated according to one of the objective functions. Swarms exchange information among them by sharing their individual findings to direct search towards the Pareto optimal set.

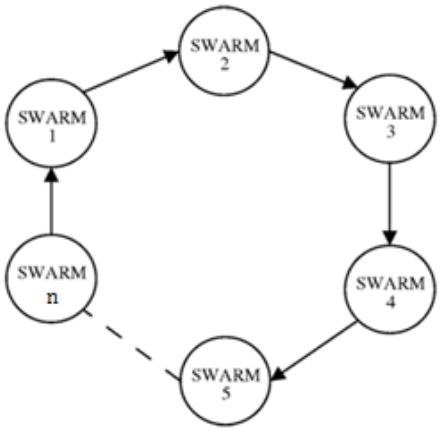


Fig. 4 The ring migration topology for n swarms

The VEPSO permits the parameter configuration of each swarm independently. So, the number of particles as well as the values of the PSO parameters per swarm may differ. A notable characteristic is the insertion of the best position of another swarm in S_n. This information exchange scheme among swarms has a prominent position in VEPSO. It can be clearly viewed as a migration scheme, where particles migrate from one swarm to another according to a connecting topology. Fig. 4 shows the ring migration topology corresponding to the following equation:

$$s = \begin{cases} n, & i = 1, \\ i-1, & i = 2, \cdots, n. \end{cases} \tag{21}$$

Then the swarms of VEPSO are updated according to the following equations:

$$v_{ij}^{k+1} = w \cdot v_{ij}^{k} + c_1 r_1 \left(p_{sj}^{k} - x_{ij}^{k} \right) + c_2 r_2 \left(g_{sj}^{k} - x_{ij}^{k} \right), \tag{22}$$

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k}. \tag{23}$$

where x_{ij}^{k} and v_{ij}^{k} are the position and velocity of the jth dimension of the ith particle in the kth iteration and s represents the ring migration defined in (19); p and g stand for the current individual best position and the global best position respectively;r_1 and r_2 are psedudo-random values in the [0,1] range; c_1 and c_2 are the cognitive and the social parameters respectively; w is inertial weight which determines how much the particle inherits from the former one and is expressed by,

$$w = w_{\max} - k \times \frac{w_{\max} - w_{\min}}{k_{\max}} \tag{24}$$

where k is the current iteration number, $k_{\max}$ is the maximum number of iterations, $w_{\max}$ and $w_{\min}$ are the maximum and minimum values of the inertia weight.

D.Chaos Embedded VEPSO (*CEVEPSO*)

For common PSO algorithm, it can be easily trapped in the local minima position. Moreover, it is difficult to get an acceptable solution without the use of a proper searching strategy. Chaos is a general nonlinear phenomenon in nature, and its behavior is nearly stochastic. For chaos exquisite internal structure, it owns characteristics of randomness, ergodicity, and regularity [11]. The chaos optimization algorithm owns strong ability of jumping out of the local extrema. In addition, PSO with chaos strategy can overcome the shortcoming of common PSO algorithm in which it easily arrives at the local extrema by maintaining the diversity of the swarm.

In this paper, the chaotic system is used to generate sequences for substituting random numbers of VEPSO where it is necessary to make a random based choice. Mathematically, chaos is randomness of a simple deterministic dynamical system and chaotic system may be considered as source of randomness. A chaotic map is a discrete time dynamical system which can be expressed as

$$z_{i+1} = f(z_i) \quad 0 < z_i < 1, \quad i = 0,1,2,\cdots \tag{25}$$

Here the tent mapping strategy is used in the CEVEPSO algorithm which can be expressed as

$$z_{i+1} = \mu\left(1 - 2\left|z_i - 0.5\right|\right), \quad 0 \le z_0 \le 1 \tag{26}$$

where $\mu \in [0,1]$ is the fork control parameter. Here $\mu = 1,$ the tent mapping is in a complete chaos state and moving through the whole range.

Then the velocity updating formulas are also modified correspondingly [12]. The equation is as follows:

$$v_{ij}^{k+1} = CM_1 v_{ij}^{k} + c_1 \cdot CM_{2j}^{k}\left(p_{sj}^{k} - x_{ij}^{k}\right) + c_2 \cdot CM_{3j}^{k}\left(g_{sj}^{k} - x_{ij}^{k}\right) \tag{27}$$

whereCM_1, CM_2, and CM_3 are functions based on the tent mapping with value between 0 and 1.

Comparing with other optimal method, CEVEPSO can search the Pareto Front more efficiently and simply. And it also can avoid some local trap through embedded chaos mapping.

E. Input Torque Design

In this part, the CEVEPSO is applied to establish the input pattern. A sinusoidal form of input is used for each control torque to obtain the optimized inputs here.

$$u_i(t) = \begin{cases} A_{i1}\sin\left(\dfrac{2\pi}{T_{i1}}t\right) & 0 \le t \le \dfrac{T_{i1}}{2} \\ -A_{i2}\sin\left(\dfrac{2\pi}{T_{i2}}t - \dfrac{T_{i1}}{2}\right) & \dfrac{T_{i1}}{2} \le t \le \dfrac{T_{i2}}{2} \\ 0 & t > \dfrac{T_{i2}}{2} \end{cases} \tag{28}$$

where$i$=1,2 and the parameters A_{i1}, A_{i2}, T_{i1}, T_{i2} are optimized by the CEVEPSO algorithm.

The parameters that are fine evaluated here can be viewed as a particle evolution in 8-dimensional search spaces with respect to fitness functions. The search space can be expressed as

$$\mathbf{X} = \begin{bmatrix} A_{11}^1 & T_{11}^1 & A_{12}^1 & T_{12}^1 & A_{21}^1 & T_{21}^1 & A_{22}^1 & T_{22}^1 \\ A_{11}^2 & T_{11}^2 & A_{12}^2 & T_{12}^2 & A_{21}^2 & T_{21}^2 & A_{22}^2 & T_{22}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{11}^n & T_{11}^n & A_{12}^n & T_{12}^n & A_{21}^n & T_{21}^n & A_{22}^n & T_{22}^n \end{bmatrix} \tag{29}$$

where n stands for the number of particles.

For the optimization problem in this paper, the fitness functions are selected as follows

$$J_1 = \frac{1}{\upsilon_{\max}}, \tag{30}$$

$$J_2 = \int_0^{\infty} u(t)\dot{\theta}(t)dt, \tag{31}$$

$$J_3 = t_{oper}. \tag{32}$$

where $\upsilon_{\max}$ is the maximum speed of the ball $u(t)$ is the torque input, $\dot{\theta}(t)$ denotes the angular velocity of the manipulator, t_{oper} is the length of operation time of the control input. Then the problem can be formulated as the following constrained optimization problem to minimize the fitness functions under the conditions,

$$\begin{gathered} A_{\min} \le A \le A_{\max} \\ T_{\min} \le T \le T_{\max} \\ \upsilon \le \upsilon_{\max} \end{gathered} \tag{33}$$

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

where A, T, and, υ are amplitude, time period of the input torque, and deformation of the flexible manipulator respectively.

After applying CEVEPSO algorithm, some solutions are detected through this optimal algorithm. These solutions are shown in Fig. 5. The solutions with the rectangle spots were detected under the condition that the particle number and iteration were 25 and 140, respectively; the solutions with round spots were in the condition that particle number was 15 and the iteration was 160; the solutions with triangle spots were found in the setting of condition of particle 10 and iteration 120. Then by applying the interpolation method we try to estimate the Pareto Fronts of this problem. The solid line is the Pareto Fronts evaluated through all of the solutions. The solutions detected here are not enough for forming the exact Pareto Fronts; however we can estimate the tendency of the solution to some extent.

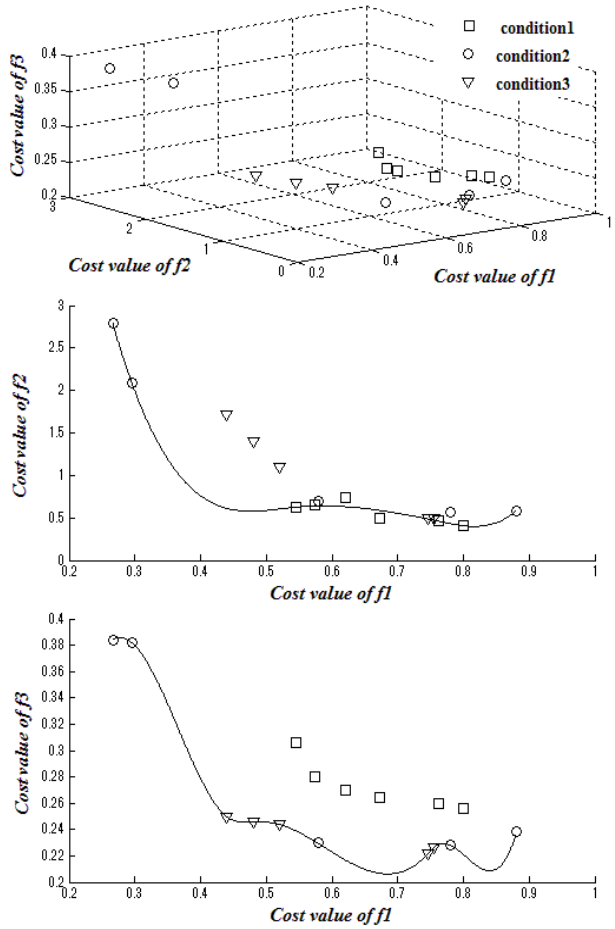


Fig. 5 Pareto fronts valued by CEVEPSO

V. Vibration Suppression Based On Input Shaping Technique

A. Input Shaping Technique

The input shaping technique is a feed-forward control method and widely used for vibration suppression. And this method is composed of a series of impulses [13]. In order to understanding how to generate impulses that move flexible systems without vibration, it is helpful to start with the simple impulses. Applying impulses, A_1 or A_2 to a flexible system will cause it to vibrate. The response of an underdamped system to such impulses is shown in Fig. 6 (a). If the time location and magnitude of each impulse is chosen correctly, then the response of A_2 may be cancelled the response of A_1. This result is shown as the total response in Fig. 6 (b).

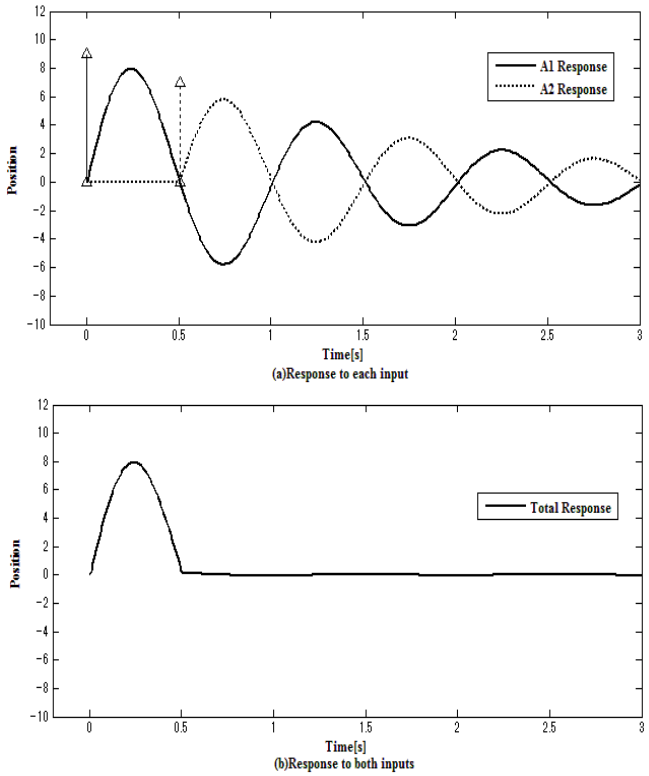


Fig. 6 Vibration caused by two impulses

If an underdamped, second-order system has an undamped natural frequency of ω_n and a damping ratio of ζ, then the residual vibration that results from a sequence of impulses can be described as

$$V(\omega_n,\zeta) = e^{-\zeta\omega_n t_n}\sqrt{\left[C(\omega_n,\zeta)\right]^2+\left[S(\omega_n,\zeta)\right]^2}, \quad (34)$$

where

$$C(\omega_n,\zeta) = \sum_{i=1}^{n} A_i e^{\zeta\omega_n t_i}\cos(\omega_d t_i),$$

$$S(\omega_n,\zeta) = \sum_{i=1}^{n} A_i e^{\zeta\omega_n t_i}\sin(\omega_d t_i).$$

A_i and t_i are the amplitudes and time locations of the impulses. n is the number of impulses in the impulse sequence, t_n is the time location of the final impulse.

To generate an impulse sequence that causes no residual

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

vibration, we can set V in (34) equal to zero and solve for the impulse amplitudes and time locations. And we also need placing a few more restrictions on the impulses because there are an infinite number of solutions that satisfy this condition. Here we can constrain the impulse as

$$\sum_{i=1}^{n} A_i = 1, \tag{35}$$

$$A_i > 0 \quad i = 1, 2, \cdots, n. \tag{36}$$

If we try to find a two-impulse sequence shown in Fig. 6 with the time location of the first impulse t_1=0, then this problem is now reduced to solve three unknowns A_1, A_2, t_2. Now according to (34) with the constrained conditions (35) and (36), the sequence of two impulses that leads to zero vibration can be stated in matrix form as

$$\begin{bmatrix} A_i \\ t_i \end{bmatrix} = \begin{bmatrix} \frac{1}{1+P} & \frac{P}{1+P} \\ 0 & 0.5T_d \end{bmatrix} \quad i = 1, 2, \tag{37}$$

where

$$P = e^{\left(-\zeta\pi/\sqrt{1-\zeta^2}\right)},$$
$$T_d = 2\pi/\omega_n\sqrt{1-\zeta^2}.$$

The impulses given in (37) are commonly referred to as a Zero Vibration (ZV) impulse sequence because they result in no vibration.

If we hope the input shaper have robustness to modeling errors, the Zero Vibration and Derivative (ZVD) shaper may be introduced here. This shaper was designed by requiring the partial derivative of the residual vibration, with respect to the frequency, to be equal to zero at the modeling frequency and this can be stated as

$$\frac{\partial V(\omega, \zeta)}{\partial \omega} = 0. \tag{38}$$

Enforcing this constraint has the effect of keeping the vibration near zero as the actual frequency starts to deviate from the modeling frequency. The result of this shaper is given as

$$\begin{bmatrix} A_i \\ t_i \end{bmatrix} = \begin{bmatrix} \frac{1}{(1+P)^2} & \frac{2P}{(1+P)^2} & \frac{P^2}{(1+P)^2} \\ 0 & 0.5T_d & T_d \end{bmatrix} \quad i = 1, 2, 3. \tag{39}$$

Note that this ZVD shaper has duration of one vibration period. However, for a small increase in rise time, a substantial amount of robustness is obtained.

B. Vibration Suppression

Fig. 7 shows the vibration suppression of the flexible beam with the ZVD shaper. Here only the frequency of the first mode which resulted in the vibration mainly was considered. The top plot shows strain data after applying one testing input. The dash line is the result without any vibration suppression control. And the solid line is the condition which torque is modified through ZVD input shaper. The bottom plot shows the control torques for the second link. Here it shows the torques before and after modified through the input shaper. Then we can find that the vibration has been suppressed after applying the ZVD shaper to the optimal control torque. But there also shows some time delay after using the ZVD input shaper. It is the price of vibration suppression by applying this method.

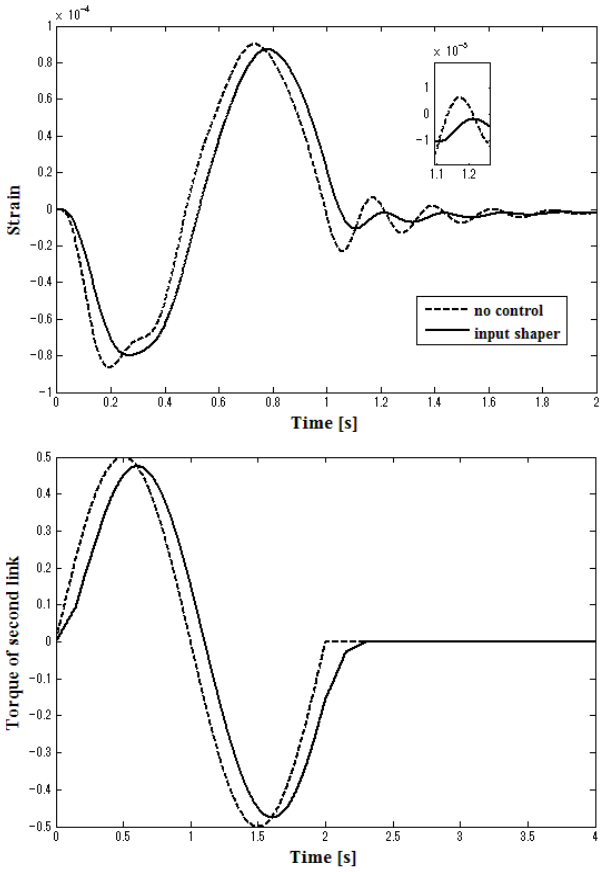


Fig. 7 Vibration suppression by input shaper

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf

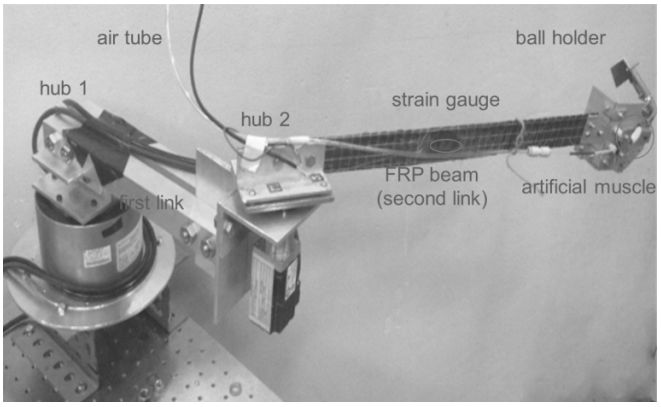


Fig. 8 Experimental setting

VI. Experiment

In this section, an experiment manipulator is conducted to check the accuracy of the model which was built in Section II. The real experimental setting is shown as Fig. 8. The first link is the rigid one driven by a direct driven motor. The flexible link enforced by the FRP materials is controlled by a servo motor. The finger of the ball holder is controlled by the artificial muscle which is actuated through the pneumatic system. So an air tube can be found on Fig. 8. The strain gauge attached on surface of the flexible link is used for gathering the strain data which will be recorded after being amplified.

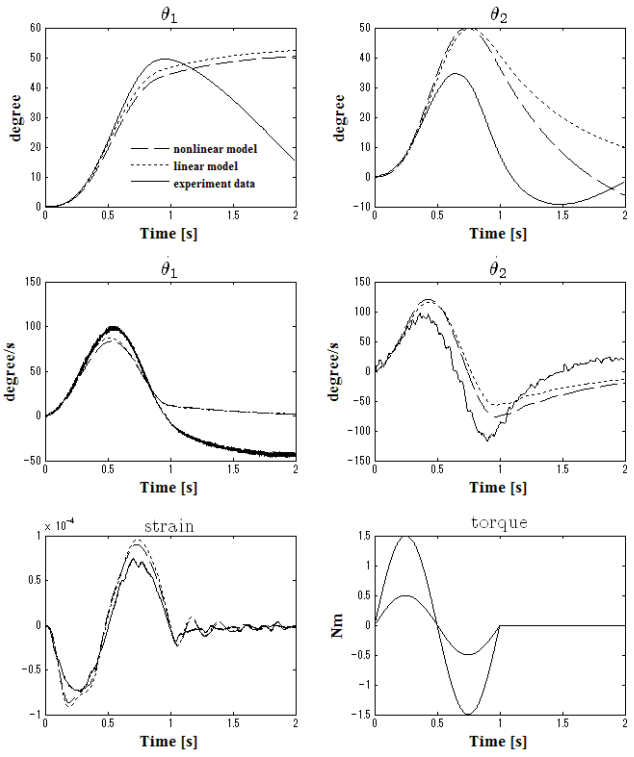


Fig. 9 Model checking

The comparison of results between simulation and experiment are shown in Fig. 9. The solid lines depict the experimental data; the broken lines show simulation results of the nonlinear model; the dotted lines here are the simulation conditions of the linearized model. The plot on the bottom right shows the control torques for the two links here. The left and the right plots on the top illustrate the rotational angle conditions of the first link and flexible link. At the beginning, we found that the simulation matched the experiment data exactly, then after some time, the errors increased rapidly. There may be two reasons for this problem. One reason can be found on middle left plot and middle right plot which show the speed conditions of the first link and flexible link. At first, the speeds of two links were matched well, but gradually the errors increased obviously which could cause the big errors of two links. The other reason for this result is that the friction coefficients of the driving motors were not measured exactly on the current stage which would also lead to the big errors. However the result of the strain gauge of the flexible beam on the bottom left plot is acceptable. Then the nonlinear model owns better performance than linearized model.

VII. Conclusion

This paper derived the state equation for the ball throwing manipulator with one flexible link. In order to obtain the optimum torque input style, a CEVEPSO algorithm was introduced to optimize the governing parameters of the input. By applying the CEVEPSO algorithm, the Pareto Fronts of this problem were evaluated roughly. Then the residual vibration of the flexible link was suppressed through ZVD shaper. Finally, an experiment was operated for accuracy checking of the system model.

Acknowledgment

This work was supported Grant-in Aid for Scientific Research (KAKENHI 22560215) from Japan Society for the Promotion of Science. The authors thank for support for this work by the China Scholarship Council and Hokkaido University.

References

[1] W. Mori, J. Ueda and T. Ogasawara, "A 1-dof dynamics pitching robot that independently controls velocity, angular velocity and direction of a ball", *Advanced Robotics*, pp. 921-942, 2012.

[2] Y. Hoshino and Y. Kobayashi, "Shock and vibration control of a golf-swing robot at impacting the ball", Journal of System Design and Dynamics, vol. 2, no. 5.

[3] R. C. Eberhart and Y. Shi, *Computing Intelligence*. Morgan Kaufmann Publishers, 2007.

[4] Y. Gai, Y. Kobayashi, R. Yamakawa, Y. Hoshino and T. Emaru, "Motion control of a flexible robotic arm by utilizing its dynamics", Asia-Pacific Vibration Conference, vol. 4, pp.1971-1979, 2011.

[5] M. I. Solihin, W. Akmeliawati and R. Akmeliawati, "PSO-based optimization of state feedback tracking controller for a flexible link manipulator",*International Conference of Soft Computing and Pattern Recognition*, pp. 72-76, 2009.

[6] A. Shayeghi, H. Shayeghi and H. E. Kalasar, "Application of PSO technique for seismic control of tall building", *World Academy of Science, Engineering and Technology*, pp. 851-858, 2009.

[7] Nadia Nedjah, Leandro dos Santos Coelho and Luiza de Macedo de Mourelle, *Multi-Objective Swarm Intelligent Systems*. Springer, 2009.

[8] K. E. Parsopoulos and M. N. Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Reference, 2009.

[9] J. G. Vlachogiannis and K. Y. Lee, "Multi-objective based on parallel vector evaluated particle swarm optimization for optimal steady-state performance of power systems", *Expert Systems with Applications*, pp. 10802-10808, 2009.
[10] S. N. Omkar, D. Mudigere, G. N. Naik and S. Gopalakrishnan, "Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures", *Computers and Structures*, pp. 1-14, 2008.
[11] H. Jiang, C. K. Kwong, Z. Chen and Y. C. Ysim, "Chaos particle swarm optimization and T-S fuzzy modeling approaches to constrained predictive control", *Expert Systems with Applications*, pp. 194-201, 2012.
[12] B. Alatas, E. Akin and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms", *Chaos, Solitons and Fractals*, pp. 1715-1734, 2009.
[13] W.Singhose and W. Seering, *Command Generation for Dynamics Systems*. Georgia Institute of Technology, 2011, pp. 33-54.

Open Science Index, Computer and Information Engineering Vol:7, No:7, 2013 publications.waset.org/16363/pdf