Journal of Advanced Engineering Vol. 6, No. 3, pp. 177-182 / July 2011

使用「直接定位及非線性規劃」設計機器手臂軌跡路徑
Three-link Manipulator Optimal Path Planning Using Direct Collocation and Nonlinear Programming

陳軍杰[*1] 林柏旭[2] 旭 葉[1]
Jyun-jye F. Chen[*1], Pohsu Lin[2], Yeshi[1]

摘要

本文使用「直接定位與非線性規劃」(DCNLP) 方法來尋找機器手臂最佳化操作及軌跡。一旦「兩端點邊界值問題 TPBVP」被轉成「參數最佳化問題」，DCNLP 便可著手尋找最佳化數值解。本文使用 Lagrange-Euler 公式模擬一具三連桿機器手臂，並假此手臂來闡明求解過程。文中的機器手臂須以最節能的最佳化操作、來接住一顆掉落中的球，接球所需之時間t_f須由最佳化技術決定，在手臂工作空間、同時存有一顆路徑障礙球，由於t_f及障礙球的存在，更加深了問題深度。
關鍵詞：機器人手臂，軌跡規劃，直接定位，最佳化控制，非線性規劃

Abstract

This study applies the Direct Collocation and Nonlinear Programming (DCNLP) method to lay out the solution for manipulator arm path planning problem. It also demonstrates how DCNLP can efficiently handle the two-point boundary-value problem (TPBVP) by converting a TPBVP into a parameter optimization problem. A three-link manipulator arm, modeled by Lagrange-Euler equation, is used as a vehicle to illustrate the procedures. The manipulator is required to intercept a falling target and a least-energy maneuver sequence control is anticipated. The final time is unspecified and an obstacle sphere is placed in the workspace.
Keywords: robot manipulator, path planning, direct collocation, optimal control, nonlinear programming

I. INTRODUCTION

Optimal control problems modeled by Euler-Lagrange (E-L) equations often end up with two-point boundary-value problems (TPBVP) [1]. The differential equation set of TPBVP is difficult to solve since the boundary values of some variables are given at the initial time t_0 and the rest are given at the final time t_f. Conventional solvers may integrate the differential equation set forward in time and iterate on the variables until all the boundary conditions are satisfied. The drawback of this kind of approaches is that the convergence is not guaranteed.

The method of Direct Collocation and Nonlinear Programming (DCNLP) converts a TPBVP into a parameter optimization problem by quantizing the trajectories of the state variables into small segments in time [2]. Parameter optimization is a commonly used method in nonlinear programming. The method itself is straightforward in tackling the problem but it requires large computer memory space and fast CPU. These are the tradeoffs. However the two tradeoffs are no longer practical concerns now-a-day due to the inexpensive costs of memory chip and fast computer. Thus this conversion guarantees the desired convergence in finding the solutions for optimal control problems. The above-mentioned advantages turn out to be the fortes of DCNLP and these advantages make DCNLP an attractive numerical solver. This paper adopts the algorithm of DCNLP and explores how the algorithm can be applied on manipulator path planning design. It makes this work unique and different from previous works [3-6] which focus on generic algorithm [7] or PID control [8], for examples.

The main idea in this work is to apply DCNLP to find out the optimal trajectory path planning for a three-link robot arm. The computer has to take the path obstacle (constraints) in the workspace into account when laying out the trajectory of $\vec{x}^*$, and in the mean time finding out the control $\vec{u}^*$ which yields least maneuvering energy.

Path planning of manipulator can be approached in numerous ways but it becomes challenging if optimization,

[1] 皇家布丹大學電機工程學系 [2] 國立高雄第一科技大學系統資訊與控制研究所
[*]Corresponding author. E-mail: felipefelipechen@yahoo.com.tw
[1] Department of Electrical Engineering, College of Science and Technology, Royal University of Bhutan, Phuentsholing, Bhutan
[2] Institute of System Information and Control, National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan, R.O.C.
Manuscript received 22 June 2010; revised 16 November 2010; accepted 17 December 2010

i.e., least maneuvering energy, is involved in the planning. This paper not only constructs a three-link manipulator model which operates in a workspace restrained with path obstacles in its operation, but also looks into the Lagrange multiplier functions of the dynamic system which carry the information about the cost sensitivity regarding the optimality. By observing the behavior of the Lagrange multipliers of the dynamic system, one may get a better overall feeling in how to best operate the manipulator with path obstacles included in the consideration.

Part II builds the model of the manipulator arm and constructs the equations of motion of the model. It briefly talks about the idea of DCNLP. Part III demonstrates the simulation results and the conclusions are drawn in Part IV.

II. MATHEMATICAL MODEL AND DYNAMICS

1. Problem Setup

An obstacle sphere is placed in the workspace of a three-link manipulator arm. The arm is required to intercept a falling target with least-energy maneuver sequence of control. The end-effectors, Joints 2 and 3 should not slew into the obstacle sphere which serves as a path constraint.

2. Mathematical Modelling

The three-link manipulator arm configuration is shown in Fig. 1. Six assumptions are made to simplify the derivations:

a. The links are assumed to be homogeneous and slim in order to simplify the inertia matrix.
b. The links are assumed to be stiff so that there is no possibility for structure vibration when the manipulator moves in high speed.
c. There is no friction at the joints.
d. There are no upper and lower bounds for the states and controls.
e. The obstacle is made into a sphere.
f. The length L_i and mass m_i of each link are selected to be 1 meter and 1 kilogram respectively in order to simplify the calculations. The inertia matrix J_i is as follows.

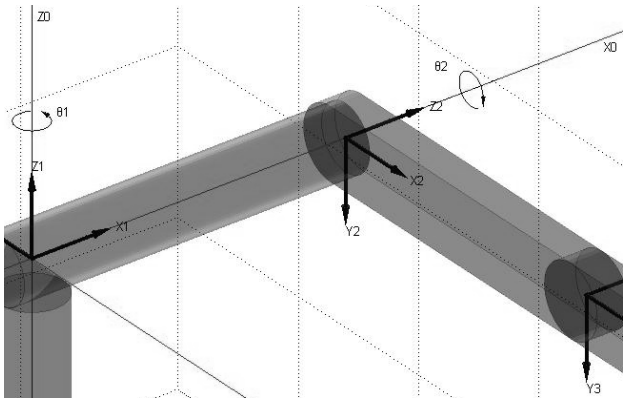

Fig. 1 Three-link robot arm configuration

$$J_i = \begin{bmatrix} \frac{1}{3}m_i L_i^2 & 0 & 0 & \frac{1}{2}m_i L_i \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2}m_i L_i & 0 & 0 & m_i \end{bmatrix} \quad (1)$$

The dynamics of the arm is described by the Lagrange-Euler (L-E) equations [9].

$$\vec{\tau} = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\vec{q}}}\right) - \frac{\partial L}{\partial \vec{q}} \quad (2)$$

$\vec{q} = [q_1 \quad q_2 \quad q_3]^T$ is the vector of joint angles. Since all the joints are rotational, the generalized variable q_i may be replaced by θ_i, i.e., $\vec{q} = [\theta_1 \quad \theta_2 \quad \theta_3]^T$. $\vec{\tau} = [\tau_1 \quad \tau_2 \quad \tau_3]^T$ is the output vector of the actuators at Joints 1, 2 and 3. L is the lagrangian function of the system and

$$L = \frac{1}{2}\sum_{i=1}^{3}\sum_{j=1}^{i}\sum_{k=1}^{i}\left[Tr\left(U_{ij} J_i U_{ik}^T\right)\cdot \dot{q}_j \dot{q}_k\right] + \sum_{i=1}^{3} m_i \vec{g}\left({}^0A_i\, {}^i\vec{r}_i\right) \quad (3)$$

where $\vec{g} = [0 \quad 0 \quad -9.81 \quad 0]$ is the gravitation acceleration vector, ${}^i\vec{r}_i$ is the position vector of the mass center on Link i with respect to Frame i, ${}^jA_i$ is the Denavit-Hartenberg (D-H) transformation matrix between Frame i and Frame j, $U_{ij} = \begin{cases} {}^0A_{j-1} Q_j\, {}^{j-1}A_i & 0 \le j < i \\ 0 & j > i \end{cases}$, $0 \le i, j \le 3$, and $Q_i = \frac{\partial\left({}^{i-1}A_i\right)}{\partial\theta_i}\cdot\left({}^{i-1}A_i^{-1}\right)$, $0 \le i \le 3$.

Then (2) can be organized into the following form.

$$\vec{\tau}(t) = D(\vec{\theta})\cdot\ddot{\vec{\theta}}(t) + h\left(\vec{\theta}, \dot{\vec{\theta}}\right) + c\left(\vec{\theta}\right) \quad (4)$$

where $D(\vec{\theta})$ is the coefficient matrix of acceleration terms, $h(\vec{\theta})$ is the collection of the terms associated with Coriolis effect, and $c(\vec{\theta})$ contains gravitational terms as in [9]. Multiply (4) with $D(\vec{\theta})^{-1}$, (4) becomes

$$\ddot{\vec{\theta}}(t) = D(\vec{\theta})^{-1}\cdot\left(-h\left(\vec{\theta}, \dot{\vec{\theta}}\right) - c(\theta) + \vec{\tau}(t)\right) \quad (5)$$

Let $\vec{x}(t) = [\theta_1 \quad \omega_1 \quad \theta_2 \quad \omega_2 \quad \theta_3 \quad \omega_3]^T$ be the state vector and $\vec{u}(t) = [\tau_1 \quad \tau_2 \quad \tau_3]^T$ be the control vector, then (5) can be transformed into six first-order differential equations

$$\dot{\vec{x}}(t) = \vec{f}\left(\vec{x}, \vec{u}, t_f, t\right)_{6\times 1} \quad (6)$$

where ω_i denotes the angular velocity of θ_i. Before DCNLP can take over, (6) has to be quantized into n tiny homogeneous segments in time. This generates $n+1$ $\vec{x}[k]$ at the $n+1$ nodes of the n segments.

In this study, $n = 100$. Let $\vec{f} = [f_1 \quad f_2 \quad f_3 \quad f_4 \quad f_5 \quad f_6]^T$, $t = k\cdot\Delta t$ and $\Delta t = \frac{t_f}{n}$, then (6) can be approximated as

$$\begin{bmatrix} \theta_1[k+1] \\ \omega_1[k+1] \\ \theta_2[k+1] \\ \omega_2[k+1] \\ \theta_3[k+1] \\ \omega_3[k+1] \end{bmatrix} = \begin{bmatrix} \theta_1[k] + \omega_1[k]\cdot\Delta t \\ \omega_1[k] + f_2[k]\cdot\Delta t \\ \theta_2[k] + \omega_2[k]\cdot\Delta t \\ \omega_2[k] + f_4[k]\cdot\Delta t \\ \theta_3[k] + \omega_3[k]\cdot\Delta t \\ \omega_3[k] + f_6[k]\cdot\Delta t \end{bmatrix}, \quad k = 1,2,\cdots 100 \quad (7)$$

Equation (7) contains 600 equality constraints.

3. The Obstacle Constraints

How does one incorporate the path constraints into this parameter optimization problem? The answer is: "as long as the distances (d) between the three concerned points (Joint 2, Joint 3 and the end-effectors) and the center of the obstacle sphere are larger than the radius of the sphere (R_s) at any time, the arm stays clear from the obstacle sphere."

$$\begin{cases} R_s < \sqrt{(x[k]-x_s)^2+(y[k]-y_s)^2+(z[k]-z_s)^2} < \infty \\ R_s < \sqrt{(x_2[k]-x_s)^2+(y_2[k]-y_s)^2+(z_2[k]-z_s)^2} < \infty, k=1,\cdots,n+1 \\ R_s < \sqrt{(x_3[k]-x_s)^2+(y_3[k]-y_s)^2+(z_3[k]-z_s)^2} < \infty \end{cases} \quad (8)$$

a. R_s is the radius of the sphere, a given constant.

b. $\vec{r}_s = [x_s \quad y_s \quad z_s]^T$ is the given coordinate vector of the center of the sphere.

c. $\vec{r}[k] = (x[k] \quad y[k] \quad z[k])^T$ is the coordinate vector of the end-effectors of the arm at any time node k. $\vec{r}_2[k] = (x_2[k] \quad y_2[k] \quad z_2[k])^T$ is the coordinate vector of Joint 2 and $\vec{r}_3[k] = (x_3[k] \quad y_3[k] \quad z_3[k])^T$ is the one of Joint 3.

d. $$\begin{cases} x[k] = (S_1C_2C_3 - S_1S_2S_3)\cdot L_3 + S_1C_2\cdot L_2 + C_1\cdot L_1 \\ y[k] = (-C_1C_2C_3 + C_1S_2S_3)\cdot L_3 - C_1C_2\cdot L2 + S_1\cdot L_1 \\ z[k] = (-S_2C_3 - C_2S_3)\cdot L_3 - S_2\cdot L_2 \end{cases} \quad (9)$$

e. $S_i = \sin(\theta_i[k])$, $C_i = \cos(\theta_i[k])$.

The coordinates of Joint 3 at any time node k are found by assigning L_3 in (9) to be zero. The coordinates of Joint 2 are found by making both L_2 and L_3 to be zero.

4. Intercepting a Falling Target

Since the manipulator has to intercept a falling target, it must complete the task before the target falls away beyond reach. In this way, the final time t_f cannot be infinite. The coordinates of the target at t_f are

$$x_T(t_f) = x_0 + V_x\cdot t_f \quad (10)$$

$$y_T(t_f) = y_0 + V_y\cdot t_f \quad (11)$$

$$z_T(t_f) = z_0 + V_z\cdot t_f + \frac{1}{2}9.81\cdot t_f^2 \quad (12)$$

To intercept the target at the final time, the following constraints must hold.

$$x[101] - x_T(t_f) = 0 \quad (13)$$

$$y[101] - y_T(t_f) = 0 \quad (14)$$

$$z[101] - z_T(t_f) = 0 \quad (15)$$

5. The Cost Function

The control $\vec{u}[k]$ is placed at the center of segment k. Equations (7), (13), (14), and (15) consist of 603 equality constraints. Equation (8) contributes 300 inequality constraints. Carefully choosing $\vec{x}[k]$, $\vec{u}[k]$ and t_f by the techniques of parameter optimization and nonlinear programming, one may yield the optimal control $\vec{u}^*[k]$ which minimizes the cost function

$$J = \frac{1}{2}\sum_{k=1}^{n}\left(\tau_1^2[k] + \tau_2^2[k] + \tau_3^2[k]\right)\cdot\frac{t_f}{n}, \quad n = 100 \quad (16)$$

without violating the constraints. At this point, the entire differential TPBVP has lost the sensation of being differential. Instead it has become a straightforward parameter optimization problem which tries to minimize J in (16) subject to 903 constraints.

The final time t_f is also singled out as one of the parameters to be determined by the optimizer. This t_f is a crucial parameter since it profoundly affects all the states and control elements at all the time nodes.

The leftmost column in Table 1 is the collection of the 902 constraints and the cost function. The top row in Table 1 collects the states, controls and t_f. Users need to prepare the Jacobian matrix as shown in Table 1. The optimizer takes the information from the 903×907 Jacobian matrix and activates the time consuming nonlinear programming iteration process. Note that there are two band matrices inhabiting in Table 1. The optimizer has considered the appearance of band matrix and has taken the advantages of it in speeding up its numerical analysis.

Nonlinear programming technique is the main tool used for refining data in this optimizer software. It also explains why the method is called Direct Collocation and Nonlinear Programming.

III. SIMULATIONS AND RESULTS

In Part III, several simulations are carried out and the results are displayed in the following sections.

1. Case 0—Nominal Trajectory

The target ball is released at an arbitrarily picked coordinates of $(1.0 \quad 0.5 \quad 0.0)$ with no initial velocity. The manipulator arm is parked at its parking position, i.e., $\vec{\theta}(0) = [0.0 \quad 0.0 \quad 0.0]^T$ and begins to chase the target once the target is released. The simulated nominal trajectory is displayed in Fig. 2 which contains eleven freeze-frames. A smoke bomb is attached at the end of the manipulator in order to visualize the trace of the end tip. The simulation results are collected into Table 2.

2. Cases 1~10—Simulations with Obstacle Sphere Present

According to the optimal control theory, the cost J will increase when the constraint becomes tougher. In this part, a sphere is placed right in the pathway of the arm. Ten optimal trajectory motions are simulated by increasing the radius of the obstacle R_s from 0.0 meter to 1.0 meter, each time with an incremental step of 0.1 meter. Figs. 3 and 4 demonstrate the trends of the cost J and final time t_f when the radius of the obstacle sphere is increased gradually. It is worth while to look at the notch of t_f at R_s =0.7 in Fig. 4.

Table 1 Jacobian matrix of the constraint and cost functions

	$\vec{x}[1]_{6\times1}^T$	$\vec{\tau}[1]_{3\times1}^T$	$\vec{x}[2]_{6\times1}^T$	$\vec{\tau}[2]_{3\times1}^T$	$\vec{x}[3]_{6\times1}^T$	$\vec{\tau}[3]_{3\times1}^T$	$\cdots$	$\vec{x}[99]_{6\times1}^T$	$\vec{\tau}[99]_{3\times1}^T$	$\vec{x}[100]_{6\times1}^T$	$\vec{\tau}[100]_{3\times1}^T$	$\vec{x}[101]_{6\times1}^T$	t_f
$\vec{f}_{6\times1}[1]=0$	$\partial\vec{f}[1]/\partial\vec{x}[1]$	$\partial\vec{f}[1]/\partial\vec{\tau}[1]$	$\partial\vec{f}[1]/\partial\vec{x}[2]$	$\partial\vec{f}[1]/\partial\vec{\tau}[2]$									$\partial\vec{f}[1]/\partial t_f$
$\vec{f}_{6\times1}[2]=0$			$\partial\vec{f}[2]/\partial\vec{x}[2]$	$\partial\vec{f}[2]/\partial\vec{\tau}[2]$	$\partial\vec{f}[2]/\partial\vec{x}[3]$	$\partial\vec{f}[2]/\partial\vec{\tau}[3]$							$\partial\vec{f}[2]/\partial t_f$
$\vec{f}_{6\times1}[3]=0$					$\partial\vec{f}[3]/\partial\vec{x}[3]$	$\partial\vec{f}[3]/\partial\vec{\tau}[3]$	$\cdots$						$\partial\vec{f}[3]/\partial t_f$
$\vdots$							$\ddots$						$\vdots$
$\vec{f}_{6\times1}[99]=0$								$\partial\vec{f}[99]/\partial\vec{x}[99]$	$\partial\vec{f}[99]/\partial\vec{\tau}[99]$	$\partial\vec{f}[99]/\partial\vec{x}[100]$	$\partial\vec{f}[99]/\partial\vec{\tau}[100]$		$\partial\vec{f}[99]/\partial t_f$
$\vec{f}_{6\times1}[100]=0$										$\partial\vec{f}[100]/\partial\vec{x}[100]$	$\partial\vec{f}[100]/\partial\vec{\tau}[100]$	$\partial\vec{f}[100]/\partial\vec{x}[101]$	$\partial\vec{f}[100]/\partial t_f$
$\vec{R}_S < \vec{f}_{3\times1}[1]$	$\partial\vec{f}[1]/\partial\vec{x}[1]$												
$\vec{R}_S < \vec{f}_{3\times1}[2]$			$\partial\vec{f}[2]/\partial\vec{x}[2]$										
$\vec{R}_S < \vec{f}_{3\times1}[3]$					$\partial\vec{f}[3]/\partial\vec{x}[3]$								
$\vdots$							$\ddots$						
$\vec{R}_S < \vec{f}_{3\times1}[99]$								$\partial\vec{f}[99]/\partial\vec{x}[99]$					
$\vec{R}_S < \vec{f}_{3\times1}[100]$										$\partial\vec{f}[100]/\partial\vec{x}[100]$			
$\vec{R}_S < \vec{f}_{3\times1}[101]$												$\partial\vec{f}[101]/\partial\vec{x}[101]$	
$f_{900}=0$												$\partial f_{900}/\partial\vec{x}[101]$	$\partial f_{900}/\partial t_f$
$f_{901}=0$												$\partial f_{901}/\partial\vec{x}[101]$	$\partial f_{901}/\partial t_f$
$f_{902}=0$												$\partial f_{902}/\partial\vec{x}[101]$	$\partial f_{902}/\partial t_f$
$J = f_{903}$		$\partial J/\partial\vec{\tau}[1]$		$\partial J/\partial\vec{\tau}[2]$		$\partial J/\partial\vec{\tau}[3]$	$\cdots$		$\partial J/\partial\vec{\tau}[99]$		$\partial J/\partial\vec{\tau}[100]$		$\partial J/\partial t_f$

Table 2 Simulation results

Case	R_S (m)	Cost J	t_f (sec)
0	0.0	191.6247	0.6142
1	0.1	191.6247	0.6142
2	0.2	191.6247	0.6142
3	0.3	191.6247	0.6142
4	0.4	203.4836	0.6201
5	0.5	283.8874	0.6281
6	0.6	398.5822	0.6252
7	0.7	540.0401	0.6111
8	0.8	765.6784	0.6250
9	0.9	916.0312	0.6271
10	1.0	1068.7668	0.6283

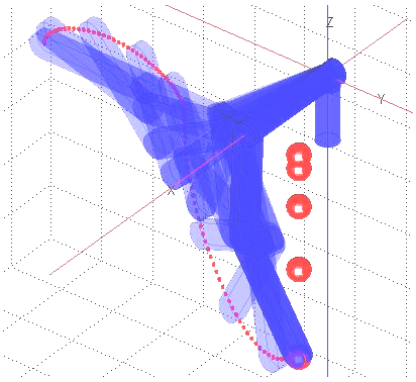

Fig. 2 Trajectory of Case 0

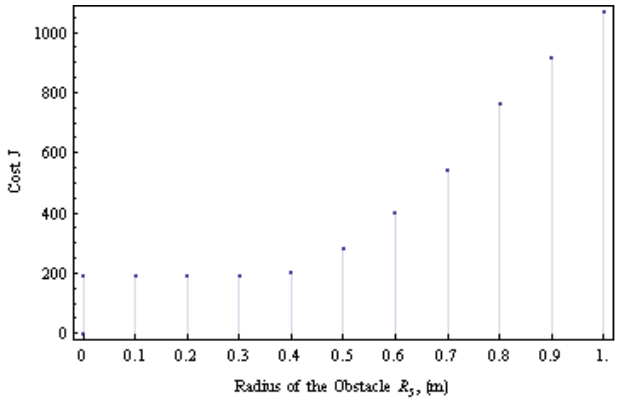

Fig. 3 Cost J against various obstacle radius R_S

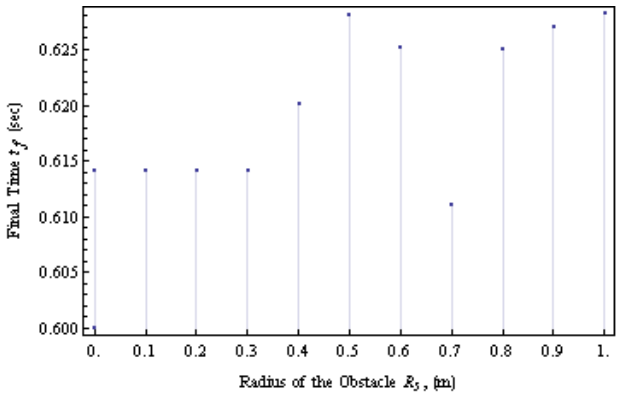

Fig. 4 Final time t_f against obstacle radius R_S

The three joint angles for each R_s are displayed in Figs. 5, 6 and 7. The three torque outputs are displayed in Figs. 8, 9 and 10.

Eight out of the eleven simulations are put into a single graph Fig. 11 in order to illustrate the path trajectories.

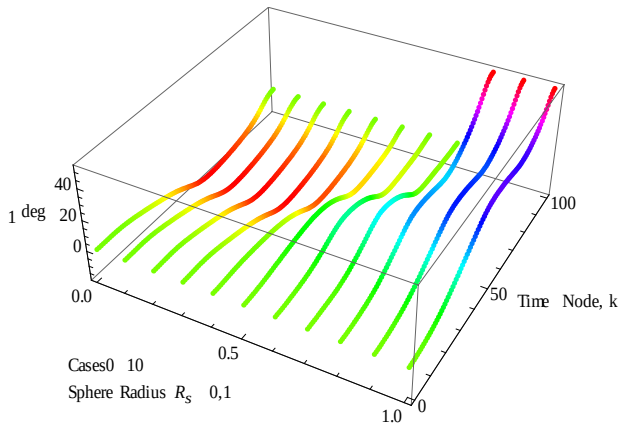

Fig. 5 θ_1 with respect to R_s

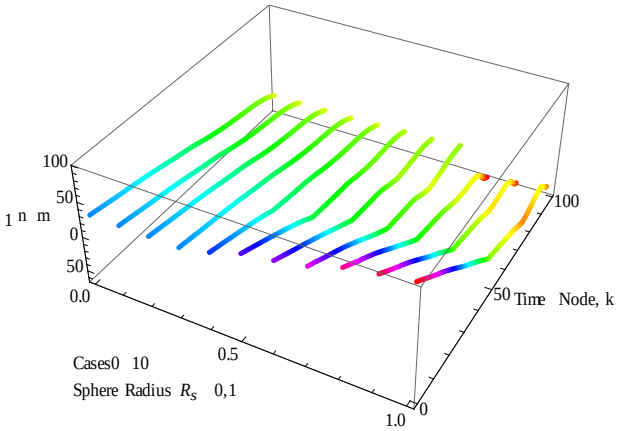

Fig. 8 τ_1 with respect to R_s

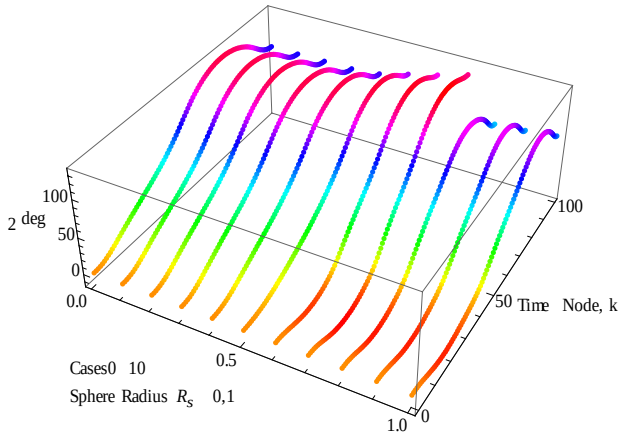

Fig. 6 θ_2 with respect to R_s

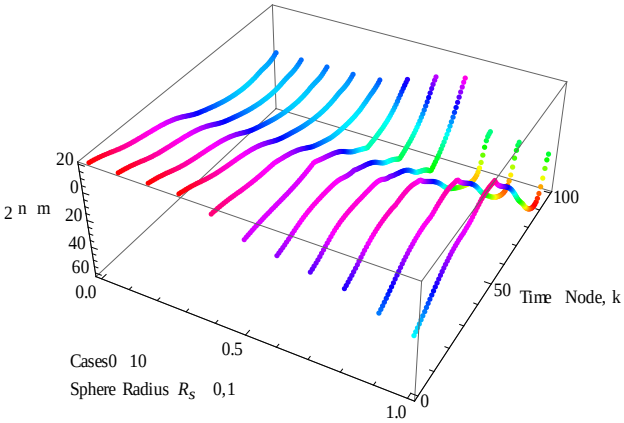

Fig. 9 τ_2 with respect to R_s

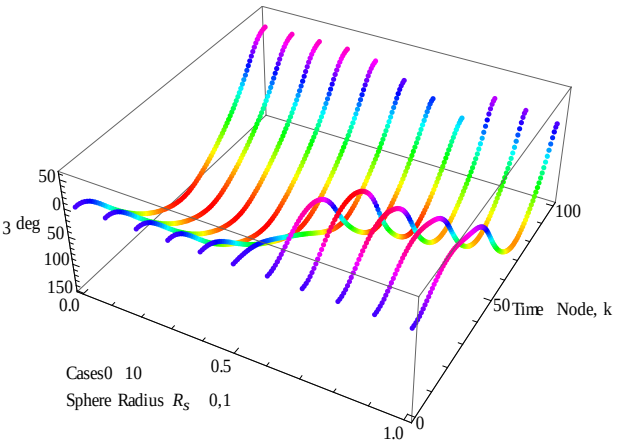

Fig. 7 θ_3 with respect to R_s

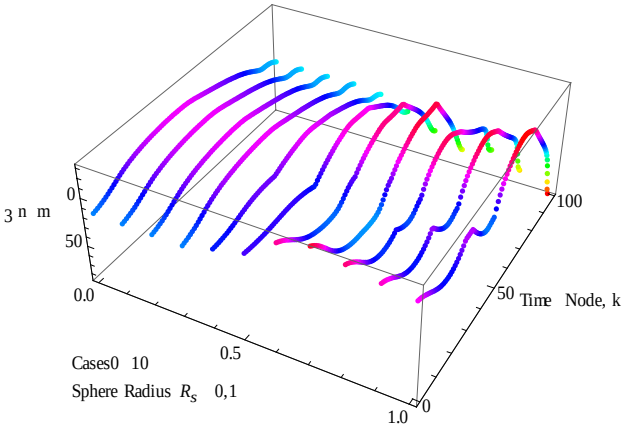

Fig. 10 τ_3 with respect to R_s

It is indispensable to validate if the simulations are correct before one may want to proceed to the results. In Case 0, the target travels through the air for 0.6142 seconds according to Table 2. On the other hand, the state vector of the three joints at the final time t_f is $\vec{\theta}[101]$ or $(0.0000°\quad 88.1012°\quad 32.9495°)^T$. Substitute the three angles into (9) and yield $|z[101]| = 1.8561$. By Newton's laws it takes any object 0.6142 seconds to drop 1.8503 meters in a free fall. This validates that the simulation is reliable and the difference between 1.8561 and 1.8503 is caused by the inaccuracy in the numerical iterations.

In Fig. 2, it can be seen that Link 2 and Link 3 try to fold up when the manipulator is retrieving. It makes sense since smaller moment of inertia leads to lower kinetic energy required in a rotational motion. The manipulator extends its arm again when the rotational motion is about to end. On an ad hoc and impromptu situation, it is unnecessarily obvious for a human to capture this folding-and-unfolding trick.

By Table 2, the results of Cases 0, 1, 2 and 3 are exactly same since the obstacle sphere has not yet grown large enough to serve as an obstacle. Case 4 is the first case confronted by the obstacle sphere.

Fig. 11 shows that the manipulator avoids the obstacle as is required and satisfies the path constraints in each case. In Fig. 11, it is also observed that Cases 5, 6, and 7 group themselves into a cluster and Cases 8, 9 and 10 group themselves into another one at the end of the motions. It seems there is an invisible gap between Case 7 and Case 8 according to Fig. 11. Also there is a significant notch appears in Fig. 4. For the case of $R_S = 0.7$ meters, t_f looks unusually smaller than its neighboring numbers yet its cost J looks pretty regular.

IV. CONCLUSIONS

This study applies least-energy control theory to lay out the path planning for a manipulator. In doing so, TPBVP is inevitably involved in the optimization processes.

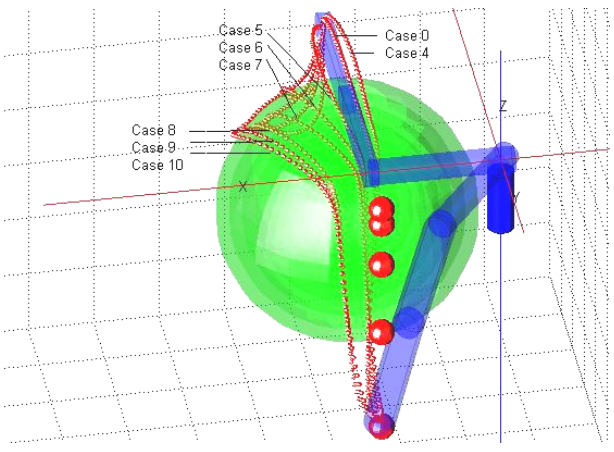

Fig. 11 Manipulator arm brushing through seven obstacle spheres with R_S chosen to be 0.0, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9 and 1.0 meter

Therefore this study suggests a new approach in solving the TPBVP encountered in the optimal control part. By using DCNLP, the TPBVP is converted into a parameter optimization problem. The solution approach is then very straightforward.

Without the helps of the optimal control theory, it would be hard for any human to come up with elegant solutions due to the highly nonlinear equations of motion of a three-link manipulator arm. The method of DCNLP, introduced in this study, successfully completes the tasks as shown in Table 2, Fig. 2 and Fig. 11. The simulated results match with the common senses of dynamics.

The study also discovers that there are at least three converged solutions exist for Case 7. It seems the zonal area between Case 7 and Case 8 is populated with local minimum solutions. It is a tangible benefit if the actual causes, which veer the convergence from the anticipated global minimum to a local minimum, can be identified. Fully understanding of the causes can help one accurately unearth the global minimum solution.

The length of the CPU time of each simulation varies from 0.5 to 9.0 sec. The cases are run on an AMD Athlon 2.08 GHz machine running Linux 9.0 operating system. These CPU times are too large to be practical since the final time t_f in the interception is not larger than 0.628 sec. Better hardware can help in racing against the clock.

ACKNOWLDEGMENTS

The programming work and numerical iterations in this work were done when the first and second authors were at National Penghu University, Penghu, Taiwan in the year of 2008. The purchase of the optimizer software (SNOPT 5.3) was funded by National Science Committee, Taiwan. Project number: NSC-20050524.

REFERENCES

[1] A. E. Jr. Bryson and Y. Ho, *Applied Optimal Control*, Hemisphere Publishing Corp., New York, 1975.
[2] P. E. Gill, W. Murray, and M. A. Saunders, *User's Guide for SNOPT 5.3: A Fortran Package for Large-scale Nonlinear Programming*, Stanford Business Software, Inc., 1998.
[3] M. Mediavilla, J. L. González, J. C. Frail, and J. R. Perán, "Reactive approach to on-line path planning for robot manipulators in dynamic environments," *Robotica*, vol. 20, no. 4, pp. 375-384, 2002.
[4] J. K. Park, "Convergence properties of gradient-based numerical motion-optimizations for manipulator arms amid static or moving obstacles," *Robotica*, vol. 22, no. 6, pp. 649-659, 2004.
[5] S. F. P. Saramago and M. Ceccarelli, "An optimum robot path planning with payload constraints," *Robotica*, vol. 20, no. 4, pp. 395-404, 2002.
[6] G. Alici and B. Shirinzadeh, "Optimum synthesis of planar parallel manipulators based on kinematic isotropy and force balancing," *Robotica*, vol. 22, no. 1, pp. 97-108, 2004.
[7] D. Zhang, Z. Xu, C. M. Mechefske, and F. Xi, "Optimum design of parallel kinematic toolheads with genetic algorithms," *Robotica*, vol. 22, no. 1, pp. 77-84, 2004.
[8] W. Wang, Y. Zhuang, and W. M. Yun, "Innovative control education using a low cost intelligent robot platform," *Robotica*, vol. 21, no. 3, pp. 283-288, 2003.
[9] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill Book Company, 1987.