The Fifteenth International Symposium on Artificial Life and Robotics 2010 (AROB 15th '10),
B-Con Plaza, Beppu,Oita, Japan, February 4-6, 2010

Swing-up Control of the Acrobot Using Genetic Programming Considering an Actuator Dynamics

Ryo Fukushima* and Eiho Uezato**

*Graduate School of Engineering and Science, University of the Ryukyus
**Faculty of Engineering, University of the Ryukyus
1 Senbaru, Nishihara, Okinawa. 903-0213
(Tel:098-895-8638 ; Fax:098-895-3636)
(fukushima@mibai.tec.u-ryukyu.ac.jp)

Abstract: We present a control method for a three-degree-of-freedom (3-DOF) acrobot which is a model of a gymnast on a horizontal bar with three links, two active joints, and a passive joint. This robot is a non-holonomic and underactuated system, the swing-up control of the acrobot is therefore difficult. The active joints of the acrobot use the DC servomotor. We model the 3-DOF acrobot considering the dynamics of the DC servomotor. We propose a control method for the 3-DOF acrobot. First, swing-up control is performed by genetic programming (GP), and stabilizing control is handled by a linear quadratic regulator (LQR). GP can search widely for the optimum input for swing-up so that the acrobot is able to reach a near balancing point. The LQR is then switched on to stabilize the system. In the simulation results, the 3-DOF acrobot could swing-up to the desired position, and the proposed method could control the acrobot effectively.

Keyword: Underactuated robot, Nonlinear system, Genetic programming, Acrobot

1 Introduction

Underactuated robots, which have fewer number of actuators than that of degree-of-freedom, are difficult to design control laws. The development of control system design to the underactuated robots can save energy and reduce the cost and weight. Moreover, they are effective to work in space with no gravity or underwater with low gravity.

The acrobot [1] is one of underactuated robots, and has non-holonomic behavior. It is also known as the model of a gymnast on a horizontal bar. There have been many studies about 2-DOF acrobot[2][3][4]. In this paper, we discuss a control method for the 3-DOF acrobot with two active joints and a passive joint. The active joints are driven by DC servomotors. We model the 3-DOF acrobot considering the actuator dynamics in order to model a more realistic system.

We propose a control method for the 3-DOF acrobot where the swing-up stage is performed by GP and the balancing stage is handled by a linear quadratic regulator (LQR). GP is an approach which has been expanded from genetic algorithm (GA)[5], and it can search widely for the optimum input feedback function for the design of the acrobot control system, which is a difficult problem. Motion control using GP was discussed by Ogawa et al[6]. It is appropriate to use GP for such a difficult control problem as the acrobot.

2 Model of the acrobot

Figure 1 shows the model of the acrobot. m_i $(i = 1, 2, 3)$ and I_i denote the mass and the moment of inertia, respectively; l_i and l_{ci} denote the length of the link and the distance to the center of mass. θ_i is the angle, and h_i is the height to the top of the link. Here, u_2 and u_3 are symbolized as input torques actuated the active joints.

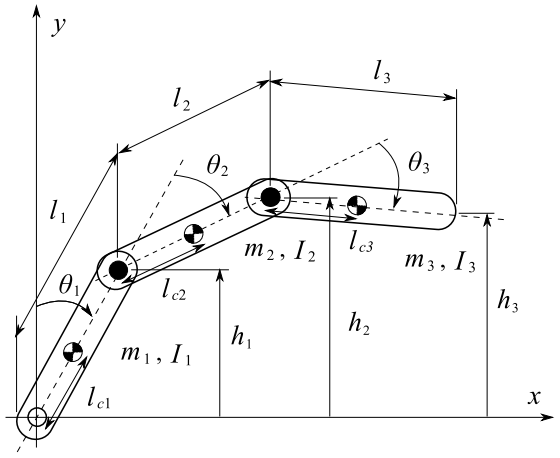

Fig. 1: Model of the acrobot

The equation of the motion of the acrobot system is

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta}) + G(\theta) = Hu, \tag{1}$$

where

$$\theta = [\ \theta_1,\ \theta_2,\ \theta_3\ ]^T, \quad u = [\ u_2,\ u_3\ ]^T,$$

M is an inertial matrix, C is a coriolis term, G is a gravity term and H is constant matrix.

In this study, DC servomotors are applied as the actuators to drive the second and the third joints. We assume that the servomotors are provided with inner feedback loops to control the position angle, therefor, the actuator can be operated by giving a reference position angle. We model the actuator dynamics as

$$u = \begin{bmatrix} -a_2\ddot{\theta}_2 - b_2\dot{\theta}_2 - c_2(\theta_2 - \theta_{a2}) \\ -a_3\ddot{\theta}_3 - b_3\dot{\theta}_3 - c_3(\theta_3 - \theta_{a3}) \end{bmatrix}, \tag{2}$$

where $\theta_a = [\ \theta_{a2},\ \theta_{a3}\ ]^T$ represents reference of the position angle of the acrobot system, and a_i, b_i, c_i are constants including a parameters of the servomotor, for example a torque constant, a reduction ratio. From Eqs. 1 and 2, the equation of the motion of this system is

$$\left\{M(\theta) + \hat{A}\right\}\ddot{\theta} + C(\theta, \dot{\theta}) + \hat{B}\dot{\theta} + G(\theta) + \hat{C}\theta = \hat{C}H\theta_a, \tag{3}$$

where

$$\hat{A} = \mathrm{diag}(0, a_2, a_3), \quad \hat{B} = \mathrm{diag}(0, b_2, b_3),$$
$$\hat{C} = \mathrm{diag}(0, c_2, c_3).$$

We regard the reference angle θ_a as new control input.

3 Control System

Figure 2 shows the block diagram for a closed-loop system. Suffix r represents the desired values. We define deviations as $e = \theta_r - \theta$, $\dot{e} = \dot{\theta}_r - \dot{\theta}$. The control goal is to swing the acrobot up to balancing point ($\theta_r = 0$, $\dot{\theta}_r = 0$). We propose a control method for the acrobot where swing-up stage is performed by GP and balancing stage is controlled by LQR.

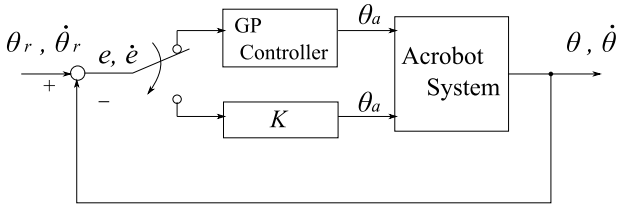

Fig. 2: Block diagram for the closed-loop system

3.1 Swing-up control using GP

GP is one of evolutionary approaches, which is the model of natural selection of genes. As evolution progresses, the individuals adapt to their given environment. In this study, GP searches for the optimum control input (feedback input function $\theta_a\,(\,e, \dot{e}\,)$) for swing-up. Each individual in GP represents tree structure which stands for a function. We operate the tree structure with genetic crossover and mutation so that the tree structure adapts to the given environment. The elements for the design of GP include the function nodes, the terminal nodes, fitness, parameters (crossover rate, mutation rate, population size), and a termination condition. We will get the desired feedback input function when the five elements are set effectively.

The calculation procedure is described below. First, set the number of generation G and the population size N. Generate an initial population, and evaluate the fitness of the population as shown below. Next, performs the crossover and mutation operations to the population with mutation rate α, and generate a new population up to $N \times \beta$, where β is the crossover rate. Evaluate the population generated, and the individual with the highest fitness value is taken into the next generation. Repeat until generation reaches G. We adopt the most excellent individual as the input function of the control system.

The fitness function for evaluating the individuals is

$$E = \min_t \Big[w_1(l_1 - h_1)^2 + w_2(l_1 + l_2 - h_2)^2 + w_3(l_1 + l_2 + l_3 - h_3)^2 + w_4(\dot{\theta}_{r1} - \dot{\theta}_1)^2 + w_5(\dot{\theta}_{r2} - \dot{\theta}_2)^2 + w_6(\dot{\theta}_{r3} - \dot{\theta}_3)^2 \Big], \tag{4}$$

where t represents the time of the swing-up motion, t_f represents the finish time of the swing-up motion, and w_i represent the weight coefficients. The lower the fitness value, the closer the acrobot reaches to the desired position. The "min" in Eq. 4 is the minimum fitness value of each step from the range $0 < t \leq t_f$. Finally, this minimum fitness value is used for one individual only. Here, the first three terms in Eq. 4 are functions relating to the highest marks of each link. To consider the height of each link, the farthest distance from the balancing point causes the high fitness value. θ_2 and θ_3 are restricted to the range $-3\pi/4 < \theta_{2,3} < 3\pi/4$ to limit the rotation of links 2 and 3. If θ_2 and θ_3 exceed the limited range while searching for the desired input function using GP, we add 10^6 to the fitness value of the individual as a penalty.

3.2 Stabilize control at balancing point

The stabilizing control uses a LQR at the near balancing point. If θ_i is sufficiently small ($\theta_i \approx 0$), we

can use the approximations $\sin\theta_i \approx \theta_i$, $\cos\theta_i \approx 1$, and neglect $\dot{\theta_i}^2$. Thus, Eq. 3 is simplified to

$$\left\{\tilde{M} + \hat{A}\right\}\ddot{\theta} + \hat{B}\dot{\theta} + \tilde{G}\theta = \hat{C}H\theta_a, \tag{5}$$

and we have eliminated C, which is the coriolis term, from Eq. 3. Here, the state variable defines $x = [\,\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3\,]^T$, and Eq. 5 is

$$\dot{x} = Ax + B\theta_a, \tag{6}$$

$$A = \begin{bmatrix} 0_{3\times3} & I_{3\times3} \\ -\left\{\tilde{M} + \hat{A}\right\}^{-1}\tilde{G} & -\left\{\tilde{M} + \hat{A}\right\}^{-1}\hat{B} \end{bmatrix},$$

$$B = \begin{bmatrix} 0_{3\times2} \\ \left\{\tilde{M} + \hat{A}\right\}^{-1}\hat{C}H \end{bmatrix}.$$

4 Simulation

We carried out a simulation with a sampling time of 20[ms] and $t_f = 3.0$[s]. The initial values for the state variables were $[\,\theta_1, \theta_2, \theta_3, \dot{\theta}_1, \dot{\theta}_2, \dot{\theta}_3\,] = [\,\pi, 0, 0, 0, 0, 0\,]$, which is the hanging position of the acrobot on the bar. We set the terminal nodes as e, $\dot{e}$ and random real numbers from the range $[-10, 10]$. The function nodes are shown in Table 1. It is expected that the term tanh in the function node would get a better result by inhibiting the input angle. We searched for the individual (input function) which minimized E by a combination of the terminal nodes and the function nodes. Table 2 shows the parameters of GP, and Table 3 shows the parameters of the acrobot system.

The fitness function is very important for getting the optimum input angle. In particular, the setting of the weight coefficients of the fitness function strongly affects the results, but there is no effective way to determine the weight coefficients; those have to be decided by trial and error. First, we evaluate the position of the acrobot by considering the highest marks of each link. That is, the fitness value becomes lower near the balancing point and higher far away from the balancing point. In Eq. 4, the first three terms are the fitness value with the position of each link. It depends on the length of each link. We set the weight coefficients of the first three terms so that the fitness value of the position of each link becomes the value of the same scale. For this reason, the weight coefficients of the first three terms are $w_1 = 40$, $w_2 = 20$, $w_3 = 10$. Next, we evaluate the angular velocity of the acrobot by considering the reaction forces. Each link of the acrobot receives reaction forces from the next link. Thus, if we restrain the angular velocity of link 2, the angular velocities of links 1 and 3 will decrease with the decline of the angular velocity of link 2. As a result, the weight coefficients of the fitness function will be $w_1 = 40$, $w_2 = 20$, $w_3 = 10$, $w_4 = 1$, $w_5 = 10$, $w_6 = 1$.

We determine a switching time from swing-up control to stabilizing control as the time with the lowest fitness value in the simulation.

A criterion function of the LQR in shown as below. Using GNU Octave, the LQR controller was designed with weighting matrices

$$J = \int_0^\infty \left(x^T Q\ x + {\theta_a}^T R\ \theta_a\right) dt, \tag{7}$$

$$Q = \mathrm{diag}(1, 1, 1, 1, 1, 1),$$

$$R = \mathrm{diag}(10, 10).$$

The state feedback controller is $\theta_a = -Kx$, where

$$K = -\begin{bmatrix} 51.99 & 30.38 & 13.86 & 18.90 & 11.50 & 5.458 \\ 38.10 & 22.30 & 10.11 & 13.84 & 8.536 & 3.853 \end{bmatrix} \tag{8}$$

Table 1: Nodes of function

Function	Number of arg.	Description
$+$	2	arg.1 + arg.2
$-$	2	arg.1 − arg.2
$*$	2	arg.1 × arg.2
tanh	1	tanh(arg.1)

Table 2: Parameters of GP

Parameter	Value
Number of generation G	200
Population N	200
Mutation rate α	0.60
Crossover rate β	0.80

Table 3: Parameters of the acrobot

Parameter	Value	Parameter	Value
m_1 [kg]	0.5	l_{c1} [m]	0.25
m_2 [kg]	0.5	l_{c2} [m]	0.25
m_3 [kg]	1.0	l_{c3} [m]	0.50
l_1 [m]	0.5	I_1 [kgm^2]	0.010
l_2 [m]	0.5	I_2 [kgm^2]	0.010
l_3 [m]	1.0	I_3 [kgm^2]	0.083

5 Result and Discussion

Figure 3 shows the fitness value of the most excellent individual at each generation. As the generation proceeds, the fitness value decrease gradually.

Figure 4 shows successful simulation results for swing-up and balancing. θ_1, θ_2 and θ_3 converged on the balance point in about 5.0[s]. $\dot{\theta}_1$, $\dot{\theta}_2$, $\dot{\theta}_3$, θ_{a2} and θ_{a3} are

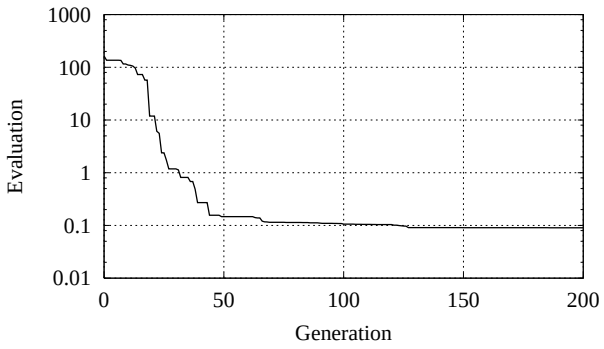

Fig. 3: Evaluation at each generation

similar results. The switching time from swing-up control to balance control is 2.88[s]. The acrobot reaches the balancing point quickly.

We obtained the optimum feedback input function using GP. The tree structure of θ_{a2} has 167 nodes and its depth is 45. The tree structure of θ_{a3} has 157 nodes and its depth is 36. The computer equipment in this study was Windows XP, Intel(R) Pentium(R) Dual CPU E2180 2.00GHz, and Java programming language was used. The computation time for the simulation was about 15 minutes per trial. The input functions are very complex because the depth and the number of nodes are large. Therefore, it is clear that performing swing-up control for a 3-DOF acrobot is very difficult.

6 Conclusion

We have proposed a control method for the 3-DOF acrobot.We modeled the acrobot considering the dynamics of the DC servomotor, which actuated the second and the third joints. GP searchers for a feedback input function, and we obtained the optimum control input using GP, but the input function is very complex. In the simulation results, the acrobot could swing-up to the desired position, and the proposed method could control the 3-DOF acrobot effectively. Further studies are needed in order to reduce the number of nodes, and real experiment should be carried out to verify the control method.

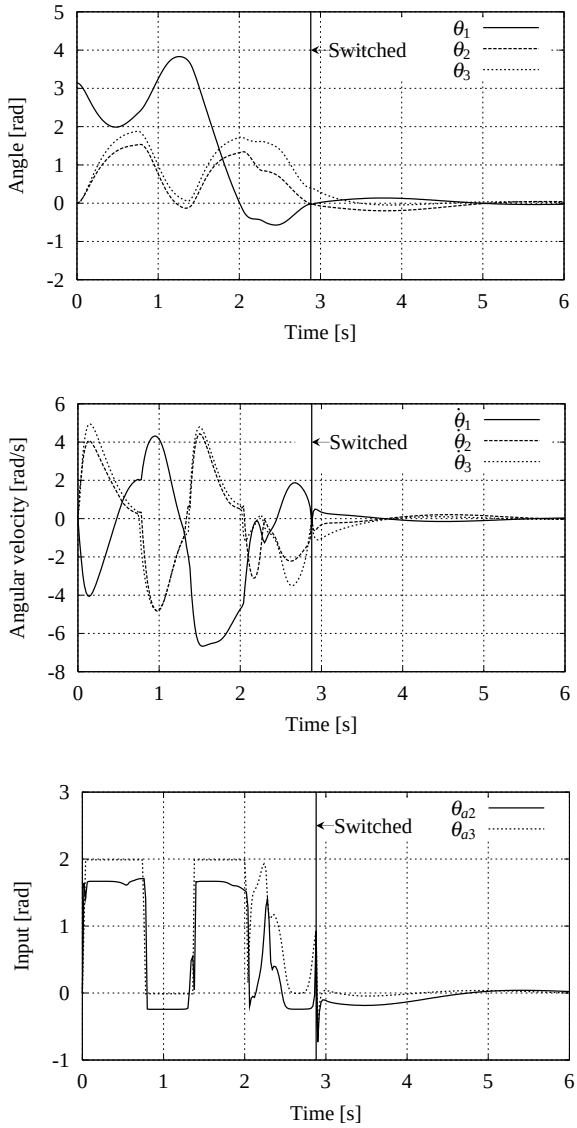

Fig. 4: Simulation results

References

[1] M. W. Spong (1995), The Swing Up Control Problem For The Acrobot. IEEE Control System Magazine, Vol.15, No.2, pp.49–55

[2] K. Kawada, M. Obika and S. Fujisawa et al (2005), Creating Swing-Up Patterns of a Acrobot Using Evolutionary Computation. Proc. 2005 IEEE Int. Symposium on Computational Intelligence in Robotics and Automation, pp.261–266

[3] I. Matsumoto, K. Yoshida (2004), Swing–up and Stabilizing Control of the Acrobot. Transactions of the Institute of Systems, Control and Information Engineers, Vol.17, No.1, pp.17–25

[4] T.K. Nam, Y. Fukuhara and T, Mita et al (2002), Swing-up Control and Avoiding Singular Problem of an Acrobot System. Proc. SICE Ann. Conf. 2002 in Osaka (SICE2002), pp.2990–2995

[5] J. Koza (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press

[6] T. Ogawa, N. Oshiro and H. Kinjo (2008), Backward Movement Control with Two-Trailer Truck System Using Genetic Programming. Proc. of the 13th Int. Symposium on Artificial Life and Robotics, pp.597–600