Proceedings of the First Asia International Symposium on Mechatronics(AISM 2004)
September 27~30, 2004, Xi'an, China

Motion Generation of Humanoid Robot based on Polynomials Generated by Recurrent Neural Network

Riadh Zaier
Fujitsu Laboratories Limited, Japan
email: zaier@stars.flab.fujitsu.co.jp
Fumio Nagashima
Fujitsu Laboratories Limited, Japan

Abstract: Humanoid robots are expected to have variety of motions that enables good interaction with real human environment. Making a program for generating several stable motions using the standard programming language such as C is not only time consuming but also hard to understand and tune. For this, a suitable recurrent neural network language (RNN) inspired from neurobiology has been developed. In this paper, a simple method of motion generation based on polynomials generated by RNN is presented. All motions are generated using a basic RNN circuit of a first order polynomial. Using this method it is easy to generate a complex motion of humanoid robot. Furthermore, Feedback controllers can be easily inserted in the RNN circuit of a motion at any desired timing. Both rhythmic and non-rhythmic motion can be generated based on the same strategy. The effectiveness of the proposed method is verified by experimental results.

Keywords: Recurrent Neural Network, Polynomials, Central Pattern Generator.

1 Introduction

Recently, humanoid robots are getting more popularity and attracted many researchers who have developed a variety of solutions dealing with its motion and behavior. Most previous works were focused on the generation of motion using Newtonian dynamics equations, which require a lot of computational effort and a long development time as well. In our research therefore, we focused on the biological based models using the Recurrent Neural Network (RNN) theory. For this, an RNN language has been developed[1,2,3,4], suitable for the programmers to reflect the biological process in generating robot motion. In contrast to the mathematical notations [6, 7], this proposed language would be expected to express the learning process of a motion, just by changing connections, and their weights for a given RNN circuit. The basic idea behind this language is taken from neurobiology, where the neural network is assumed to solve problems using 4 types of operations namely, summation, multiplication by a constant, introduction of a time delay constant, and switching. Accordingly, the proposed RNN language is limited to these four operations. Using this language, several RNN circuits are easily designed termed Central Pattern Generators (CPG).

In this paper, generation of both rhythmic and non-rhythmic motions is presented. The procedure adopted to generate a non-rhythmic motion is based on a simple RNN circuit that generates a first order polynomial. For example, a kicking motion consists of a set of basic circuits. All circuits are linked to only one neuron. Once the motion is designed, each joint motion is smoothened through a neuron to assure the continuity of the angular velocity. The designed motion has only two parameters; the kicking speed and angle. Then motion is tuned so that it remains stable for a wide range of kicking speed. As for the rhythmic motion, for instance the walking motion, it is designed for one motion cycle following the same steps of generating a non-rhythmic motion, previously mentioned. To make the motion a cyclic one, a time shift is considered from an upper level layer so that the motion will be regenerated consecutively.

Moreover, compliance and feedback control are also realized for the stability and smoothness of the robot motion, where compliance control is activated at the landing phase, while the feedback controller is enabled during the lifting phase. The feedback controllers are designed for a second order plant model. The controllers input signals are from both gyro and sole sensors. A low pass filters are introduced into the control loop. In addition, gait control is also considered, where motion switching is controlled by an upper level layer that has input signals from the gyro sensors.

As a result, using our proposed method, it is easy to implement both feedback and compliance controllers as well as ensuring any switching condition. An experiment is conducted by having the humanoid robot HOAP-2[5] of Fujitsu walk stably on an obstacle of 12mm height. The robot can also walks with a large step.

2 Proposed RNN Language[1]

2.1 Mathematical model of a neuron

The mathematical model of a single neuron in the proposed RNN language is given by eq.1. Figure 1 shows its graphical representation.

$$\varepsilon_i \frac{dy_i}{dt} + y_i = \sum_j c_{ij} y_j \qquad (1)$$

where y_i is the output of neuron i, ε_i is a time delay constant, and y_j is the output of neuron j that represents an input of neuron i through a weighting factor c_{ij}. Notice that ε_i can be interpreted as the rise time constant of a step input [6].

2.2 Basic elements

In the proposed RNN language, four basic elements are defined and summarized in Fig.2, which are neuron, wire, cable-C, and cable-W. Their definitions are stated below

2.2.1 Neuron

To create a neuron with a time delay constant ε and initial value V_0, we can proceed as follows

var a(ε) = V_0;

2.2.2 Wire

The function of a wire is to connect 2 neurons. For example, neuron "a" is connected to neuron "b" by a wire of weighting factor C as follows

a := C * b;

2.2.3 Cable-W

Cable-W consists of changing wire's weighting factor. It is created as follows,

var C(ε) = V_0;

var a(ε_1);

var b(ε_1);

a := C * b;

2.2.4 Cable-N

The function of cable-W is to change other neuron's time delay constant.

var eps(ε) = V_0;

var a(eps) = W_0;

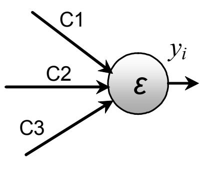

Fig. 1 Representation of a single neuron

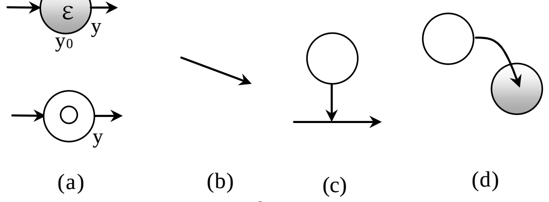

Fig. 2 Basic elements of the proposed RNN language (a): neuron, (b): wire, (c): cable-W, and (d): cable-N

2.3 Permitted Operations

Only 4 types of operations, as shown in Fig.3, are permitted in the proposed RNN language, namely summation, multiplication by a constant, introduction of a time delay, and switching. Notice that a threshold can be considered as a switcher.
The symbol "#" in fig 3 is used for comment out.
The graphical notations of switchers are shown in fig. 5.

Remark

Both cable-W and cable-N involve a nonlinear operation. This does not contradict the definition of the proposed language, since cables belong to an upper level layer with a different time scale.

2.4 RNN Circuits

RNN circuit is a set of neurons connected to each other by wires. A circuit presents an output and input connections. For example, Fig. 4 shows how to create a circuit "Joint".

```
var v(0.0);     }Dead              }
var w(0.0);          neurons       }    Create four
var y1(ε1);                        }    neurons
var y2(ε2);

y := C1 * y1;                #  Multiply by a constant
y := C1 * y1 + C2 * y2;      #  Summation
v := if(0.2 < y1) 1.0 * y;   #  Switcher
w := 1.0 * (0.2 <) y;        #  Threshold
```

Fig. 3 Permitted operation of the proposed language

```
input in[2];
output out[1];
circuit Joint {var p; …}
out[0] := 1.0 * ::Joint::p + 1.0 * in[0] +1.0 * in[1];
```

Fig. 4 Syntax of creating RNN circuit

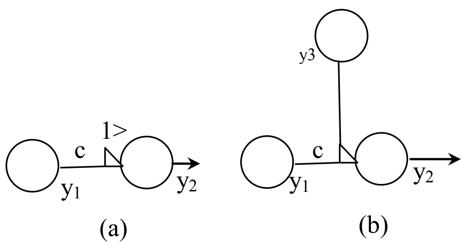

Fig. 5 Switching in RNN language (a): threshold, (b): switcher

3 Proposed Motion Generation Method

The proposed method consists of designing CPGs of several motions using RNN language and based on a basic circuit generating first order polynomials.

3.1 Basic RNN circuit

The basic RNN circuit generating first order polynomial is shown in fig. 5. It consists of 2 neurons *u* and *n*, and sub-circuit P_1. Neuron *u* generates a unit constant that will be used in the whole RNN circuit, while neuron *n* generates a first order polynomial by inserting the output of *u* into neuron *n* through a wire of weight C_s. All neurons of the sub-circuit P_1 have null time delay constant, in other words they are just connections and digital switchers.

The output *y* of the basic RNN circuit is shown in fig.7. It consists of one degree rotation starting at time t_s with a constant angular velocity $k=c_s.m$. The parameter s*w* is used to switch ON/OFF a motion.

By fixing the value of ε equals to 1, the output y_1 can be expressed as follows,

$$v_1 = c_s t - t_s \qquad (2)$$

Where t_s is a parameter expressed in a number of time cycle of a given reference motion *T*, and c_s is a constant s.t., $c_s = 1/T$.

3.2 Motion Generation using basic RNN circuit

Motion can be regarded as a sequence of basic motions generated by polynomials of low order degree. For example, to pick up an object, human will not enjoy generating a complex motion rather than a sequence of simple and smooth motions. Therefore, as shown in fig.8, our idea consists of generating a sequence of motions using the basic RNN circuit of fig. 6.

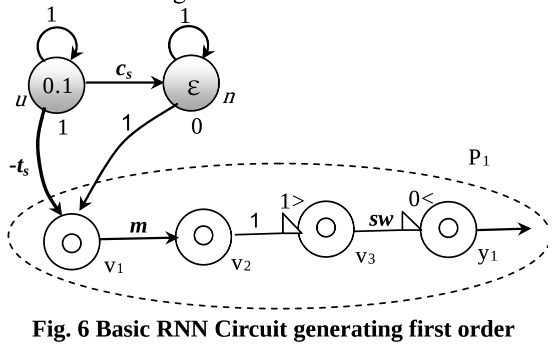

Fig. 6 Basic RNN Circuit generating first order polynomial

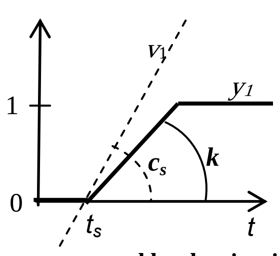

Fig. 7 Output generated by the circuit of fig.6

Now writing the expression of the motion output of joint j_i as follows,

$$j_i = \sum_{i=0}^{n} c_i y_i \qquad (3)$$
$$\text{with } y_i = k_i (t - t_i)\delta_i \le 1$$

Where $k_i \ge 0$ and $\delta_i = 1$ *if* $t \ge t_i$ *else* $\delta_i = 0$.

Notice that with regards to fig6, $k_i = c_s m_i$

Since the parameter m_i determines the velocity of the motion from time t_i to t_{i+1}, it should be designed carefully taking into account the dynamics of the motion. Trial and error method can be very effective way to find the appropriate value of this parameter.

As it can be seen from fig. 8, the design of the motion using the proposed method is straightforward and it can be generated based on sensory feedback system.

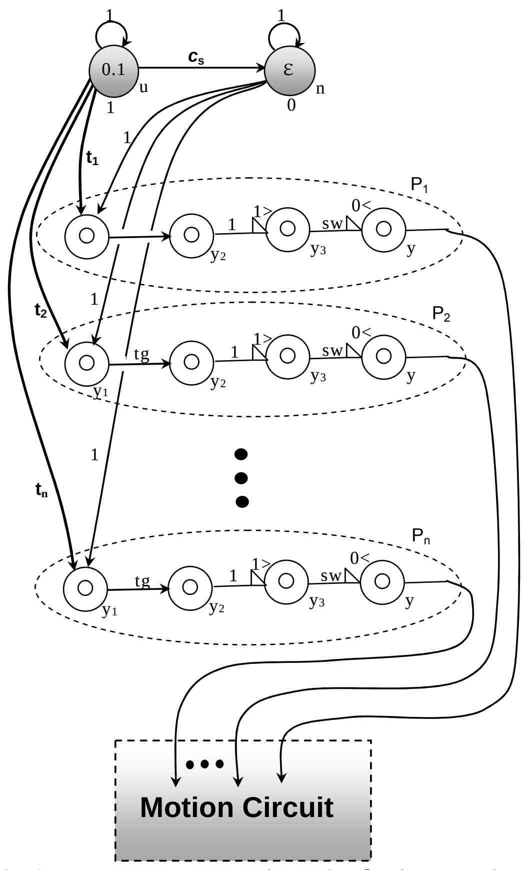

Fig. 8 How to generate motion using basic RNN circuit

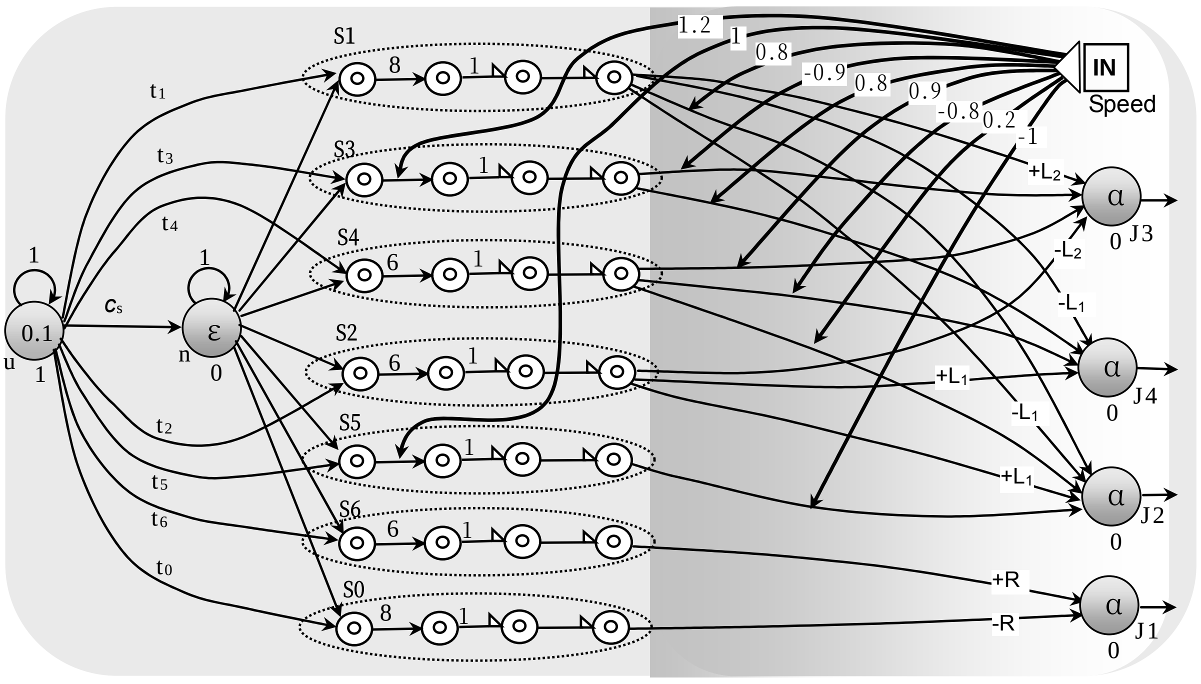

Fig. 9 Kick motion generation using basic RNN circuit

3.3 Generation of Kick Motion

As shown in fig.9, the kick motion consists of a sequence of 7 basic motions at different timing t_i. Only one parameter named "Speed" that this motion depends on, which may vary within a defined interval. By using cable-W, this parameter "Speed" changes the wires' weights so that the robot motion remains stable at different kicking speed. To determine this parameter, trial and error method was adopted. It can be also determined using recursive method with sensory feedback and a defined evaluation function. Moreover, to ensure the continuity of the angular velocity of the motion, each joint's output is passed through a neuron of a delay α. The sub-circuits S0 and S6 are used to generate the rolling motion of amplitude "R". Notice that if the kicking speed parameter is null, the motion of the robot will be reduced to a rolling and lifting only. The parameter L_1 and L_2 correspond to the lifting amplitude of the heap-ankle and knee, respectively. Also it should be noticed that the same sub-circuits *Si* can be used for several motions, such as walking, standing etc...

Remark: For the readability of fig. 9, the kicking angle parameter was ignored.

4 Motion Stabilizers

The state space configuration of a system is converted to RNN structure based on the definition of the neuron model given by eq.1. Let's consider the state space of a single input single output (SISO) controlled system of order n, which has the following expression.

$$\frac{dx}{dt} = \boldsymbol{A}x + \boldsymbol{B}u \qquad (4)$$

$$y = \boldsymbol{C}x \qquad (5)$$

The index form of eq. 4 can be written as follows.

$$\frac{dx_i}{dt} = \sum_{j=1}^{n} a_{ij}x_j + b_i u \qquad (6)$$

That can be re-written as

$$\frac{\delta_i}{a_{ii}}\frac{dx_i}{dt} + x_i = (1+\delta_i)x_i + \sum_{j=2}^{n}\frac{\delta_i a_{ij}}{a_{ii}}x_j + \frac{\delta_i b_i}{a_{ii}}u \qquad (7)$$

where $\delta_i = sign(a_{ii})$.

As a result, the right side of eq.7 represents the value of the neuron output x_i, and the left side parts are the inputs to the neuron.

Figure 10 shows the RNN circuit of the second order SISO system (n=2), with the following parameters

$$k = 1+\delta\ , \varepsilon_i = \frac{\delta_i}{a_{ii}},\ \ d_i = \frac{\delta_i bi}{a_{ii}},\ \ f_i = \frac{\delta_i a_{ij}}{a_{ii}}; i \neq j$$

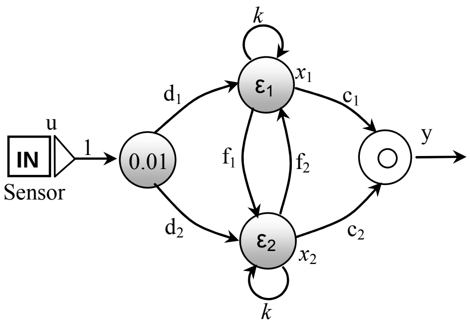

Fig. 10 RNN structure of state model of eqs.4, 5

4.1 Feedback and compliance controllers

State space controllers are designed for a second order plant model of an inverse pendulum shown by fig.10. The initial value of the controllers' parameters were obtained by simulation then tuned during experimentation. For instance, to stabilize the pitching motion of the robot, two controllers can be used. One controller has angular velocity input from the gyro sensor and the second one has input from the sole sensor. As for the compliance controllers they are also state space controller but with different purpose than that of the feedback controller. By changing the gain of the controller's input, the stiffness changes accordingly. Therefore, this parameter can be controlled according to the type of motion. For example, it can be adjusted for a soft landing of the robot leg during walking, based on the sole sensor's input. The merits of having an RNN implementation of the controller are, 1) easy to tune, 2) easy to switch at the desirable timing, and 3) the whole motion circuit remains simple.

5 Motion Control System

The real-time control algorithms are implemented in real-time threads running at RT-Linux kernel space. Linux user space applications are in charge of network interface, user command parsing, message distribution, and data server management. The real-time threads communicate with user space applications for data transfer by using high speed FIFOs, while commands are sent to the kernel by slow speed FIFO. Kernel mode shared memory (SM) is constructed for communication between real-time threads. Fig.11 shows the diagram of the system software, where application program interface was made to connect with a client program that can run in either Windows or Linux.
The control algorithms were implemented in real-time control thread. After the construction of kernel RNN circuit, the control thread was scheduled for periodical execution. The control period is 1ms. The RNN interface with the robot was realized by real-time USB driver thread.
Figure 12 shows the RNN circuit that generates a set of motions. An interface layer (RNN_IO) switches the appropriate motion upon user's request.

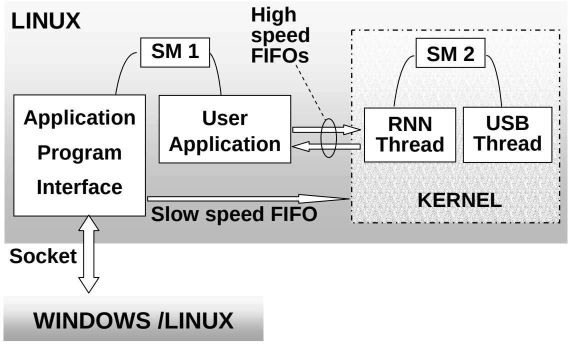

Fig. 11 Structure of motion control system

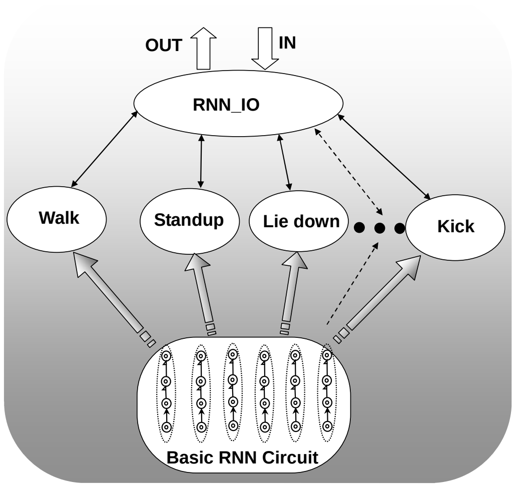

Fig. 12 RNN circuit for various motions

5.1 Walking motion

Using FUJITSU Robot HOAP-2, the walking motion with compliance controller generated by the proposed method enabled the robot to walk with large step (20cm) and fast speed as well (11.8cm/s). The outputs of the CPG circuit to the knee joint of the right leg (pitching and the rolling) are shown in fig. 13. The procedure of making this rhythmic motion is the same as that of making kick motion with the addition of time shift by one walking cycle. Notice that this motion is generated using 11 basic CPG circuits of fig. 6.
Inserting the compliance controller in the RNN motion circuit yields the outputs of figs. 14, 15 and 16, where the sensory feedback considerably has changed the pitching motion of the heap, knee, and ankle. As a result, the robot can walk stably and get adapted with the terrain. Using this combination of CPG and sensory feedback the robot becomes able to walk not only on horizontal terrain but also on a slope, and stairs with different step height.

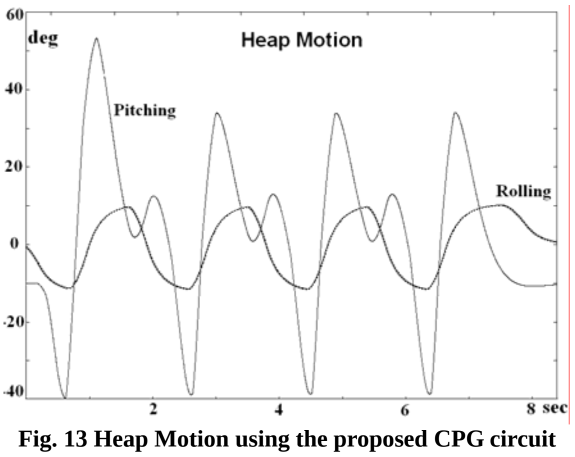

Fig. 13 Heap Motion using the proposed CPG circuit

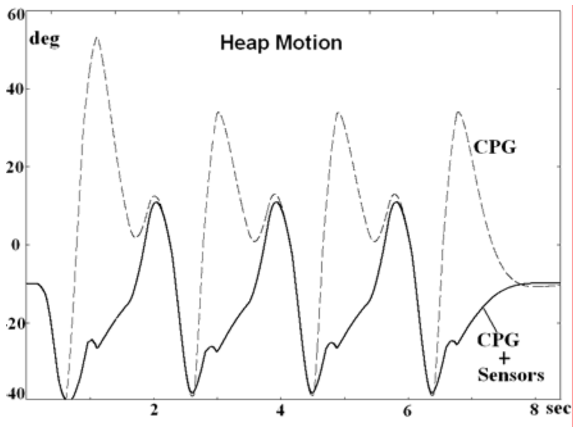

Fig. 14 Pitching motion of the heap joint

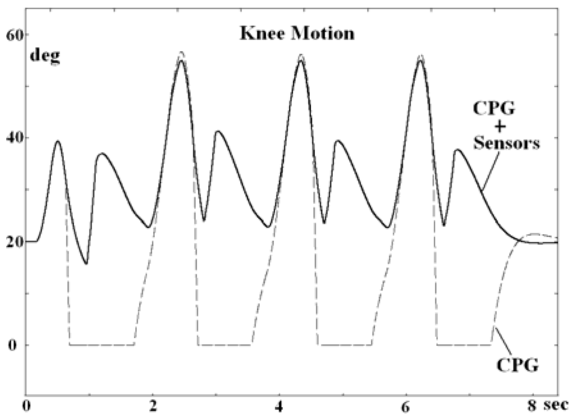

Fig. 15 Pitching motion of the knee joint

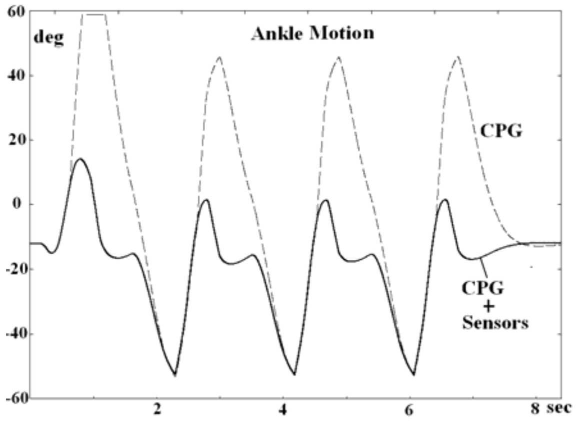

Fig. 16 Pitching motion of the ankle joint

Moreover, the gain of the compliance controller, the robot can walk on an obstacle of 12mm height. By controlling the timing t_i of the basic RNN circuits in eq.3 from an upper level layer, the gait can be controlled so that during motion.

6 Conclusion

In this paper, a simple method for generation of both rhythmic and non-rhythmic motions was proposed. The procedure adopted to generate a non-rhythmic motion is based on a simple RNN circuit that generates a first order polynomial. To illustrate this idea a kicking motion was design based on the proposed method. The steps of making motion were straightforward and the tuning can be easily achieved. Furthermore, the walking motion was also generated based on the same procedure of the kick motion with time shift by one walking cycle.

On the other hand, to make a generated walking motion adaptive, compliance and feedback control were realized so that the robot may walk on horizontal terrain, slope, and stairs with different heights. As a result, using our proposed method, it was easy to implement both feedback and compliance controllers as well as ensuring any switching condition. An experiment was conducted by having FUJITSU humanoid robot HOAP-2[5] walks stably on an obstacle of 12mm height. The robot can also walks with a large step.

References

1. Zaier, R, Nagashima, N., Recurrent Neural Network Language for Robot Learning, *The 20th Annual Conference of the Robotics Society of Japan,*2002
2. Nagashima, F., A Motion Learning for a Robot using CPG/NP, *The 20th conf. of Robotics Society of Japan*, 2002
3. Jiang, S., Nagashima, F., Biologically Inspired Spinal locomotion Controller for Humanoid Robot, *19th Annual Conference of the Robotics Society of Japan*, 517-518, 2001
4. Jiang, S., Nagashima, F., Neural LocomotionController Design and Implementation for Humanoid Robot HOAP-1, *The 20th Annual Conference of the Robotics Society of Japan*, 2003
5. Murase, Y., Yasukawa, Y., Sakai, K., Ueki, M., Design of Compact Humanoid Robot as a Platform, *19th Annual Conference of the Robotics Society of Japan*, 789-790, 2001
6. Matsuoka, K., Mechanisms of Frequency and Pattern Control in the Neural Rhythm Generators, *Biol. Cybern,* 56, 345-353, 1987
7. Taga, G., A model of the neuro-musculo-skeletal system for human locomotion, I. Emergence of basic gait, *Boil. Cybern*, 73, 97-111, 1995