

A Motion Learning Method using CPG/NP

Fumio Nagashima¹

¹Fujitsu Laboratories, 10-1, Morinosato-Wakamiya, Atsugi 243-0197, Japan shiro@stars.flab.fujitsu.co.jp

keywords: Central Pattern Generator CPG, Numerical Perturbation NP
Recurrent Neural Network RNN, Motion Generation

Abstract

I propose a motion learning method for a robot using a Central Pattern Generator with Numerical Perturbation (CPG/NP). The Central Pattern Generator (CPG) is modeled as a sub circuit of Recurrent Neural Network (RNN). Numerical Perturbation (NP) determines coefficients for each perturbed order RNN, step by step. System inputs are a motion outline, the direction of perturbation and some pieces of advice. The experiments using HOAP-1 indicate that this method can generate a variety of motions, however, the calculation time dramatically decreases.

1. Introduction

Recently, there have been several studies about human friendly robots or humanoid robots. I can imagine a scene where a humanoid robot is working along side a human and performing several tasks [1]. And it is possible they will be able to perform a great variety of tasks. To create a software system for such a state requires a lot of software engineers and time, because the engineers must analyze the tasks and create algorithms for their particular target tasks. Then we must consider the efficiency of creating a software system. One way to keep the cost down for such software is to create middle-ware for robot software. Some researchers stand by this viewpoint. This is not only the way for future robot software, but also a learning is.

The back propagation (BP) of the Layered Neural Network (LNN) and Genetic Algorithm (GA) are known as the learning system. BP is a powerful method for finding a solution for an LNN, while a GA is useful for any problems with a large parameter space and has a capability to find a global minimum. However, the BP is only for use with an LNN and a GA requires a lot of time.

I assume that a motion, such as walking, is governed by a non-linear equation system. Some researchers try to solve these equations using a numerical calculation technique, while others try to do so using their own experiments.

In this paper, I discuss a robot motion learning system using a Central Pattern Generator (CPG) [2, 3] with Numerical Perturbation (NP). The CPG is defined by Recurrent Neural Network (RNN) sub circuits and the NP determines the coefficients of the RNN circuit, step by step. Fig.1 shows the strategy for this method and an outline of the system using this method is shown in Fig.2.

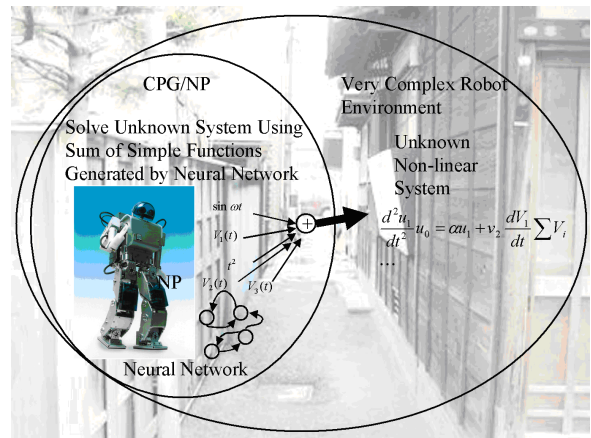


Figure 1: Strategy of the Proposed Method

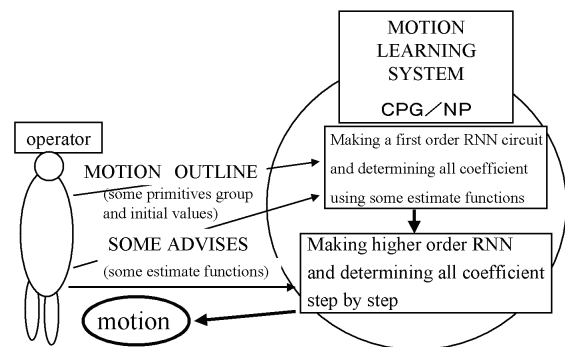


Figure 2: Outline of the Proposed System

2. CPG model

2.1. RNN model

I use 4 assumptions to make a RNN model. The most important assumption is “delay”. To create a pattern, we need something time related in a neural network. There are several methods to consider in regards to time. For example, the differential relationship, the difference relationship between the state which has different time. Or we can convert a space relationship to time relationship. I chose a simple analog delay for such a purpose, as follows:

$$\varepsilon_i \frac{dV_i}{dt} + V_i = input. \quad (1)$$

Where V_i is neuron value, ε_i is delay, t is time, $input$ is an input to the neuron and i is neuron ID number. Fig.3 shows the non-delay neuron model. It is permitted to have ε_i equal to zero. I call such a neuron “dead neuron”. In this case, any connections of neurons cannot generate a variety of outputs. Fig.4 shows the work of the simple analog delay. The equation of this is

$$\varepsilon_1 \frac{dV_1}{dt} + V_1 = V_0. \quad (2)$$

Next two assumption are “multiplication by constant” and “summation”, as follows:

$$\varepsilon_i \frac{dV_i}{dt} + V_i = \sum_{j=1}^n C_{ij} V_j. \quad (3)$$

Using these assumptions, a RNN circuit can be created as shown in Fig.5. The equations of this circuits are obtained using the balance of each neuron as

$$\begin{cases} \varepsilon_1 \frac{dV_1}{dt} + V_1 = C_{12} V_2 + V_0, \\ \varepsilon_2 \frac{dV_2}{dt} + V_2 = C_{21} V_1. \end{cases} \quad (4)$$

These equations can be reduced to the relationship between V_0 and V_1 ,

$$\varepsilon_1 \varepsilon_2 \frac{d^2 V_1}{dt^2} + (\varepsilon_1 + \varepsilon_2) \frac{dV_1}{dt} + (1 - C_{12} C_{21}) V_1 = V_0. \quad (5)$$

This shows the simplest RNN circuit that can generate some useful patterns for motion. This circuit has four types of output which depend on its coefficients, the curve increasing rapidly, being almost linear except origin, the curve increasing slowly and decreasing vibration as shown in Fig.6. There are several kinds of circuits that can generate special functions.

The last assumption is “switch”. I assume that a neuron can control the connection weight between neurons in a lower layer as

$$\varepsilon_i^k \frac{dV_i^k}{dt} + V_i^k = \sum_{j=1}^n C_{ij}^k V_j^k, \quad (6)$$

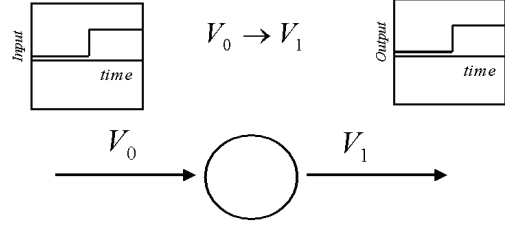


Figure 3: Information without Delay

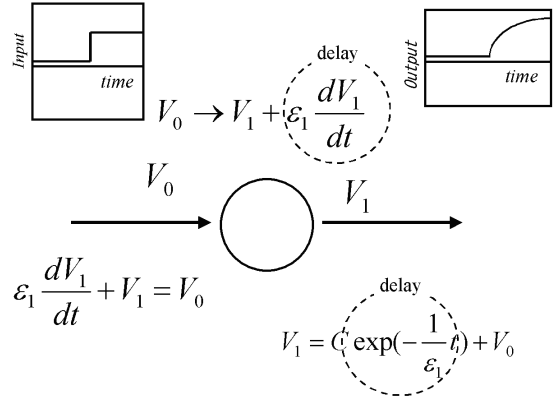


Figure 4: Information with Delay

$$C_{ij}^k = \begin{cases} const \\ or \\ V_j^l (k < l). \end{cases} \quad (7)$$

Because it is important to keep the network understandable at the beginning of research, two way communication between layers are not permitted. It is not necessary to consider the non-linear effect to generate a motion in a non-linear environments as shown in the section 3. As well, some threshold is used to realize digital effect, e.g., a play of joint.

The proposed RNN circuit can generate useful patterns for control. See appendix A for a some examples. An RNN based on 4 assumptions can create a variety of equations for control.

2.2. CPG model

Some RNN circuits can create special functions such as sine and polynomial functions as shown in Table 1. And they also can create the solution of variable coefficient equations. These RNN sub circuits can generate a group of functions. I call these RNN sub circuits “CPG” and the group “CPG function group”.

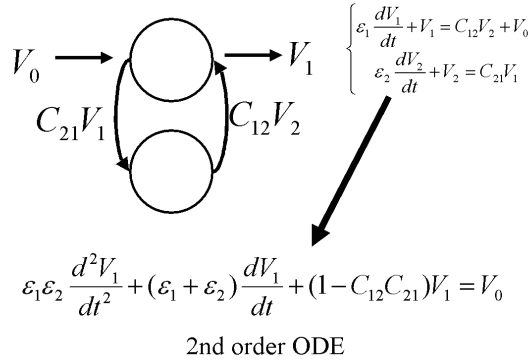


Figure 5: Simple RNN Circuit

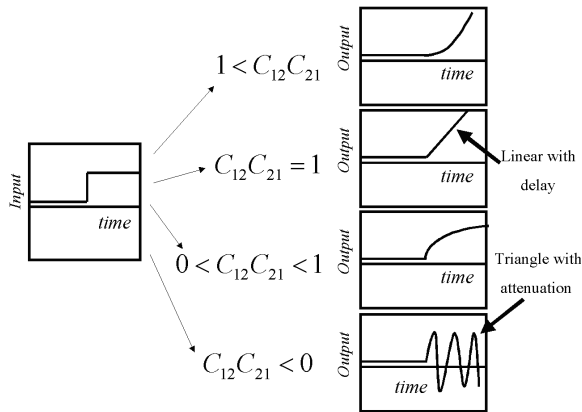


Figure 6: Output of Simple RNN Circuit

Table 1: Typical CPG examples

function	RNN circuit	note
$\sin \omega t,$ $\cos \omega t$		$\omega = C/\varepsilon$
$\sin \omega t,$ $\cos \omega t$		$C = \frac{12 \sqrt[3]{-108 + 12\sqrt{93}}}{(12 - (-108 + 12\sqrt{93})^{2/3})}$
polynomial		$C_0 + C_1 t + C_2 t^2$

To make the software for RNN, if I use the matrix calculation for the simulation and execution of RNN, the system needs a lot of memory space and the empty calculations. To avoid such a losses, I design the language for describing RNN. An input/output terminal such as sensor/motor can be thought as one of the neurons. Examples of the typical CPGs written in this language are shown in Table 2 Zaier,Nagashima [4] use this language for showing how it works.

3. NP method


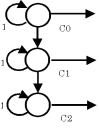
3.1. Perturbation method

The perturbation method is well known method for getting approximate solutions of non-linear equations system. The perturbation method is employed in astronomy, fluid dynamics and by other theoretical researchers, to obtain approximate solutions to non-linear equation systems. They use polynomials or eigen function series which are suitable for their own equation system to get the approximate equations and solutions. In some cases, they use Fourier expansion in perturbation method.

3.2. NP method

I use an idea of perturbation method for obtaining motion. I assume the motion is a solution of non-linear equation system. If I get the complete equation system for a motion, the

Table 2: RNN language Examples

RNN circuit	RNN language
	<pre> circuit sin { var y1(0.1) = 0.0; var y2(0.1) = 1.0; y1 := 1.0 * y1 + 1.0 * y2; y2 := -1.0 * y1 + 1.0 * y2; } </pre>
	<pre> circuit pol { var y1(0.1) = 1.0; var y2(0.1) = 0.0; var y3(0.1) = 0.0; y1 := 1.0 * y1; y2 := 1.0 * y2 + 1.0 * y1; y3 := 1.0 * y3 + 1.0 * y2; } </pre>

solution can be derived by substituting a series of eigen functions into that equation system and solving each order equations. However, it is hard to get the complete equation system for motion, because the environment is very complex and changes so often. Then I use the idea of perturbation method numerically without constructing an equation system. I assume the solution is expressed by

$$y = \delta_0 y_0 + \delta_1 y_1 + \delta_2 y_2 + \delta_3 y_3 + \dots, \quad (8)$$

where y is the output of RNN, y_i is a function generated by CPG, δ_i is a coefficient of order i . First of all, I determine the first approximate coefficient δ_0 with trial and error using a lowest order of eigen functions y_0 , e.g., $\sin \omega t$.

Next, I determine the higher order eigen functions coefficients $\delta_1, \delta_2, \delta_3, \dots$ also by trial and error. Until I can obtain the motion which can satisfy the intended purpose, I keeps increasing to a higher order.

The strategy is simple. I use a numerical sum of linear equations' solutions generated by CPGs to solve the unknown non-linear motion pattern equation system. Fig.7 shows the image of simple growing process, The weights of each connection are determined step by step from lowest order.

4. Learning system outline

4.1. CPG model and NP method

The CPG can generate several type of functions. Some functions are orthogonal to each other. Some functions are useful to generate motion patterns.

The NP method can narrow down the number of coefficients that must be determined simultaneously, making it easier to determine coefficients.

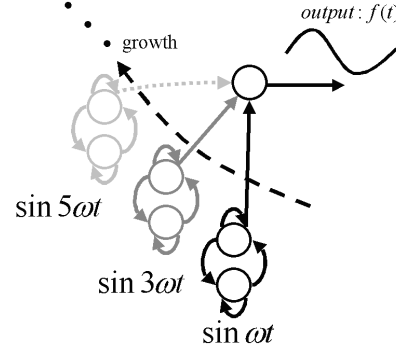


Figure 7: Example of NP method

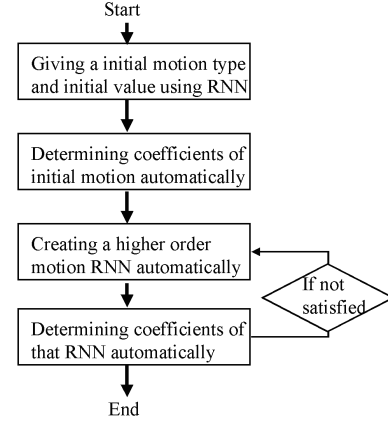


Figure 8: Outline Algorithm of the Proposed Method

4.2. Advises

A learning system needs an estimate function to determine the RNN coefficient. I use the sum of quadratic value, each value being an evaluation function, such as a spending energy, stability of upper body and so on. And I call one of them "advice". The learning process is equal to the process to find a minimum value of the sum of all the advice.

4.3. Learning system outline

Fig.8 shows an abstract algorithm for learning. System input are some pieces of advice and initial motion, and output is RNN circuit for motion.

5. Experiments using HOAP-1

HOAP-1, a humanoid robot for research by Fujitsu Automation Ltd., was used for examining the proposed method. It is 20 DOF robot and has an angular velocities sensor, accelerations sensor in its chest, 4 pressure sensors on the corners of its foot and 2 cameras on its head. I use only triangular func-

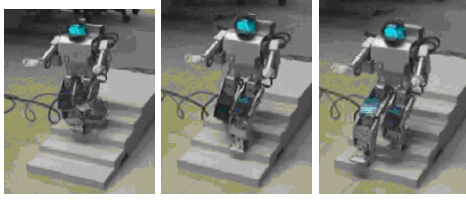


Figure 9: Experiment using HOAP-1

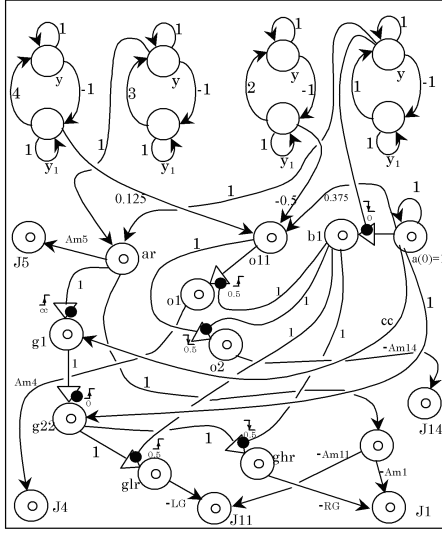


Figure 10: A RNN Circuit Sample for HOAP-1 Experiment

tions and polynomials as RNN generated patterns. All data from sensors are smoothed using a neuron delay function. The output from the angular velocities sensor is converted to upper body angles using RNN circuit. The foot sensors are used for switching to right or left lifting motion. See reference [5] for more detail.

As a result, a 2nd order solution is sufficient for walking on a flat floor, though it is a little bit unstable. I call this motion “baby walking”. A 4th order solution is stable. It can almost do static walking. A 6th order solution can go up/down stairs. I tried testing using the lower order solution to go up/down stairs but it always failed. The variation of motion increases as the NP order increases. Fig.9 shows the snapshot of experiment. Fig.10 shows the part of RNN for this experiment. See reference [6] for the movies of these motions.

6. Discussion

6.1. Comparison with CPG/GA

Because the CPG/GA method determines all coefficient simultaneously, they are related with each other. If I want to determine the new coefficients or just want to modify motions a little, the calculation requires a lot of time. In this sub-section, I will discuss the all the search cases of solutions. Assuming the resolution of neuron value and that con-

nection weight n , the sum of neurons and connections m , the total trial number becomes n^m with CPG/GA, $jn^{m/j}$ with CPG/NP. j is the number of orders. For example, in the case of $n = 16, m = 50, j = 5$, I get 1.15×10^{73} for CPG/GA, 1.4×10^{16} for CPG/NP. I tried to find a pattern of walking for HOAP-1 using CPG/GA in experiments but I could not find it until now.

Furthermore, the estimate function can be changed at the start of each perturbation order as the number of neurons changes while learning.

6.2. Comparison with LNN/BP

LNN/BP and CPG/NP are quite different methods. I think LNN/BP cannot create a reasonable motion and that we must use the combination of these two methods.

6.3. Comparison with the Newtonian-dynamics based method

The ZMP method and other Newtonian-dynamics based method need an inverse solving technique for a non-linear dynamics equation system of motion. The motion capture based method also needs a complex mathematical problem for ZMP compensation. Recently, an approximate method or some practical method have been established. We need to discuss and compare about the merits and demerits of the proposed method and Newtonian-dynamics based methods.

7. Conclusion

In this paper, I propose a robot learning method using CPG/NP. CPG is defined by an RNN sub circuit and the NP method determines coefficients of RNN circuit, step by step. I have shown an outline of learning system and examined this method using HOAP-1. As a result, this method can generate a gait patterns including stairs’ walking for a humanoid robot.

Acknowledgments

I wish to acknowledge the discussion and comments provided by Dr. Susumu Kawakami and Professor Masafumi Yano. And I would like to thank Dr. Jiang and Dr. Zaeir for the assistance with the experiment.

References

- [1] Fumio Nagashima, “A Motion Learning using CPG/NP,” *The 20th Annual Conference of the Robotics Society of Japan* (2003, in Japanese).
- [2] H. Kimura, Y. Fukuoka. “Adaptive dynamic walking of the quadruped on irregular terrain-autonomous adaptation using neural system model,” *Proc. of the 2000 IEEE*, p.436-442 (2000).
- [3] G. Taga. “A model of the neuro-musculo-skeletal system for human locomotion,” *I. Emergence of basic gait. Bio-*

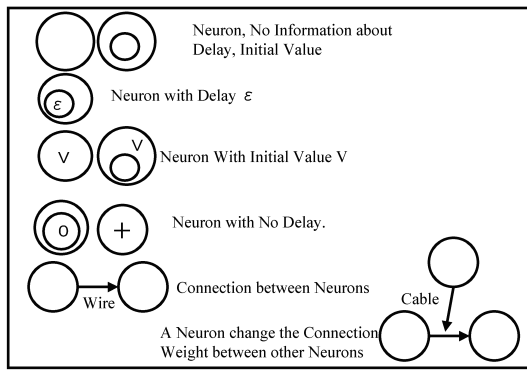


Figure 11: Notation Samples of Neural Network

logical Cybernetics, 73(2), p.97-111 (1995).

- [4] Zaier Riadh, Fumio Nagashima, "Recurrent Neural Network Language for Robot Learning," *The 20th Annual Conference of the Robotics Society of Japan* (2003).
- [5] Jiang Shan, Fumio Nagashima, "Neural Locomotion Controller Design and Implementation for Humanoid Robot HOAP-1," *The 20th Annual Conference of the Robotics Society of Japan* (2003).
- [6] <http://www.automation.fujitsu.com/products/products076.html> (in Japanese)

A RNN circuit

Fig.11 gives a notation samples of the proposed RNN circuit. It consist of neurons, wires and cables. The neuron has delay and initial value, the wire is connection between two neurons, and the neurons can affect to the connection of neurons by cable.

Because the proposed RNN circuit is the mathematical circuit in a sense, it can also describe an ordinary controller such as PD, PID, an integrator and so on. Table 3 shows some typical examples. Using these circuit, it's easy to create a powerful control software including ordinary methods and the proposed method using a simple notation without any expansion. Even if someone wants to create a Newtonian-dynamics simulator for a controller using this circuit, they can do it.

It is important to keep the circuit understandable. Using cable, a user can generate a non-linear equation's solution easily but this cable is for variable coefficient linear ODE. This means the cable is only for inter-layer connection, from the upper layer to the lower layer, that does not have two-way communication. The parametric resonance is a typical RNN example of the inter-layer connection.

Table 3: Some Useful RNN Circuit

RNN Circuit	Note
<p>Low Pass Filter</p>	Simple Low Pass Filter
<p>Integrator</p>	Simple Integrator
<p>PID Controller</p>	Simple PID Controller
<p>Motion Generator and PID Controller</p>	Ordinary Control Method PID
<p>Inner Product</p>	for Sensing or Measurement
<p>Parametric Resonance</p>	$\frac{d^2V}{dt^2} + \omega^2(1 + f(t))V = 0,$ $\omega = C/\epsilon$