The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

# A new method in modeling Central Pattern Generators to control quadruped walking robots

Duc Trong Tran, Ig Moo Koo, Gia Loc Vo, Se-gon Roh, Sangdeok Park, Hyungpil Moon, and Hyouk Ryeol Choi*, *Member, IEEE*

***Abstract*— Aiming to real easy application in several quadruped robot platforms, this paper introduces a new method of modeling central pattern generators (CPG) to control quadruped locomotion. Not only can this new model generate all the primary gaits of quadrupeds stably with limit cycle effect, but it also has the ability of tuning the periodic outputs with arbitrary waveforms. The core idea is to combine strong points of two mathematical tools: Fourier series and Recurrent neural networks. In addition, a new biomimetic controller is also introduced using the proposed CPG model and several reflex modules. Finally, dynamic simulations are performed to validate the efficiency of the proposed controller.**

## I. Introduction

RECENTLY, many previous studies on legged robots locomotion control based on biologically inspired approach have been performed [1]-[5]. As the core of these approaches, Central Pattern Generator (CPG) has become the vital part in any biomimetic controller [1]-[6].

The problem in modeling CPG to control legged robots is usually expressed as searching some autonomous dynamic system of coupled oscillators that generate all the primary gaits of quadrupeds [4][9]-[11]. The general equation for these oscillators is:

$$\dot{x}_i = F(x_i) \tag{1}$$

where $x_i \in \mathbf{R}^n$ is the state variables of oscillator i and the vector field $F : \mathbf{R}^n \to \mathbf{R}^n$ models the dynamics of the whole system. The basic assumption is that the waveforms of $x_i$ are similar but shifted in different phases as shown in table 1 [4][5][9]. The main goal is that the CPG model should maintain these phase relations stably even in the presence of external perturbations. It means that some stable limit cycle should be generated. There have been research works to solve this problem. Matsuoka proposed a self-sustain oscillator [4] that can generate several basic gaits for quadruped robots. This model is used and improved by adding several reflex modules later by Kimura in control of a real robot named Tekken [1]. Another great work is done by Buono et. al. by proposing a network of eight dynamic cells with a special structure that can generate stable gaits for the wide range of quadrupeds [9]-[11]. Similar research in insect locomotion is also introduced by Bailey [5]. Some other work is done using neural networks [7]. However, it is still not clear how these proposed models can be applied, in general, for all the real quadruped robots effectively. The difficult point is that these models just can satisfy the requirements of gait generation but not the leg motion planning problem. Currently, to control quadruped robot locomotion means to handle simultaneously the motion of several manipulators. Furthermore, to control these legs' joints, we need to plan some trajectories in position level, velocity level, or torque level, etc. It means, somehow, that we need well defined patterns. It is, however, difficult for these previous approaches to ensure to have such outputs as predefined waveforms.

TABLE I
PRIMARY GAITS FOR QUADRUPEDS

| | $\varphi_1$ | $\varphi_2$ | $\varphi_3$ | $\varphi_4$ |
|---|---|---|---|---|
| Walk | 0 | T/2 | T/4 | 3T/4 |
| Trot | 0 | T/2 | T/2 | 0 |
| Pace | 0 | T/2 | 0 | T/2 |
| Jump | T/4 | T/4 | 0 | 0 |
| Bound | 0 | 0 | T/2 | T/2 |
| Pronk | 0 | 0 | 0 | 0 |

In this research, our motivation is aiming to useful applications in wide range of quadruped robot platforms. The main contribution is to propose a new CPG model that not only preserves the ability to generate all primary gaits for quadruped robots but also overcomes the mentioned problem. The proposed model can produce any predefined patterns regardless of the complexity of their waveforms. In addition, the stable limit cycle is also created. The main idea is to use Fourier series and recurrent neural network simultaneously. Besides, to show how it works in real robot, we introduce a biomometic controller using several reflex modules. Finally, several dynamic simulations are performed to validate the efficiency of the proposed controller.

This paper is organized as follows. The detail of the proposed CPG model is presented in Section II. Then, Section III introduces the structure of biomimetic controller and reflex modules. The dynamic simulation and results are discussed in Section IV followed by conclusion in Section V. Appendices provide details of our recurrent learning algorithm and examplary parameter values of the proposed CPG model.

Duc Trong Tran, Ig Moo Koo, Gia Loc Vo, Segon Roh, Hyungpil Moon and Hyouk Ryeol Choi are with school of Mechanical Engineering, Sungkyunkwan University, Suwon, 440-746, Korea.
Sangdeok Park is with Korea Institute of Industrial Technology, Korea.
*Corresponding author (email: hrchoi@me.skku.ac.kr)

## II. Central Pattern Generator

### A. Problem definition

We consider a general problem as follows: given an arbitrary desired pattern $q(t,k) : \mathbf{R} \times \mathbf{R}^m \rightarrow \mathbf{R}$ that is a periodic function of time t with a period of T. In a practical sense, this pattern usually represented desired joint trajectories calculated from some predesigned leg motions. The vector k is composed of some characteristic parameters to change the waveforms of q. For example, this k vector is used to control the stroke length, velocity or other variables related to the motion of each leg.

The task is to build a dynamic CPG model whose the output signals should satisfy the following requirements:

1. The outputs should be $q_i(t,k)=q(t+\varphi_i,k)$ with the set of phase shifting value $\varphi_i$ that can realize all the quadruped primary gaits as shown in table 1.
2. The model should cover all of the possible predesigned trajectories. It means that if we want to change the waveforms of $q(t+\varphi_i,k)$ by controlling vector k or change the period T, we just need to change some inner parameters of the model without rebuilding the whole system again from the scratch.
3. These outputs have to be stable and robust to external perturbations.
4. The waveforms and the phase relations among $q(t+\varphi_i,k)$ should be reformed after some transient time of cancelling the effect of external inputs.

It is important to notice that all the parameters related to $q(t+\varphi_i,k)$ are still assumed to be arbitrary to ensure that this model can be easily applied for wide range of quadruped robot platforms.

In nonlinear dynamics, a stable periodic solution of a dynamic system is referred to a stable limit cycle. To generate some arbitrary predesigned set of stable limit cycles seems to be an impossible task if we stick directly into finding some differential equations as in Eq. 1. However, if we have some pre-analysis of the desired patterns $q(t+\varphi_i,k)$, then we can build a nice CPG model that can satisfy all the above requirements. The detail of the idea is introduced in the next section.

### B. Proposed CPG Model

There are two main difficulties in designing a CPG: one is to ensure a set of arbitrary waveforms of the desired patterns $q_i(t,k)$ and their phase relation, and the other is to maintain these characteristics stably even in the presence of external perturbations. To solve these problems, we use two mathematical tools: Fourier series and Recurrent Neural Network (RNN). The core idea is to use Fourier series to decompose the arbitrary patterns q(t,k) into simpler dynamic sine wave oscillators and generate these oscillators using RNN. The detail structure of the proposed model is shown in Fig. 1.

First, we start with the only assumption of the desired pattern where q(t,k) is a periodic function of time with some

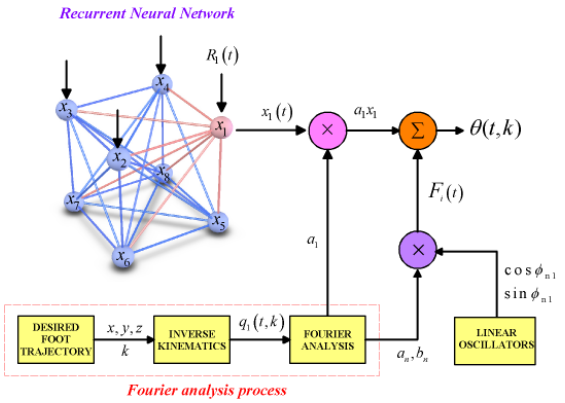


Fig. 1. The proposed CPG model structure.

period T. It is well-known that any periodic function can be easily approximated using Fourier series:

$$\theta_i(t,k) = \sum_{n=0}^{N} \{a_n(k)\cos\phi_{ni} + b_n(k)\sin\phi_{ni}\} \approx q_i(t,k) \qquad (2)$$

where

$$a_n(k) = \frac{2}{T}\int_0^T q(t,k)\cos(\omega_n t)\,dt\ ; \qquad \phi_{ni} = \omega_n(t+\varphi_i)$$

$$b_n(k) = \frac{2}{T}\int_0^T q(t,k)\sin(\omega_n t)\,dt\ ; \qquad \omega_n = \frac{2\pi n}{T}$$

We just use N=6 first terms in the Fourier series to approximate q(t,k). By this way, any arbitrary, even complicated, periodic pattern q(t,k) will be decomposed into a sum of N simple $\sin\phi_{ni}$ and $\cos\phi_{ni}$ oscillators with different frequencies $\omega_n$ and phase $\varphi_i$. In addition, Fourier coefficients $a_n(k)$ and $b_n(k)$ (n=1..N) are functions of vector k such that it preserves the ability of controlling the waveform of $\theta_i(t,k)$ by tuning k directly. Henceforth, to complete the task, we need to find some nonlinear autonomous dynamic system that generates these sine and cosine functions with the limit cycle effects.

Without lost of generality, we choose only the first frequency cosine function $\cos\phi_{1i}$ as the target to be generated by some nonlinear oscillators, the other sine and cosine are simply produced by linear oscillators. For doing this, we use Recurrent Neural Network. This network includes eight neurons as depicted in Fig.1. There are four input-output neurons (1 to 4) and the other four hidden calculating neurons (5 to 8).

In this work, we use eight one-dimensional neurons fully interconnected so that the total dimension of the network reduces by a half in comparison with Buono's model [11]. We use nonlinear connection with the typical dynamic equation as follows.

$$\frac{1}{\tau_k}\dot{x}_k(t) = -x_k(t) + \alpha_k \tanh\left(\sum_{l=1}^{8} w_{kl}x_l\right) + R_k(t) \qquad (3)$$

where $x_k$(t) represents the inner state of neuron $k$; $\tau_k$ is time constant of neuron $k$; $w_{kl}$ is the weight connection from neuron $l$ to neuron $k$; tanh is squash activation function; $\alpha_k$ is additional scaling parameter to enhance the accuracy of the training process and $k$=1..8. The $R_k$(t) is the total input signal to the neuron k.

For training this network, we use the Real-time Recurrent Learning method [13]. The detail calculation of this training process is shown in Appendix A. The targets for the four input-output neurons are $\cos\phi_{1i}$ functions and we choose the targets for the four hidden neurons are $\sin\phi_{1i}$ accordingly. By numerical simulation, for each $\varphi_i$, the limit cycle always appears and behaves stably at the desired patterns, but it is still very hard to prove the existence and stability of a limit cycle in eight-dimensional space. A rigorous analysis of this network is our on going research.

Finally, combining all the parts together, we have the final outputs:

$$\theta_i(t) = a_1 x_i(t) + F_i(t) \tag{4}$$

$$F_i(t) = a_0 + b_1 \sin\phi_{1i} + \sum_{n=2}^{N} \{a_n \cos\phi_{ni} + b_n \sin\phi_{ni}\}$$

where $x_i$(t) is the output from neuron i; $a_n$, $b_n$ and $\phi_{ni}$ are the same in Eq. 2; $\sin\phi_{ni}$ and $\cos\phi_{ni}$ are the outputs from linear oscillators. Here, $x_i$(t) is to maintain the stable limit cycle characteristic and F(t) acts like waveform regulation part to ensure the output of CPG model $\theta_i$(t,k) to be exactly equal to the predesigned signal $q_i$(t,k).

Even be successfully to generate any complicated periodic patterns, but this method fails to generate constant functions. Besides, when there is some external perturbation, the RNN preserves only the phase relations among $x_k$(t) but not the phase relation among $x_k$(t) and linear oscillators. Therefore, there would be some delaying time among these signals. For solving this problem, we put some phase regulation inputs $r_i$ to the RNN neurons which have the form:

$$r_i = c(\cos\phi_{1i} - x_i) \tag{5}$$

where $x_i$ is the inner state variable of neuron i (i=1..4); c is some constant coefficient. These inputs $r_i$ are a part of the total inputs $R_k$(t) in Eq. 3. By this way, the whole system phase relation is guaranteed.

### C. Simulation results

To verify the proposed model, several numerical simulations are performed. We choose an example of the predesigned patterns as shown in Fig. 2(a), which are actually the desired trajectories for controlling the knee joints of the quadruped robot simulation model in section V to realize trot gait. The period of these patterns is 1 second and the phase relations among them are shown in the phase plane depicted in Fig. 2(b). Besides, it is easily to notice that the basic waveforms of the four patterns are different. For convenience, we just show the graphical results of the model. The detailed parameters are shown in Appendix B.

To test the stability of the model numerically, we apply

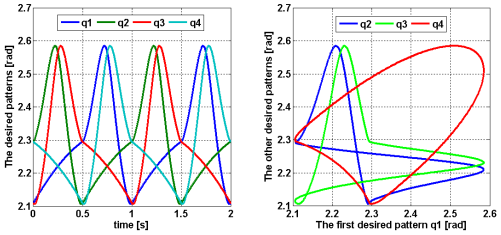


Fig. 2. An example of predesigned patterns for CPG (a) The desired patterns; (b) The phase plane.

several perturbations into the RNN network including sudden impact, constant, ramping and some special periodic signals separately and simultaneously. The result is shown in Fig. 3. In this simulation, we put an impact about 2000 rad/s into neuron 1 at t=2s, a constant signals about -20rad/s into neuron 2, a ramping signal with the rate 40rad/s$^2$ into neuron 3 and a special signal 20*exp(sin(2πt/T)) [rad/s] all during 2 second from t=1s to t=2s.

As shown in Fig. 3, in the steady state, the outputs of CPG model approximate quite well the desired patterns. Not only that, the continuity of the outputs of CPG is always maintained right after the external inputs are applied. It takes about 1 second (one period) for the model to adapt and return to the steady state after the disappearance of the inputs. Obviously, not only the desired waveform is maintained but also the desired phase relation among the outputs.

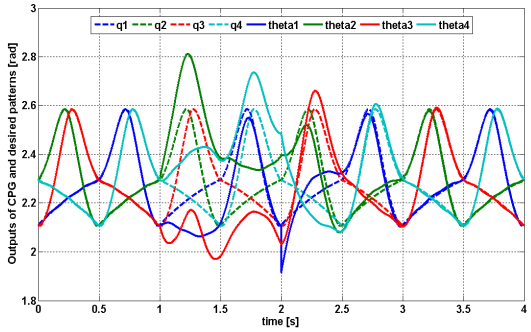


Fig. 3. The behaviors of the outputs signals of CPG when several external inputs are applied.

## III. Biomimetic Controller

The structure of the proposed controller is depicted in Fig. 4 and Fig. 5. There are 3 main parts: sensor group, CPG model, and the central controller.

In this controller, we assume that the motion of each joint in one leg is controlled independently and the four same joints in four legs are controlled by a CPG unit as introduced in the previous section. If each leg has m degrees of freedom, then the CPG model needs m independent units to control the motion of the four legs as shown in Fig. 4. All the sensory signals including joint angle, joint velocity, contact force, roll pitch yaw (RPY) angle and body acceleration are fed into the

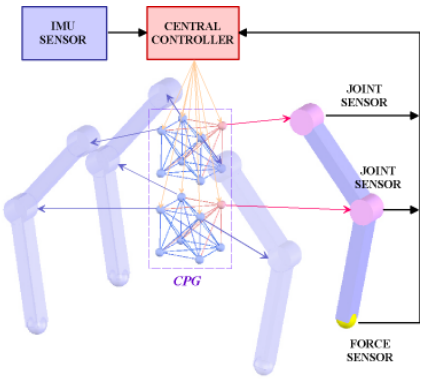


Fig. 4. The structure of proposed biomimetic controller.

central controller. Then, this central controller will process and generate some reflex stimulus signal to feed into the CPG model. The detail interaction among these parts is shown in Fig. 5.

### A. CPG units

The detail structure of the CPG unit is already introduced in the previous section. Here, we discuss how to control the outputs of the CPG by tuning the vector k.

As mentioned in Section II, the vector k is composed of parameters to control the leg motion such as stroke length, feet velocity and so on. The point is that the integration to calculate these coefficients is usually complicated to have closed-form answers as functions of k and numerical method seems to be a better choice. Therefore, instead of considering a continuous space of vector k, we can divide it into a discrete set of values in the available range. And then, we process the Fourier calculation for all of these values of k and save the results in a library. Later, we can load the desired value directly from the library into the CPG model. By this way, we can save the time consuming for calculating the Fourier coefficients including also the inverse kinematics calculations.

### B. State analysis module

This module functions as an observer that monitors all the signals from sensors and compares them with some preset thresholds. If the sensory values exceed these thresholds, then this module will send an activating command to some suitable reflex modules. In addition, all the basic calculations including forward kinematics, inverse of jacobian matrix, and so on are done here in this module.

### C. Roll Pitch Yaw (RPY) balance module

The changes of RPY angles in small range are balanced by this module. We use a detail calculation as follows.

$$r^i_{rpy} = -k_{rpy} J^{-1} \frac{\left(R_{desire} - R_{real}\right) p_i}{T_{reflex}} \qquad (6)$$

where $r^i_{rpy}$ is the reflex signal sent to neuron i in CPG; $J^{-1}$ is the inverse of jacobian matrix; $R_{desire}$ and $R_{real}$ are the desired and real rotation matrix of robot body with respect to center frame; $p_i$ is the position vector of shoulder joint of leg i; $T_{reflex}$ is desire reflex time and $k_{rpy}$ is a scale constant.

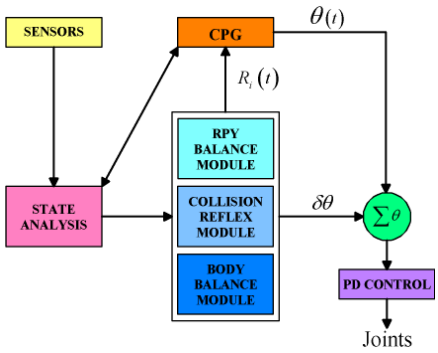


Fig. 5. Detailed structure of the central controller.

Obviously, $(R_{desire}-R_{real})p_i/T_{reflex}$ is the necessary average velocity vector to move the shoulders from current position to the desired position in $T_{reflex}$ seconds. Pre-multiplying this by inverse of jacobian matrix would be the necessary angular average velocity for the joints. This is reasonable because the reflex vector now is in the same space with the differential of inner state variable of the RNN inside of CPG model; it means joint angular velocity space.

### D. Collision reflex module

This module is designed to deal with some sudden collision between the feet and the terrains. The idea is to move along the direction of reaction force to reduce the contact force effect. The reflex signal is calculated as follows.

$$r^i_{col} = J^{-1} \frac{F_{contact}}{k_{vs} T_{reflex}} \qquad (7)$$

where $r^i_{col}$ is the reflex signal sent to neuron i; $J^{-1}$ is the inverse of jacobian; $F_{contact}$ is contact force value and $T_{reflex}$ is some reflex time constant. The number $k_{vs}$ acts like the stiffness of a virtual spring attach to the feet in the direction of reaction force. Thus $F_{contact}/(k_{vs}T_{reflex})$ would be necessary average velocity for the feet to move in the direction to reduce the contact force in $T_{reflex}$ seconds. In fact, the contact force value in real platform is usually very noisy and unreliable, so we can use directly a constant reflex velocity v instead of $F_{contact}/(k_{vs}T_{reflex})$ in Eq. 7. Certainly, the effect is not always best for all of the cases.

### E. Body balance module

This module deals with the big changes of roll-pitch angles, or a sudden lateral acceleration due to external force. For simplicity, we just add some lateral motion to the desired feet motion in the direction that cancels down the inertia effects created by the sudden external force (opposite rotation direction that the robots roll or pitch to, or same direction that the body is accelerated). The reflex calculation is:

$$r^i_{body} = k_{body} J^{-1} \frac{\Delta y}{T_{reflex}} \qquad (8)$$

where $r^i_{body}$ is the reflex signal; $k_{body}$ is -1 or 1 up to the necessary reflex direction; $\Delta y$ is the distance that robot should step laterally, here we choose it to be a half of the robot width; $T_{reflex}$ is some time constant.

## IV. Simulation Results

To verify the proposed controller, a dynamic simulation of a quadruped walking robot was developed using Open Dynamic Engine [15].

The structure of this robot consists of 20 DOFs, 5 DOFs for each leg including 4 actuated hinge joints and 1 passive slider joints. The length of this robot is about 84 centimeters. The width is about 40 cm and the working height is about 80 cm. The total weight of this robot is about 150 kgs. The fiction coefficient is about 0.7 in all terrains (we assume this is a rubber – concrete contact surface [14]).

At first, we build a slope terrain with 12 degrees inclined angle up and down. The height of this terrain is about 2.3 m. The robot starts from the plate ground and walks forward the slope. It walks up and down the slope very stable at the speed 0.5m/s. The RPY angles of the robot during walking through this slope are shown in Fig. 7.

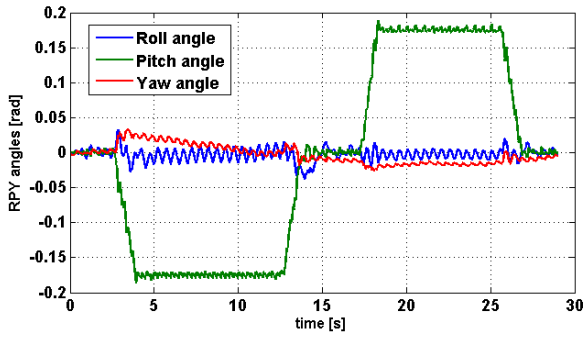


Fig. 7. RPY angles during walking on a 12 degrees slope.

Secondly, we test the behavior of the robot when a sudden perturbation is applied. In this test, we let the robot walks forward at the speed 1m/s. After a while, we give a sudden impact force about 2000N to the left side of robot body. Certainly, the robot immediately rolls down to the right side but the controller also detects this unstable state and moves the four legs laterally to compensate the perturbation. And as expected, after some time of balancing, the robot succeeds to overcome the inertia force created by the perturbing force and returns to the stable walking. The recorded RPY angles in this test are shown in Fig. 8.

Finally, we build a simple but challenging rough terrain using rock bars. In this terrain, several rock bars with 12 centimeter width and 2 to 6 centimeters height are arranged crossly randomly on the ground to make a grid net. Therefore, the terrain becomes very discrete with different height at every point. Because it is discrete terrain, there are several holes like region insides of this terrain, the robot legs are easily stuck in these holes. Thus to maintain the stability during walking through this terrain is very difficult. In this test, similar to the above cases, we let the robot walks through this terrain at the speed 1m/s. Certainly, as in the normal rocky terrains, the robot feet collide continuously with the bars. Sometime, some foot of the robot sticks into some hole for a while. Even so, the robot can still successfully traverse this terrain and move forward stably. The RPY angles change in this test is shown in Fig. 9.

A video file records all the simulations above is attached with this paper.

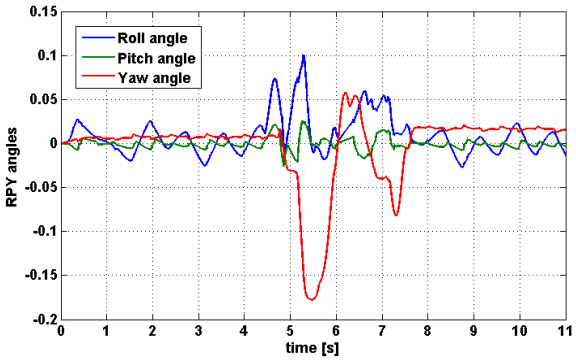


Fig. 8. RPY angles during sudden given lateral impact.

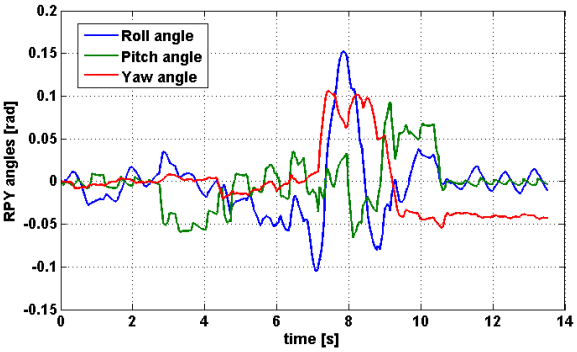


Fig. 9. RPY angles during walking on rock bars terrain.

## V. Conclusion

In this paper, we introduced a new method to model the central pattern generators in control of quadruped locomotion. Not only can this new model generate all the primary gaits of quadrupeds stably with limit cycle effect, but it also has the ability of tuning the periodic outputs with arbitrary waveforms. In addition, a biomimetic controller using the proposed CPG is presented. This controller also includes several additional reflex modules to help CPG to adapt with the change of terrains and external perturbation. Finally, the efficiency of the proposed controller is validated by performing several dynamic simulation of a quadruped robot.

## Appendix

### A. *Realtime recurrent learning algorithm*

This algorithm originally deals with the discrete variables, so we first change the Eq. (2) into discrete form. Let $t_n=n\Delta t$ is discrete points in time space with time step $\Delta t$ and $x_k(n)=x_k(t_n)$ for k=1..8. We use the first order Taylor approximation:

$$x_k(n+1) = x_k(n) + \dot{x}_k(n)\Delta t$$

and substitute into Eq. (2), and then rearrange the whole equation, we have:

$$x_k(n+1) = \left(1-\tau_k\Delta t\right)x_k(n) + \tau_k\Delta t\tanh\left(net_k(n)\right) \quad (9)$$

Let $d_k(t)$ is the desired output value of neuron k and the training time is up to $t_N$. Therefore, the total error function would be:

$$E = \sum_{n=0}^{N} e(n) = \sum_{n=0}^{N} \sum_{k=1}^{8} \frac{1}{2} \left( d_k(n) - x_k(n) \right)^2$$

From now on, we use gradient descent method to find the minimum of the error function. The gradients are calculated as follows.

$$\frac{\partial E}{\partial w_{ij}} = \sum_{n=0}^{N} \frac{\partial e(n)}{\partial w_{ij}} = \sum_{n=0}^{N} \sum_{k=1}^{8} \left( d_k(n) - x_k(n) \right) \frac{\partial x_k(n)}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial \tau_i} = \sum_{n=0}^{N} \frac{\partial e(n)}{\partial \tau_i} = \sum_{n=0}^{N} \sum_{k=1}^{8} \left( d_k(n) - x_k(n) \right) \frac{\partial x_k(n)}{\partial \tau_i}$$

$$\frac{\partial E}{\partial \alpha_i} = \sum_{n=0}^{N} \frac{\partial e(n)}{\partial \alpha_i} = \sum_{n=0}^{N} \sum_{k=1}^{8} \left( d_k(n) - x_k(n) \right) \frac{\partial x_k(n)}{\partial \alpha_i}$$

For convenience, we define:

$$p_{ij}^k(n) = \frac{\partial x_k(n)}{\partial w_{ij}};\ q_i^k(n) = \frac{\partial x_k(n)}{\partial \tau_i};\ r_i^k(n) = \frac{\partial x_k(n)}{\partial \alpha_i}$$

Using Eq. (9), we calculate:

$$p_{ij}^k(n+1) = \left(1 - \tau_k \Delta t\right) p_{ij}^k(n) + \alpha_k \tau_k F(n) \Delta t \left( \sum_{l=1}^{8} w_{kl} p_{ij}^l(n) + \delta_{ki} x_j(n) \right)$$

$$q_i^k(n+1) = \left(1 - \tau_k \Delta t\right) q_i^k(n) + \delta_{ik} \alpha_k \Delta t \tanh\left(net_k(n)\right) - \delta_{ik} x_k(n) \Delta t + \alpha_k \tau_k F(n) \Delta t \left( \sum_{l=1}^{8} w_{kl} q_{ij}^l(n) \right)$$

$$r_i^k(n+1) = \left(1 - \tau_k \Delta t\right) r_i^k(n) + \delta_{ik} \tanh\left(net_k(n)\right) + \alpha_k \tau_k F(n) \Delta t \left( \sum_{l=1}^{8} w_{kl} r_i^l(n) \right)$$

Assuming the inner state are independent with the initial state of all parameter of the RNN, then we have:

$$p_{ij}^k(0) = q_i^k(0) = r_i^k(0) = 0 \qquad \forall k, i, j$$

Using the recursive equations above, we can calculate all p, q and r through time. Therefore, the updating step for all parameters would be:

$$\Delta w_{ij} = -\gamma_w \frac{\partial E}{\partial w_{ij}};\ \Delta \tau_i = -\gamma_\tau \frac{\partial E}{\partial \tau_i};\ \Delta \alpha_i = -\gamma_\alpha \frac{\partial E}{\partial \alpha_i}$$

where $\gamma_w$, $\gamma_\tau$ and $\gamma_\alpha$ are descent steps for w, $\tau$ and $\alpha$.

### *B. Results of CPG model in section II*

The connection weight matrix is:

$$W = \begin{bmatrix} S & S \\ S & S \end{bmatrix}$$

where

$$S = \begin{bmatrix} 0.0602 & -0.0213 & -0.0602 & 0.0213 \\ 0.0213 & 0.0602 & -0.0213 & -0.0602 \\ -0.0602 & 0.0213 & 0.0602 & -0.0213 \\ -0.0213 & -0.0602 & 0.0213 & 0.0602 \end{bmatrix}$$

The time constant $\tau_k \cong 17.7938$ and $\alpha_k \cong 4.2255$ for all k=1..8.

The Fourier coefficients for the $q_1$ and $q_2$ is

$a_n = \begin{bmatrix} 2.2941 & -0.0982 & -0.0916 & 0.0073 & 0.0039 & -0.0026 & -0.0032 \end{bmatrix}$

$b_n = \begin{bmatrix} 0 & -0.1366 & 0.0275 & 0.0332 & -0.0030 & -0.0003 & 0.0017 \end{bmatrix}$

The Fourier coefficients for the $q_3$ and $q_4$ is

$a_n = \begin{bmatrix} 2.2941 & 0.0982 & -0.0916 & -0.0073 & 0.0039 & 0.0026 & -0.0032 \end{bmatrix}$

$b_n = \begin{bmatrix} 0 & -0.1366 & -0.0275 & 0.0332 & 0.0030 & -0.0003 & -0.0017 \end{bmatrix}$

## ACKNOWLEDGMENT

This research was supported in part by the project of the dual-use technology for military and civilian missions ("Development of Quadruped Robots") of the Ministry of Commerce, Industry and Energy (MOCIE), Korea.

## REFERENCES

[1] Y. Fukuoka, H. Kimura and A.H. Cohen, Adaptive Dynamic Walking of a Quadruped Robot on Irregular Terrain Based on Biological Concepts, *Int. J. Robotics Research*, vol. 22, 2003, pp 187-202.

[2] L. Righetti and A.J. Ijspeert, "Design methodologies for central pattern generators: an application to crawling humanoids", *in IEEE Int. Conference on Robotics and Automation*, Italy, 2007, pp 262-268.

[3] A. Fujii, N. Saito, K. Nakahira, A. Ishiguro and P.Eggenberger, "Generation of an Adaptive Controller CPG for a Quadruped Robotwith Neuromodulation Mechanism", *Int. Conference on IntelligentRobots and Systems*, 2002, pp 2619-2624.

[4] K. Matsuoka, Sustained oscillation generated by mutually inhibiting neurons with adaptation, *J. Biological Cybernetics*, vol.52, 1985, pp 367-376.

[5] S.A. Bailey, *Biomimetic control with a feedback coupled nonlinearoscillator: Insect experiments, design tools, and hexapedal robotadaptation results*, PhD. Thesis, Stanford University, USA, 2004.

[6] M. MacKay-Lyons, Central Pattern Generation of Locomotion: AReview of the Evidence, *J. Physical Therapy*, vol. 82, 2002, pp 69-82.

[7] V. Scesa, B. Mohamed, P. Henaff and F.B. Ouezdou, "Dynamic Recurrent Neural Network for Biped Robot Equilibrium Control: Preliminary Results", *Proceedings of IEEE Int. Conference on Robotics and Automation*, Spain, 2005, pp 4114-4119.

[8] M. Golubitsky, I. Stewart, P.K. Buono and J.J. Collins, Symmetry in locomotor central pattern generators and animal gaits, *J. Nature*, vol. 401, 1999, pp 693-695.

[9] J.J. Collins and I.N. Stewart, Couple Nonlinear Oscillators and the Symmetries of Animal Gaits, *J. Nonlinear Science*, vol. 3, 1993, pp 349-392.

[10] P.L. Buono and M. Golubitsky, Models of central pattern generators for quadruped locomotion, *J. Mathematical Biology*, vol. 42, 2001, pp 291-326 .

[11] K. Seo and J.J.E. Slotine, "Models for Global Synchronization in CPG-based Locomotion", *in IEEE Int. Conference on Robotics and Automation*, Italy, 2007, pp 281-286.

[12] L.R. Palmer III and D.E. Orin, "Quadrupedal Running at High Speed Over Uneven Terrain", *Int. Conference on Intelligent Robots and Systems*, 2007, pp 303-308.

[13] R. J. Williams and D. Zipser, A learning algorithm for continually running fully recurrent neural networks, *Neural Computation*, vol. 1, 1989, pp. 270-280.

[14] http://www.roymech.co.uk/Useful_Tables/Tribology/co_of_frict.htm

[15] ODE, *http://www.ode.org.*