The Pole Balancing Problem with Enzymatic Numerical P Systems

Domingo Llorente–Rivera[1], Miguel A. Gutiérrez–Naranjo[2]

[1] Department of Computer Science and Artificial Intelligence
University of Seville, Seville, Spain
domingo.llorente.rivera@gmail.com

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012, Spain
magutier@us.es

Summary. Pole balancing is a control benchmark widely used in engineering. It involves a pole affixed to a cart via a joint which allows movement along a single axis. In this problem, the movement of the cart is restricted to the horizontal axis by a track and the pole is free to move about the horizontal axis of the pivot. The system is extremely unstable and, the cart must be in constant movement in order to preserve the equilibrium and avoid the fall of the pendulum.

In this paper, we study the pole balancing problem in the framework of Enzymatic Numerical P Systems and provide some clues for using them in more complex systems.

1 Introduction

Numerical P systems (NPS for short) were introduced in [7] with the aim of adding ideas from economic and business processes to the framework of Membrane Computing. They represent a break with respect to the previous P system models since they introduce the concept of *variable* and *real numbers* in the framework of Membrane Computing. In the general framework of Membrane Computing (called *symbolic* P systems, in order to stress the differences with *numerical* P systems), membranes can be seen as encapsulations of the Euclidean space where multisets of objects are placed. The computation in such devices is performed by the application of rules which send objects from one to other membrane (maybe modified) or modify the membrane structure (see [8]). In NPS, membranes do not contain multisets of objects. They contain variables with associated numerical values. These numerical values can be integer, rational or real numbers. Instead of using rules inspired in biochemical reactions, the computation of these new devices is performed by *programs* consisting of two parts: a *production function* and a *repartition protocol*. Production functions are real-valued functions of type $F : \mathbb{R}^k \to \mathbb{R}$ which take

the k variables which appear in the membrane where the program is defined and computes a real value. The computed number is then distributed among different variables according to the *repartition protocol*.

In spite of its undoubted potential as computational devices, in the literature there are very few papers devoted to this model (see, e.g., [1, 2, 3, 4, 5, 9, 10, 11] and references therein). Most of them devoted to *enzymatic* numerical P system (ENPS), a model introduced in [3] where enzymatic-like variables are introduced in the NPS in order to avoid the non-determinism in the choice of a program in a membrane.

Although the original inspiration of numerical P system was the economic processes, the main field of the applications has been control problems. These problems are on the basis of many industrial processes and the design of software controllers for more and more sophisticated devices is nowadays a challenge for researchers. The household thermostat is a classic example of control problem: provided the changing temperature outside, the thermostat must maintain the temperature inside home close to a desired level. This implies react to the changes in an unpredictable real-world providing an appropriate response in a short interval of time.

Beyond simple examples, the design of controllers for many real world is an extremely complex task. If we extend the thermostat example to a more general climate control system, a linear controller will not be able to regulate the temperature adequately.

Usually, the control system is a software program that takes the right decision for the input. For this input-output interaction, the software receives an input from the sensor and takes a decision as output. It is crucial for the final solution to obtain a real-time response in less than 10 milliseconds. For this reason, the control software must be as small as possible in order to obtain a quick response.

In this paper we go on with the study of NPS as devices for control problem (see, e.g. [2, 4]). As pointed out by Gh. Păun in [6], controlling drones can be a good application for this model and it can be an extension of the use of NPS for 2D travelling robots found in the literature. *Drone* is the popular name for an unmanned aerial vehicle which can be seen as a mobile 3D robot. From a technical point of view, the main difference between the control of 2D travelling robots and drones is the stability. The drone must keep the horizontal position as much as possible regardless the air conditions. This implies the effective real-time control of the different engines according to the changes in the environment. The control of drones is nowadays a research field for the industry and it is a really hard task. In a certain sense, the stability problem of a drone can be seen as the generalization of a well-known problem in control, the *pole-balancing* problem.

The pole-balancing problem is a feedback control system with the desired behavior of balancing a pole (an inverted pendulum) that is connected to a motor driven cart by a ball-baring pivot (see Fig. 1). In this problem, the movement of the cart is restricted to the horizontal axis by a track, and the pole is free to move about the horizontal axis of the pivot. The system is extremely unstable and the

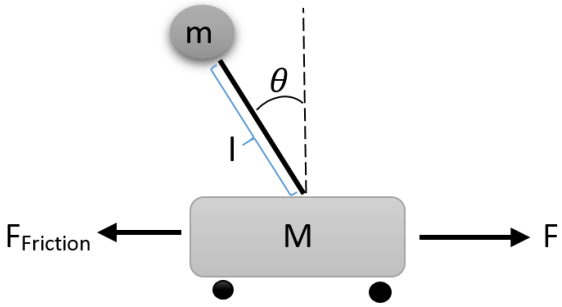

Fig. 1. Pole Balancing problem

cart must be in constant movement in order to preserve the equilibrium and avoid the fall of the pendulum. In a more general situation (a drone, by example) the movement of the device must be controlled in three degrees of freedom, but it is essentially the same problem, so the pole-balancing problem can be seen as a first approach.

In this paper, we provide a theoretical study of the pole-balancing problem in the framework of the ENPS and provide some ideas for further uses of ENPS in control problems. The paper is organized as follows: Firstly, a brief introduction to ENPS and to the Pole Balancing Problem is given. Next we provide some hints about how the problem can be dealt with ENPS and finally some conclusions and future work lines are presented.

2 Enzymatic Numerical P Systems

Next, we briefly recall the definition of enzymatic numerical P systems, More details can be found in [3]. An enzymatic numerical P system is formally expressed by:

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \ldots, (Var_m, Pr_m, Var_m(0)))$$

where:

- m is the number of membranes used in the system (degree of Π) ($m \geq 1$);
- H is an alphabet that contains m symbols (the labels of the membranes);
- μ is a tree-like membrane structure;
- Var_i is a set of variables from membrane i, and the initial values for these variables are $Var_i(0)$, $i \in \{1, \ldots, m\}$;
- Pr_i is the set of programs from membrane i, $i \in \{1, \ldots, m\}$. Programs process variables and have one of the following forms:
  (a) Non-enzymatic form

$$Pr_{j,i} = F_{j,i}(x_{1,i}, \ldots, x_{k_i,i}) \rightarrow c_{j,1}|v_1 + \cdots + c_{j,n_i}|v_{n_i}$$

(b) Enzymatic form

$$Pr_{j,i} = F_{j,i}(x_{1,i}, \ldots, x_{k_i,i})(e_j \rightarrow) \rightarrow c_{j,1}|v_1 + \cdots + c_{j,n_i}|v_{n_i}$$

where $e_j \in Var_i$ is an enzyme-like variable which controls the activation of the rule.

Rules have two components, a *production function* and a *repartition protocol*. The l-th program of the membrane i has the following form:

$$Pr_{l,i} = (F_{l,i}, c_{l,1}|v_1 + \cdots + c_{l,n_i}|v_{n_i})$$

where $F_{l,i} : \mathbb{R}^{card(Var_i)} \rightarrow \mathbb{R}$ is a real-valued function such that computes a real number from the values of the variables in Var_i; $c_{l,1}, \ldots, c_{l,n_i}$ are natural numbers and $v_1, \ldots, v_{n_1}$ are the variables of the membrane i together with the variables from the immediately upper membrane, and those from the immediately lower membranes. If the corresponding c_i is 0, the expression $0|v_i$ is omitted.

If $card(Pri) = 1$ for $i \in \{1, \ldots, m\}$, then there is one production function per each membrane and the system is deterministic. In case of multiple programs per membrane, one rule is non-deterministically selected.

A universal clock is considered and, at each time t, all the variables have associated a value. The computation is performed by computing the new value of the variables. Such computation is performed in the following way. A rule is *active* if it is in the non enzymatic form or if the associated enzyme has a greater value than one of the variables involved in the production function. In parallel, in each membrane an active program is chosen and its production function is used in order to calculate a *production* from the value of the local variables. Once calculated, the repartition protocol is used in order to compute the proportion of such value that it is send to each variable. The coefficients $c_1 \ldots c_n$ in the repartition protocol $c_1|v_1 + \cdots + c_n|v_n$ specify the proportion of production distributed to each variable $v_1 \ldots v_n$. Namely, such protocol sends to the variable v_i the value

$$q_i = \frac{production \times c_i}{\sum_{i=j}^{n} c_j}$$

The new value of the variable is the addition of the contribution of each applied program. In each membrane of the system one uses one program at the time, and this happens in parallel in all membranes.

A variable x is called *productive* if it does appear in a production function, and then is *consumed* and reset to zero, otherwise the initial value is added to the received contributions. The values of the variables at next time step are computed by using repartition protocols, and so, portions distributed to variables are added to form the new value.

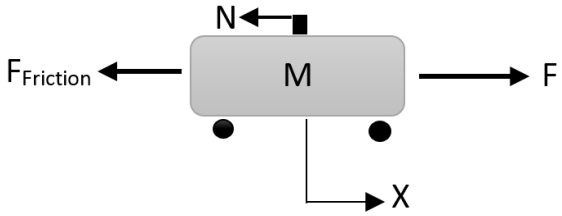

Fig. 2. Cart of the Pole Balancing

3 The Pole Balancing Problem

Pole balancing is an control benchmark historically used in engineering. It involves a pole affixed to a cart via a joint which allows movement along a single axis. The cart is able to move along a track of fixed length.

A trial typically begins with the pole off-center by a certain number of degrees. The goal is to keep the pole from falling over by moving the cart in either direction, without falling off either edge of the track. The controller receives as input information about the system at each time step, such as the positions of the poles, their respective velocities, the position and velocity of the cart, etc. An even more difficult extension of this problem involves a cart which can move in a three dimensional space via three or more engines. In such situation the target is not keeping a pole in a vertical position but keeping the cart as horizontal as possible. In this paper we do not consider such generalization and focus on the simple pole balancing problem.

The pole balancing problem can be analysed as the conjunction of two models: focusing on the cart (see Fig. 2) and focusing on the bar (see Fig. 3). Obviously, the applied force over one of these models results in the modification of the state of the other model. In the first model (Fig. 2) several parameters must be considered: F, force for controlling the system; $F_{Friction}$, force of the friction of the cart in its movement on the railway; M, mass of the cart; N, force of the pole over the cart. The second model focus on the bar of the pole balancing (Fig. 3), where θ is the angle of the bar with respect to the vertical, l is the length of the bar and m is the mass of the ball placed on the top of the bar. For the control of the pole balancing, the control software (the NPS in our study) has to know the current state of the pole, (x, θ) and ($\dot{x}$, $\ddot{x}$, $\dot{\theta}$, $\ddot{\theta}$), where x represents the *position* of the cart, and $\dot{x}$, $\ddot{x}$ the *speed* and *acceleration* respectively. The angle θ represents the angle of the bar with respect to the vertical position and $\dot{\theta}$, $\ddot{\theta}$ the *angular* speed and acceleration (resp).

The equations that define this system are:

$$F = M\ddot{x} + b\dot{x} + N \tag{1}$$

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{2}$$

The system of control is represented by the equation (3), which is the result of adding the equations (1) and (2), where F is the output for the system of control and the force that the controller has to apply to the system, and b is the friction of the cart.

$$F = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{3}$$

For computing $\cos\theta$ and $\sin\theta$ using ENPS, we use the same idea proposed in [5] where the functions are approximated by using their analytic expressions as infinite sums shown in equations (4) and (5). There approaches will be calculated in the designed ENPS by the membranes *Cosine* and *Sine*, respectively.

$$cos(x) = \sum_{0}^{\infty}(-1)^n \frac{x^{2n}}{(2n)!} \tag{4}$$

$$sin(x) = \sum_{0}^{\infty}(-1)^n \frac{x^{(2n+1)}}{(2n+1)!} \tag{5}$$

The analytic expression of the *cosine* can be written as

$$cos(x) = \sum_{n=0}^{\infty} ac_n$$

where $ac_0 = 1$ and ac_n is recursively obtained as follows:

$$ac_{n+1} = (-ac_n) \times \frac{\theta^2}{(2n)(2n-1)}$$

Analogously, the analytic expression of the *sine* can be written as

$$sin(x) = \sum_{0}^{\infty} as_n$$

where $as_0 = 1$ and as_n is computed as

$$as_{n+1} = (-as_n) \times \frac{\theta^2}{(2n)(2n+1)}$$

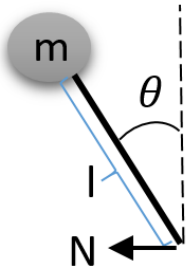

Fig. 3. Bar of the Pole Balancing

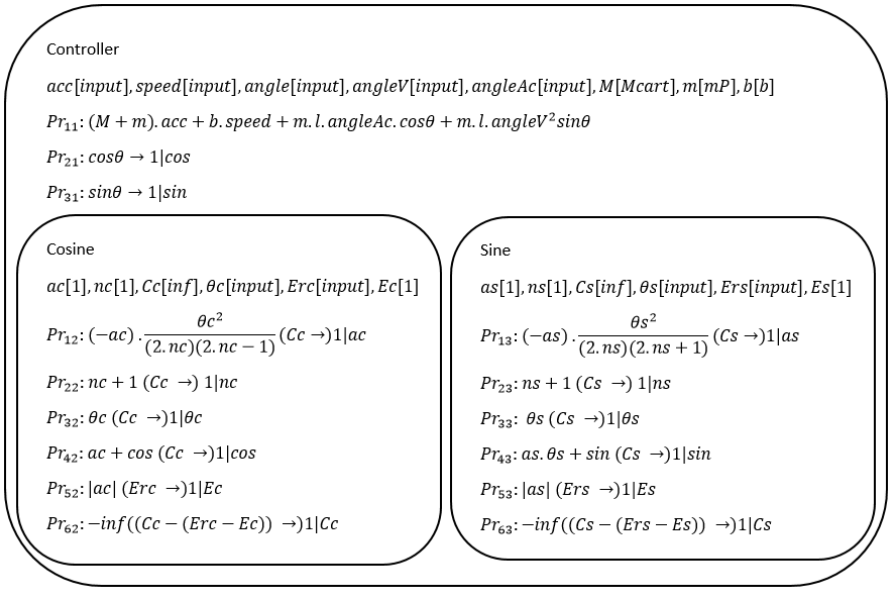

Fig. 4. ENPS membrane applied system for control pole balancing

4 ENPS Applied to the Pole Balancing Problem

In this section, we report a work-in-progress on the design of an ENPS as a software solution for the control of the pole balancing problem. To this aim, the different forces that affect the system are examined and the interaction among them are computed as a flow of information between the variables of the ENPS. The basic schema is chown in Fig. 3.

The membrane system shown in Fig. 4 is proposed as a preliminar solution for the pole balancing problem, using three membranes: the first membrane *Controller* calculates the necessary force in order to keep the vertical position; the membranes *Cosine* and *Sine* calculate the cos and sin functions for the angle θ. The ENPS can be considered as a software module which receives as input the data $\dot{x}$, $\ddot{x}$, θ, $\dot{\theta}$, $\ddot{\theta}$ and outputs the force F for controlling the system.

The control of the pole balancing is calculated by the rule Pr_{11} which encodes the Equation 3. This rule needs the constants: M, the cart mass; m, the mass of the ball; and l, the length of the bar. It takes as input the state of the system, encoded in the variables: *acc*, acceleration of the cart ($\ddot{x}$); *speed*, velocity of the cart ($\dot{x}$); *angleSpeed* ($\dot{\theta}$) and *angleAcc*, ($\ddot{\theta}$) angle speed and acceleration. In order to approximate $\cos\theta$ and $\sin\theta$ from θ, the *Controller* membrane uses the rules Pr_{21} and Pr_{31}. The *cosine* and *sine* are computed recursively by the rules Pr_{12} for the *cosine* and Pr_{13} for the *sine*, until the current errors, Ec for the *cosine* and Es for the *sine*, are less than Erc and Ers respectively as it is proposed in

[12].Finally, the system returns the control related to equation 3 with the $\cos\theta$ and $\sin\theta$ calculated previously.

Membranes *Cosine* and *Sine* approximate the *cos* and *sin* functions by using the analytic expressions from Eq. (4) and (5). These membranes return *cos* by the rule Pr_{12} and *sin* by the rule Pr_{13}, where the system adds the result for each one in *cos* and *sin*. The membranes stop when the current error is less than the errors provided as parameters, Erc and Ers. The system stop is controlled by rule Pr_{62} for the cosine and Pr_{63} for the sine as the current error is lower than the parameter Erc for the cosine membrane and Ers for the sine membrane.

The following trace shows how the system from Fig. 4 should work:

- Membrane Controller:
  - The input of the system $\dot{x}_0$, $\ddot{x}_0$, θ_0, $\dot{\theta}_0$, $\ddot{\theta}_0$ are the values of the corresponding variables in the initial configuration. We also consider two variables *cosApp* and *sinApp* where the approximated values of the *cos* and *sin* functions will be stored.
  - Production Function:
    - $F_1 = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta$;
    - $F_2 = \cos\theta$;
    - $F_3 = \sin\theta$;
- Membrane Cosine:
  - Variables: ac has an initial value of 1, nc has an initial value of 1, Cc has an initial value of ∞, Ec has an initial value of 1;
  - Production function:
    - $F_4 = (-ac) \times \frac{\theta^2}{(2nc)(2nc-1)}$;
    - $F_5 = nc + 1$;
    - $F_6 = \theta$;
    - $F_7 = ac + cos$;
    - $F_8 = |ac|$;
    - $F_9 = -\infty$;
  - Reparation protocol: ac receives 1 ($C_{21} = 1$), nc receives 1 ($C_{22} = 1$), cos receives 1 ($C_{23} = 1$), Cc receives 1 ($C_{24} = 1$), Ec receives 1 ($C_{25} = 1$);
- Membrane Sine:
  - Variables: as has an initial value of 1, ns has an initial value of 1, Cs has an initial value of ∞, Es has an initial value of 1;
  - Production function:
    - $F_{10} = (-as) \times \frac{\theta^2}{(2ns)(2ns+1)}$;
    - $F_{11} = ns + 1$;
    - $F_{12} = \theta$;
    - $F_{13} = as * \theta$;
    - $F_{14} = |as|$;
    - $F_{15} = -\infty$;
  - Reparation protocol: as receives 1 ($C_{31} = 1$), ns receives 1 ($C_{32} = 1$), sin receives 1 ($C_{33} = 1$), Cs receive ∞ ($C_{34} = 1$), Es receive 1 ($C_{35} = 1$);

- Step 1
  - Membrane Cosine:
    - $ac_{21} = 1$, $nc_{22} = 1$,$cos_{23} = 0$, $\theta = 1$, $Cc = \infty$, $Ec = 1$, $Erc = 0.0001$;
    - Compute productions function's value:
      - $F_4 = (-ac_{21}) \times \frac{\theta^2}{(2nc_{22})(2nc_{22}-1)} \Rightarrow F_4 = -\frac{1}{2}$;
      - $F_5 = nc_{22} + 1 \Rightarrow F_5 = 2$;
      - $F_6 = \theta \Rightarrow F_6 = 1$;
      - $F_7 = ac_{21} + cos_{23} \Rightarrow F_7 = 1$;
      - $F_8 = ac_{21} \Rightarrow F_8 = 1$;
      - F_9 is not executed, because $Cc - (Erc - Ec) = \infty - (0.0001 - 1) = \infty + 1$ is not bigger than Cc;
    - Compute 'unitary portion':
      - $q_4 = F_4/C_{21} \Rightarrow ac_{21} = -\frac{1}{2}$;
      - $q_5 = F_5/C_{22} \Rightarrow nc_{22} = 2$;
      - $q_6 = F_6/\theta \Rightarrow \Theta = 1$;
      - $q_7 = F_7/C_{23} \Rightarrow cos_{23} = 1$;
      - $q_8 = F_8/C_{25} \Rightarrow Ec = 1$;
  - Membrane Sine:
    - $as_{31} = 1$, $ns_{32} = 1$,$sin_{33} = 0$, $\theta = 1$, $Cs = \infty$, $Es = 1$, $Ers = 0.0001$;
    - Compute productions function's value:
      - $F_{10} = (-as_{31}) \times \frac{\theta^2}{(2ns_{32})(2ns_{32}+1)} \Rightarrow F_8 = -\frac{1}{6}$;
      - $F_{11} = ns_{32} + 1 \Rightarrow F_9 = 2$;
      - $F_{12} = \theta \Rightarrow F_{10} = 1$;
      - $F_{13} = as_{31} + sin_{33} \Rightarrow F_{11} = 1$;
      - $F_{14} = |as_{31}| \Rightarrow F_{14} = 1$;
      - F_{15} is not executed, because $Cs - (Ers - Es) = \infty - (0.0001 - 1) = \infty + 1$ is not bigger than Cs;
    - Compute 'unitary portion':
      - $q_{10} = F_{10}/C_{31} \Rightarrow as_{31} = -\frac{1}{6}$;
      - $q_{11} = F_{11}/C_{32} \Rightarrow ns_{32} = 2$;
      - $q_{12} = F_{12}/\theta \Rightarrow \Theta = 1$;
      - $q_{13} = F_{13}/C_{33} \Rightarrow sin_{33} = 1$;
      - $q_{14} = F_{14}/C_{35} \Rightarrow Es = 1$;
- Step 2:
  - Membrane Cosine:
    - $ac_{21} = -\frac{1}{2}$, $nc_{22} = 2$,$cos_{23} = 1$, $\theta = 1$, $Cc = \infty$, $Ec = 1$, $Erc = 0.0001$;
    - Compute productions function's value:
      - $F_4 = (-ac_{21}) \times \frac{\theta^2}{(2nc_{22})(2nc_{22}-1)} \Rightarrow F_4 = \frac{1}{24}$;
      - $F_5 = nc_{22} + 1 \Rightarrow F_5 = 3$;
      - $F_6 = \theta \Rightarrow F_6 = 1$;
      - $F_7 = ac_{21} + cos_{23} \Rightarrow F_7 = -\frac{1}{2}$;
      - $F_8 = ac_{21} \Rightarrow F_8 = \frac{1}{2}$;
      - F_9 is not executed, because $Cc - (Erc - Ec) = \infty - (0.0001 - 1) = \infty + 1$ is not bigger than Cc;

  - Compute 'unitary portion':
    - $q_4 = F_4/C_{23} \Rightarrow ac_{23} = \frac{1}{24}$;
    - $q_5 = F_5/C_{22} \Rightarrow nc_{22} = 3$;
    - $q_6 = F_6/\theta \Rightarrow \Theta = 1$;
    - $q_7 = F_7/C_{23} \Rightarrow cos_{23} = \frac{1}{2}$;
    - $q_8 = F_8/C_{25} \Rightarrow Ec = \frac{1}{2}$;
- Membrane Sine:
  - $as_{31} = -\frac{1}{6}$, $ns_{32} = 2$,$sin_{33} = 1$, $\theta = 1$, $Cs = \infty$, $Es = 1$, $Ers = 0.0001$;
  - Compute productions function's value:
    - $F_{10} = (-as_{31}) \times \frac{\theta^2}{(2ns_{32})(2ns_{32}+1)} \Rightarrow F_8 = \frac{1}{120}$;
    - $F_{11} = ns_{32} + 1 \Rightarrow F_9 = 3$;
    - $F_{12} = \theta \Rightarrow F_{10} = 1$;
    - $F_{13} = as_{31} + ns_{33} \Rightarrow F_{11} = 1 - \frac{1}{6} = \frac{5}{6}$;
    - $F_{14} = |as_{31}| \Rightarrow F_{14} = \frac{1}{6}$;
    - F_{15} is not executed, because $Cs - (Ers - Es) = \infty - (0.0001 - 1) = \infty + 1$ is not bigger than Cs;
  - Compute 'unitary portion':
    - $q_{10} = F_{10}/C_{33} \Rightarrow as_{33} = \frac{1}{120}$;
    - $q_{11} = F_{11}/C_{32} \Rightarrow ns_{32} = 3$;
    - $q_{12} = F_{12}/\theta \Rightarrow \Theta = 1$;
    - $q_{13} = F_{13}/C_{33} \Rightarrow sin_{33} = \frac{5}{6}$;
    - $q_{14} = F_{14}/C_{35} \Rightarrow Es = \frac{1}{6}$;

- Step N-1:
  - Using the same reason for the membranes *Cosine* and *Sine*, both membranes are executed until error is less than Erc, $Ec < Erc$, for the Cosine and Ers, $Es < Ers$, for the Sine. Then the execution stops.
  - Membrane Controller:
    - $\cos\theta = 1$, $\sin\theta = 1$, $F[0]$;
    - Compute productions function's value:
      - $F_1 = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \Rightarrow F_1 = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta} - ml\dot{\theta}^2$;
      - $F_2 = \cos\theta \Rightarrow F_2 = \cos$;
      - $F_3 = \sin\theta \Rightarrow F_3 = \sin$;
    - Compute 'unitary portion':
      - $q_1 = F_1/(C_{11} + C_{12}) \Rightarrow F_{13} = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta} - ml\dot{\theta}^2$;
      - $q_2 = F_2/C_{11} \Rightarrow \cos\theta = \cos$;
      - $q_3 = F_3/C_{12} \Rightarrow \sin\Theta = \sin$;
- Step N:
  - Membrane Controller:
    - $\cos\theta = \cos$, $\sin\theta = \sin$, $F_{13} = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta} - ml\dot{\theta}^2$;
    - Compute productions function's value:
      - $F_1 = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \Rightarrow F_1 = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos - ml\dot{\theta}^2\sin$;
      - $F_2 = \cos\theta \Rightarrow F_2 = \cos$;

- $F_3 = \sin\theta \Rightarrow F_3 = \text{sin}$;
- Compute 'unitary portion':
  - $q_1 = F_1/(C_{11} + C_{12}) \Rightarrow F_{13} = (M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos - ml\dot{\theta}^2\,\text{sin}$;
  - $q_2 = F_2/C_{11} \Rightarrow \cos\theta = \text{cos}$;
  - $q_3 = F_3/C_{12} \Rightarrow \sin\Theta = \text{sin}$;

5 Conclusions and Future Work

In this paper, we study the use of the ENPS model in a control benchmark widely used in engineering and report our work-in-progress on the design of an efficient system able to control real-life pole balancing devices. Such design can be seen of a first approach to more complex control systems. One of the most important features of such control systems is the simplicity since they must provide an answer as soon as possible in order to effectively solve real-time problems. In this first approach, the solution is based on the mathematical approach known as *PID*, Proportional Integral Derivative, but other approaches are possible.

After completing the design, the immediate future work is to prove the designed NPS by integrating a NPS simulator as SNUPS [1] with a physics simulation environment as *Webots*. The experimental results will provide useful feedback in order to improve our design to make competitive with other control software.

A future second stage will be to generalize the design to 3D vehicles and check the design with the appropriate drone flight simulator.

Acknowledgements

MAGN acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

References

1. Buiu, C., Arsene, O., Cipu, C., Patrascu, M.: A software tool for modeling and simulation of numerical P systems. Biosystems 103(3), 442–447 (2011)
2. Buiu, C., Vasile, C., Arsene, O.: Development of membrane controllers for mobile robots. Information Sciences 187, 33–51 (2012)
3. Pavel, A., Arsene, O., Buiu, C.: Enzymatic numerical P systems - a new class of membrane computing systems. In: BIC-TA. pp. 1331–1336. IEEE (2010)
4. Pavel, A.B., Buiu, C.: Using enzymatic numerical P systems for modeling mobile robot controllers. Natural Computing 11(3), 387–393 (2012)
5. Pavel, A.B., Vasile, C.I., Dumitrache, I.: Robot Localization Implemented with Enzymatic Numerical P Systems. In: Prescott, T.J., Lepora, N.F., Mura, A., Verschure, P.F.M.J. (eds.) Living Machines. Lecture Notes in Computer Science, vol. 7375, pp. 204–215. Springer (2012)

6. Păun, Gh.: Some quick research topics., In these proceedings.
7. Păun, Gh., Păun, R.A.: Membrane computing and economics: Numerical P systems. Fundamenta Informaticae 73(1-2), 213–227 (2006)
8. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)
9. Vasile, C.I., Pavel, A.B., Dumitrache, I., Kelemen, J.: Implementing obstacle avoidance and follower behaviors on koala robots using numerical P systems. In: García-Quismondo, M., Macías-Ramos, L.F., Păun, Gh., Valencia-Cabrera, L. (eds.) Tenth Brainstorming Week on Membrane Computing. vol. II, pp. 215–227. Fénix Editora, Sevilla, Spain (2012)
10. Vasile, C.I., Pavel, A.B., Dumitrache, I., Păun, Gh.: On the power of enzymatic numerical P systems. Acta Informatica 49(6), 395–412 (2012)
11. Vasile, C.I., Pavel, A.B., Dumitrache, I.: Universality of enzymatic numerical p systems. International Journal of Computer Mathematics 90(4), 869–879 (2013)
12. Ana Brânduçsa Pavel, Cristian Ioan Vasile and Ioan Dumitrach: Robot Localization Implemented with Enzymatic Numerical P Systems. Living Machines 2012, 204-215, 2012