**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 1–1

# Editorial

Dear readers of *Theory and Applications of Mathematics & Computer Science,*

We are pleased to launch a scientific journal in the field of the exact sciences, **Theory and Applications of Mathematics & Computer Science (TAMCS)**. It is a direct follower of the Annals of "Aurel Vlaicu" University of Arad, Series: Mathematics and Computer Science (founded in 1992), yet aiming at a wider impact in the scientific community. Those were volumes comprising contributions presented by researchers in mathematics and informatics during the last two decades at the seven Scientific Communication Meetings held at the "Aurel Vlaicu" University of Arad (1992 - 2004). Meanwhile, these Annals made a step forward by gaining a more prestigious dimension, as the Scientific Communication Meeting became the International Symposium "Research and Education in Innovation Era" organized by "Aurel Vlaicu" University of Arad (2006, 2008 and 2010). One of the main symposium sections, that of Mathematics and Computer Science, involved a lot of distinguished participants from Romania and abroad, whose oral presentations were published as original scientific papers in three volumes of "Proceedings of the International Symposium Research and Education in Innovation Era".

Therefore, **TAMCS** is neither a newcomer on the stage of scientific production, nor a mere local enterprise. Although it unfolds here its first issue, one can state that it has a tradition. Belonging to the Faculty of Exact Sciences, it covers a large area of topics; the **TAMCS** journal publishes original papers of high scientific value in all domains of pure and theoretical applied mathematics, and computer science, with a special preference for the domains of interest represented by the members of the editorial board.

Occasionally, special thematic issues will be devoted to topics of particular interest; proposal for such issues are invited.

The **TAMCS** is a peer-reviewed online journal and is published with a volume yearly, each volume consisting of two distinct issues (usually the issues are scheduled to appear in Spring and Autumn, respectively).

I would like to thank all the editors that have contributed to the success of the journal.

**Editor-in-Chief**

**Sorin Nădăban**

"Aurel Vlaicu" University of Arad

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 2–10

# Stability Analysis Results Concerning the Fuzzy Control of a Class of Nonlinear Time-Varying Systems

Radu-Emil Precup[a,*], Emil M. Petriu[b], Claudia-Adina Dragoş[a], Radu-Codruţ David[a]

[a] *Politehnica University of Timişoara, Department of Automation and Applied Informatics, Bd. V. Parvan 2, RO-300223 Timişoara, Romania.*
[b] *University of Ottawa, School of Information Technology and Engineering, 800 King Edward, Ottawa, Ontario, Canada, K1N 6N5.*

**Abstract**

The paper offers new stability results concerning the Takagi-Sugeno (T-S) fuzzy control systems dedicated to a class of Single Input-Single Output (SISO) nonlinear time-varying systems. Lyapunov's approach based on quadratic positive definite Lyapunov function candidates is emplyed to derive a sufficient uniform asymptotic stability condition. An illustrative example validates the stability analysis results by the design of a T-S fuzzy control system for a SISO nonlinear process.

*Keywords:* Lyapunov function candidates, nonlinear time-varying systems, Takagi-Sugeno fuzzy control systems, uniform asymptotic stability.
*2000 MSC:* 93C42, 93C10, 37M05.

## 1. Introduction

Fuzzy controller (FCs) prove to be useful for processes which are subjected to difficulties in deriving mathematical models or coping with performance limitations. These nonlinear controllers are easily understandable version of other complex nonlinear controllers used in such situations (Johanyák & Kovács, 2006), (Johanyák *et al.*, 2006), (Škrjanc *et al.*, 2004), (Vaščák, 2008), (Vaščák, 2009).

One of the major concerns of dynamical systems theory is the necessity to guarantee their stability therefore thorough stability analyses must be conducted. The importance of this problem increases when a control system is involved. Although fuzzy controllers have been proposed for a long time and applied successfully in many applications (Zhao *et al.*, 2010), (Kruszewski *et al.*, 2008), (Chin-Tzong & Yung-Yih, 2008), (Precup *et al.*, 2009), (Haber *et al.*, 2010), (Kurnaz *et al.*, 2010), (Precup *et al.*, 2010) a comprehensive work on the proof of stability for fuzzy control systems has begun only recently. These systems have resulted in convenient and relatively easily understandable approaches to nonlinear control of complex and even ill-defined processes. A proof of stability in the conditions of all possible automatic control system operating conditions is necessary before the fuzzy control system is put into real practice.

In principle, for the stability analysis of fuzzy control systems controlling nonlinear processes any method can be utilized to be suitable for the analysis of nonlinear dynamical systems. Which method is the best one to use depends only on the prerequisites. There, the structure of the system, the type of information describing the process and the

*Corresponding author
*Email addresses:* radu.precup@aut.upt.ro (Radu-Emil Precup), petriu@site.uottawa.ca (Emil M. Petriu), claudia.dragos@aut.upt.ro (Claudia-Adina Dragoş), davidradu@gmail.com (Radu-Codruţ David)

type of sufficient conditions for the stability are usually the key points.

Many stability analyses become difficult to apply when there are developed fuzzy controllers to cope with complex processes including the nonlinear time-varying (LTV) ones. LTV systems are used in practice because most real-world systems are time-varying as a result of their parametric modifications over time as it is reported in (Neerhoff & van der Kloet, 2001), (van der Kloet & Neerhoff, 2002), (Yoneyama, 2007). LTV systems may also be a result of linearizing nonlinear systems in the vicinity of a set of operating points or of a trajectory. Several techniques are employed in the analysis and development of control systems meant for LTV systems. These techniques deal mainly with the popular eigenstructure assignment (Choi *et al.*, 2001), (Lee & Jiang, 2005).

Several methods for the stable design of FCs employing the stability analysis in the framework of considering the fuzzy control systems as nonlinear systems have been proposed recently. Some critical points of view on fuzzy controllers and on the stability analysis of fuzzy control systems are presented in (Michels *et al.*, 2006), (Arino & Sala, 2007), (Mozelli *et al.*, 2009), (Feng, 2010), (Lendek *et al.*, 2010).

The first objective of the paper is to give a mathematical characterization of a class of Takagi-Sugeno (T-S) models for Single Input-Single Output (SISO) nonlinear time-varying (NTV) systems viewed as controlled processes. On the basis of these models the second objective is to propose a stability analysis for the accepted class of control systems with T-S fuzzy controllers controlling the SISO NTV processes. The contribution of this paper with respect to (Tomescu *et al.*, 2008) concerns the new details inserted in the theoretical part while an extended set of digital simulation results is included.

The paper is organized as follows. Section 2 expresses the T-S fuzzy models for SISO NTV systems. Section 3 presents the new stability analysis results expressed as an original theorem based on Lyapunov's theorem for time-varying systems. Details on the Lorenz chaotic systems expressed in terms of the well known and accepted three equations are presented as case study in Section 4. Section 5 designs a stable fuzzy control system for the Lorenz chaotic system considered as illustrative example of NTV systems. The conclusions are outlined in Section 6.

## 2. Definition of a class of fuzzy control systems

In this paper the fuzzy control system is accepted to consist of a process and a T-S FC as shown in Fig. 1. The process of extracting the knowledge from human operators in the form of fuzzy control rules is by no means trivial, nor is the process of deriving the rules based on heuristics and good understanding of the process and control systems theory is needed.

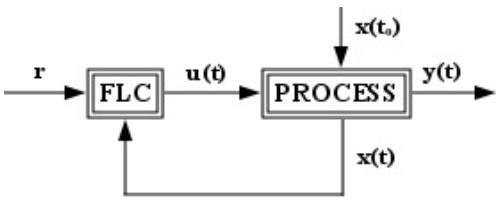


Figure 1. Fuzzy control system structure.

Let $D \subset \mathbb{R}^n$ be a universe of discourse. Consider the nonlinear autonomous system of the following form representing the state-space equations of the controlled process:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) + \mathbf{b}(\mathbf{x}, t)u \tag{2.1}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0$$

where: $\mathbf{x} \in D$, $\mathbf{x} = [\ x_1 \quad x_2 \quad ... \quad x_n\ ]^T$, is the state vector, $n \in \mathbb{N}^*$, $\dot{\mathbf{x}} = [\ \dot{x}_1 \quad \dot{x}_2 \quad ... \quad \dot{x}_n\ ]^T$ is the derivative of $\mathbf{x}$ with respect to the time variable $t$, $f, b : [0, \infty) \times D \to IR^n$ are continuous in $t$, and:

$$\mathbf{f}(\mathbf{x}, t) = [f_1\ (\mathbf{x}, t) \quad f_2(\mathbf{x}, t) \quad ... \quad f_n(\mathbf{x}, t)\ ]^T \tag{2.2}$$

and

$$\mathbf{b}(\mathbf{x}, t) = [\ b_1(\mathbf{x}, t) \quad b_2(\mathbf{x}, t) \quad ... \quad b_n(\mathbf{x}, t)\ ]^T \tag{2.3}$$

are functions describing the dynamics of the process, $u$ is the control signal fed to the process, obtained by the weighted-sum defuzzification method for T-S FCs.

The FC consists of $r$ fuzzy rules. The *i-th* IF-THEN rule in the fuzzy rule base of the FC, referred to as Takagi-Sugeno fuzzy rule, is expressed in terms of the following form:

$$\begin{aligned} &\textit{Rule } i : \text{IF } x_1 \text{ IS } X_{i,1} \text{ AND } x_2 \text{ IS } X_{i,2} \text{ AND } ... \text{ AND } x_n \text{ IS } X_{i,n} \\ &\text{THEN } u = u_i(x),\ i = \overline{1, r}, r \in \mathbb{N}^*, \end{aligned} \tag{2.4}$$

where $r$ is the total number of rules, $X_{i,1}, X_{i,2}, .., X_{i,n}$ are fuzzy sets that describe the linguistics terms (LTs) of input variables, $u = u_i(x)$ is the control signal of rule $i$, similar to the case of parallel distributed compensation, and the function AND is a t-norm. $u_i$ can be a single value or a function of the state vector, $\mathbf{x}(t)$.

Each fuzzy rule generates an activation degree referred to also as firing strength and defined in (2.5):

$$\alpha_i \in [0, 1],\ i = \overline{1, r},$$

$$\alpha_i(\mathbf{x}) = AND(\mu_{\tilde{X}_{i,1}}(x_1), \mu_{\tilde{X}_{i,2}}(x_2), ..., \mu_{\tilde{X}_{i,n}}(x_n)). \tag{2.5}$$

It is assumed that for any $\mathbf{x}$ belonging to the input universe of discourse, , there exists at least one $\alpha_i$ among all rules that is not equal to zero.

The control signal $u$, which must be applied to the process, is a function of $\alpha_i$ and $u_i$. By applying the weighted-sum defuzzification method, the output of the FC is given by (2.6):

$$u = \frac{\sum_{i=1}^{r} \alpha_i u_i}{\sum_{i=1}^{r} \alpha_i}. \tag{2.6}$$

## 3. Stability analysis of Takagi-Sugeno fuzzy control systems

The stability analysis theorem presented here is based on the well acknowledged Lyapunov's theorem for time-varying systems referred in (Slotine & Li, 1991), (Khalil, 2001). The theorem ensures sufficient stability conditions for the fuzzy control systems with the structure described in the previous section. This section is focused on Theorem 3.1 that can be expressed in terms of a stability analysis algorithm.

Let $V : D \times [0, \infty) \to \mathbb{R}$, $V(\mathbf{x}, t) = \mathbf{x}^T P \mathbf{x} \cdot g(t)$, where $P$ is an $n \times n$ constant positive definite matrix and $g \geq 0, \forall t \geq 0$, a continuously-differentiable function. The time derivative of $V(\mathbf{x}, t)$ along the open-loop trajectory (3.6) is given by:

$$\dot{V}(\mathbf{x}, t) = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial x}\dot{\mathbf{x}} = \mathbf{x}^T P \mathbf{x} \cdot \dot{g}(t) + (\dot{\mathbf{x}}^T P \mathbf{x} + \mathbf{x}^T P \dot{\mathbf{x}}) g(t) =$$

$$\mathbf{x}^T P \mathbf{x} \cdot \dot{g}(t) + g(t)(f(\mathbf{x}, t) + b(\mathbf{x}, t)u)^T P x + g(t) x^T P (f(\mathbf{x}, t) + b(\mathbf{x}, t)u) =$$

$$F(\mathbf{x}, t) + B(\mathbf{x}, t)u, \tag{3.1}$$

where:

$$F(\mathbf{x}, t) = g(t) f(\mathbf{x}, t)^T P \mathbf{x} + g(t) \mathbf{x}^T P f(\mathbf{x}, t) + \mathbf{x}^T P \mathbf{x} \cdot \dot{g}(t) \tag{3.2}$$

and

$$B(\mathbf{x},t) = g(t)b(\mathbf{x},t)^T P\mathbf{x} + g(t)\mathbf{x}^T Pb(\mathbf{x},t). \tag{3.3}$$

The time derivative of $V(\mathbf{x})$ along the trajectory (2.1) for $\mathbf{u} = \mathbf{u}_k(\mathbf{x})$ is

$$\dot{V}_k(\mathbf{x},t) = F(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}_k(\mathbf{x}) \tag{3.4}$$

The next theorem is based on the Lyapunov's theorem for time-varying systems (Khalil, 2001) and (Slotine & Li, 1991). Its proof is presented in (Precup *et al.*, 2009).

**Theorem 3.1.** *Consider the fuzzy control system consisting of the T-S FC defined in Section 2 and the nonlinear time-varying process with the state-space equations (2.1). Let* $\mathbf{x} = \mathbf{0}$ *be an equilibrium point for (2.1) and* $D \subset \mathbb{R}^n$ *be a domain containing the origin* $\mathbf{x} = \mathbf{0}$*. Let the Lyapunov function candidate* $V : D \times [0,\infty) \to \mathbb{R}$, $V(\mathbf{x},t) = \mathbf{x}^T P\mathbf{x} \cdot g(t)$ *where* $P$ *is an* $n \times n$ *constant positive definite matrix and* $g \geq 0, \forall t \geq 0$ *a continuously-differentiable function, such that:*

$$V(\mathbf{x},t) \leq W^1(\mathbf{x}) \tag{3.5}$$

$$\dot{V}_k(\mathbf{x},t) = F(\mathbf{x},t) + B(\mathbf{x},t)\mathbf{u}_k(\mathbf{x}) \leq -W_k^2(\mathbf{x}), \forall k = \overline{1,r}, \forall t \geq 0, \forall \mathbf{x} \in D, \tag{3.6}$$

*where* $W^1(\mathbf{x})$ *and* $W_k^2(\mathbf{x})$ *are continuous positive definite functions on* $D$*. Then* $\mathbf{x} = 0$ *is uniformly asymptotically stable.*

The above stability theorem ensures sufficient stability conditions concerning the considered class of T-S fuzzy control systems described briefly in Section 2.

## 4. Case study

This section presents preliminary information about the Lorenz chaotic system known also as the Lorenz attractor (Lorenz, 1963), (Lorenz, 1993). In the first scientific papers about concept of deterministic chaos, chaotic behavior has been regarded as an exotic phenomenon that might be of interest only as a mathematical speculation and would never be encountered in practice. The Lorenz equation is commonly defined as three coupled ordinary differential equations expressed in (4.1) to model the convective motion of fluid cell, which is warmed from below and cooled from above:

$$\begin{array}{l} \frac{dx}{dt} = \sigma(y-x) \\ \frac{dy}{dt} = x(\rho - z) - y \\ \frac{dz}{dt} = xy - \beta z \end{array} \tag{4.1}$$

where the three parameters $\sigma, \rho, \beta > 0$ are called the Prandtl number, the Rayleigh number, and a physical proportion, respectively. These constants determine the behavior of the system and these three equations exhibit chaotic behavior i.e. they are extremely sensitive to initial conditions. A small change of the initial conditions leads quickly to large modifications of the corresponding solutions. The classical values used to demonstrate chaos are $\sigma = 10, \beta = \frac{8}{3}$ and $\rho$ is variable in time.

## 5. Stable design of fuzzy control systems

The design of the fuzzy control system with T-S FC starts with rewriting the ordinary differential equation (4.1) in the following form representing the state-space equations of the nonlinear time-varying controlled process:

$$\dot{\mathbf{x}} = \begin{pmatrix} \sigma(x_2 - x_1) \\ x_1(\rho(t) - x_3) - x_2 \\ x_1 x_2 - \beta x_3 \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} u, \quad x(t_0) = x_0 \tag{5.1}$$

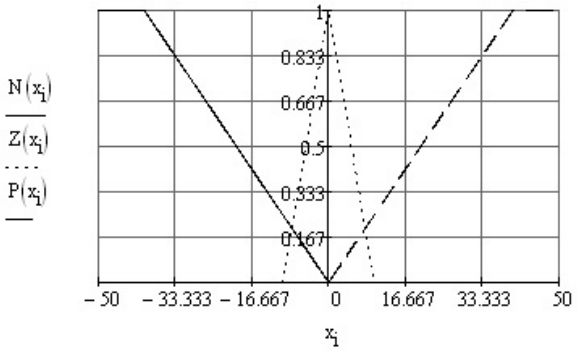


Figure 2. Membership functions of $x_1$ and $x_2$ .

where $\sigma = 10, \beta = \frac{8}{3}$ and $\rho(t) \in (0, 100]$. Next, the fuzzification module of T-S FC is set according to Fig. 2 which illustrates the membership functions that describe the LTs of the linguistic variables of $x_1$ and $x_2$. The linguistics terms representing Positive, Zero and Negative values are highlighted by P, Z and N, respectively.

The inference engine employs the fuzzy logic operators AND and OR implemented by the MIN and MAX functions, respectively, as pointed out in Section 2. The inference engine is assisted by the complete set of fuzzy control rules illustrated in Table 1, and the weighted sum defuzzification method is utilized. Summarizing, the only parameters to be calculated are the consequents $u_i$, $i = \overline{1,9}$, in the nine fuzzy control rules.

In order to find the values of $u_i$ for which the system (5.1) can be stabilized with the above described T-S FC, we will apply Theorem 3.1.

Table 1
Fuzzy Control Rule Base

| Rule | Antecedent | | Consequent |
|---|---|---|---|
| | $x_1$ | $x_2$ | $u$ |
| 1 | P | P | $u_1$ |
| 2 | N | N | $u_2$ |
| 3 | P | N | $u_3$ |
| 4 | N | P | $u_4$ |
| 5 | P | Z | $u_5$ |
| 6 | N | Z | $u_6$ |
| 7 | Z | P | $u_7$ |
| 8 | Z | N | $u_8$ |
| 9 | Z | Z | $u_9$ |

Let the universe of discourse be $D = (-100, 100] \times (-100, 100]$. Let us consider the next Lyapunov function:

$$V(\mathbf{x}, t) = (x_1^2 + x_2^2 + x_3^2)(1 + e^{-t}) \leq 2(x_1^2 + x_2^2 + x_3^2) = W^1(\mathbf{x}) \tag{5.2}$$

which is a continuously differentiable positive function on domain D. The total derivative of V with respect to time using (5.1) is:

$$\dot{V}(\mathbf{x}, t) = [-2(\sigma x_1^2 + x_2^2 + \beta x_3^2) + 2x_1 x_2(\sigma + \rho(t)) + 2x_1 u](1 + e^{-t}) -$$

$$-e^{-t}(x_1^2 + x_2^2 + x_3^2). \tag{5.3}$$

After analyzing each fuzzy control rule, in respect to Theorem1, results:

**For rule 1:** $u_1 = -100(\sigma + \rho(t))$.

**For rule 2:** $u_2 = 100(\sigma + \rho(t))$.

**For rule 3:** $u_3 = 0$.

**For rule 4:** $u_4 = 0$.

**For rule 5:** $u_5 = -10(\sigma + \rho(t))$.

**For rule 6:** $u_6 = 10(\sigma + \rho(t))$.

**For rule 7:** $u_7 = -x_2(\sigma + \rho(t))$.

**For rule 8:** $u_8 = -x_2(\sigma + \rho(t))$.

**For rule 9:** $u_9 = -x_2(\sigma + \rho(t))$.

We choose

$$W^2(\mathbf{x}) = W_k^2(\mathbf{x}) = (\sigma x_1^2 + x_2^2 + \beta x_3^2) \tag{5.4}$$

for any $u_k$.

Considering the values of process parameters $\sigma = 10, \beta = \frac{8}{3}, \rho(t) \in (0, 100]$, the initial state $x_1(0) = 1$, $x_2(0) = -1$ and $x_3(0) = 1$, the responses of $x_1$, $x_2$ and $x_3$ versus time in the closed-loop system are shown in Figs. 3-6. But, as already mentioned, the simulation of the behavior of chaotic system is extremely important due to the initial conditions of the system of ordinary differential equations (3.6). Therefore the Adams method has been employed here in the numerical solving (integration) of the ordinary differential equations afferent to the mathematical model of the fuzzy control system (4.1). The mathematical model is obtained by coupling the equations (2.1)-(2.4) and (5.3).

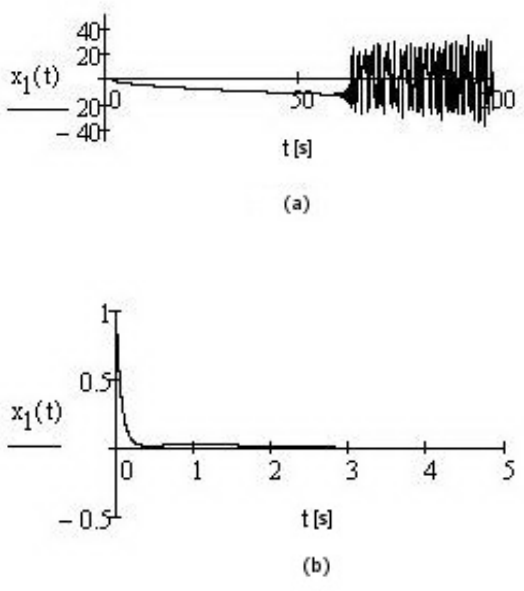


Figure 3. State variable $x_1$ versus time of Lorenz chaotic without FC (a) and with FC (b).

## 6. Conclusion

New results concerning the uniform asymptotic stability analysis of T-S fuzzy control systems dedicated to a class of nonlinear processes characterized by SISO nonlinear time-varying systems have been introduced. The theoretical results have been applied resulting in a simple and effective Takagi-Sugeno FC viewed as an alternative to the stabilization of the Lorenz chaotic system.

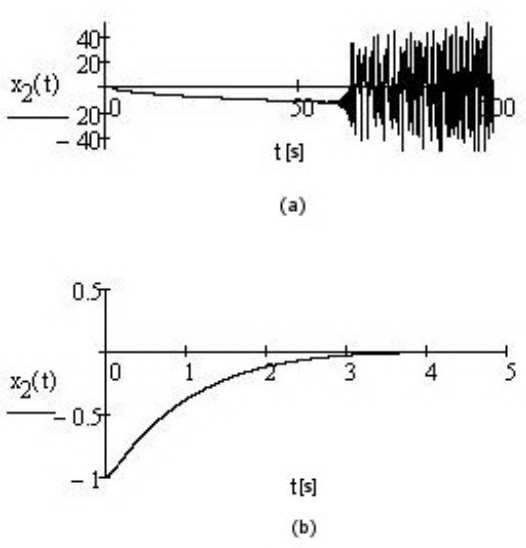


Figure 4. State variable $x_2$ versus time of Lorenz chaotic without FC (a) and with FC (b).

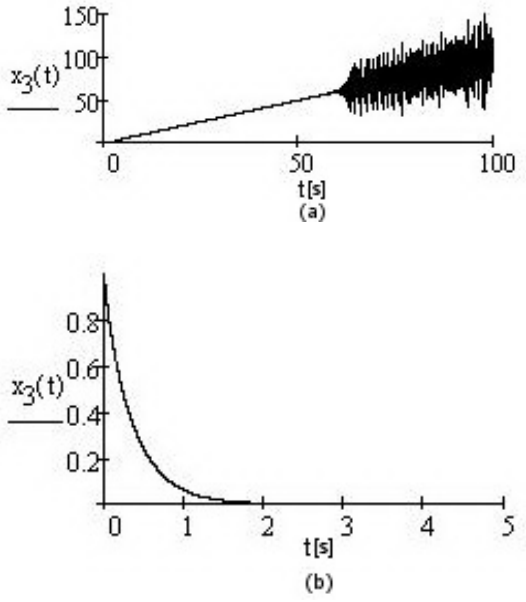


Figure 5. State variable $x_3$ versus time of Lorenz chaotic without FC (a) and with FC (b).

The stability analysis results proposes in this paper can be extended and applied in situations when the system has an equilibrium point different to the origin and / or the reference input r in Figure 1 is nonzero by an appropriately defined state transform. The new stability analysis results are different to the original Lyapunov's theorem and it is also different to the variety of more or less conservative LMI-based results. It allows more applications and it is well suited for controlling processes where the Lyapunov function candidates are not quadratic and the derivatives of the Lyapunov function candidates are negative definite. Therefore the applications of Theorem 3.1 to nonlinear processes controlled by T-S FCs will be successful for wide area of nonlinear time-varying systems. However the transparency of the proposed approach is not so good in comparison with that of LMI-based approaches.

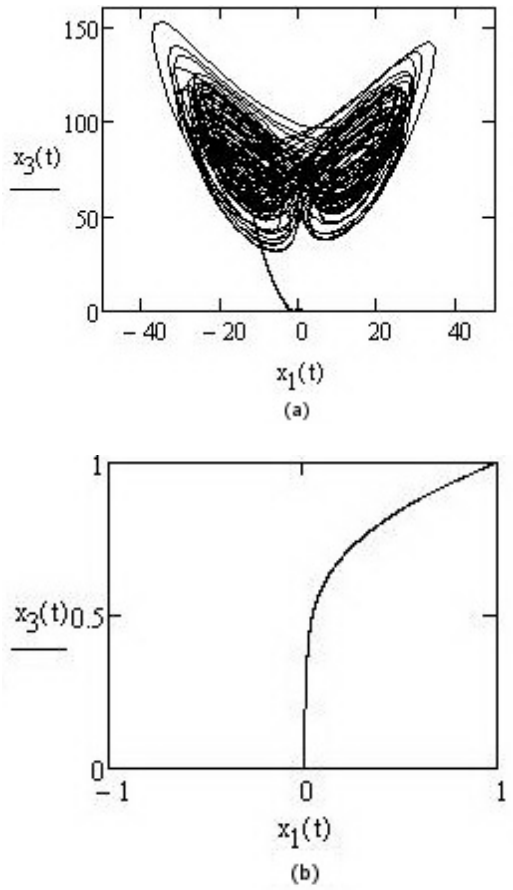


Figure 6. Phase portrait of Lorenz system without control (a) and with FC (b).

The digital simulation results presented in Section 5 prove that the proposed stability analysis approach is simpler than the nonfeedback control solutions proposed by Lima and Pettini (Lima & Pettini, 1990), the OGY method proposed by Ott, Grebogi and Yorke (Ott *et al.*, 1990) and Pyragas's method (Pyragas, 1992). The suggested T-S FC structure can be implemented as low cost automation solution.

Further research will be dedicated to offering other low cost fuzzy solutions for chaotic systems based on similar approaches and applications [(Horváth & Rudas, 2004), (Škrjanc *et al.*, 2005), (Johanyák & Kovács, 2007), (Precup, 2007), (Precup *et al.*, 2008)]. The careful stability analysis is necessary in all applications aiming the analysis of the stablity analysis algorithms by giving correct estimates of their complexity.

## References

Arino, C. and A. Sala (2007). Relaxed LMI conditions for closed-loop fuzzy systems with tensor-product structure. *Engineering Applications of Artificial Intelligence* **20**(8), 1036–1046.

Chin-Tzong, P. and L. Yung-Yih (2008). On the stability of takagi-sugeno fuzzy systems with time-varying uncertainties. *IEEE Transactions on Fuzzy Systems* **16**(1), 162–170.

Choi, J.W., H.C. Lee and J.J. Zhu (2001). Decoupling and tracking control using eigenstructure assignment for linear time-varying systems. *International Journal of Control* **74**(5), 453–464.

Feng, G. (2010). *Analysis and Synthesis of Fuzzy Control Systems: A Model-Based Approach*. CRC Press: Boca Raton, FL.

Haber, R.E., R.M. del Toro and A. Gajate (2010). Optimal fuzzy control system using the cross-entropy method. a case study of a drilling process. *Information Sciences* **18**(14), 2777–2792.

Horváth, L. and I.J. Rudas (2004). *Modelling and Solving Methods for Engineers*. Elsevier, Academic Press, Burlington, MA.

Johanyák, Z.C. and S. Kovács (2006). *Fuzzy rule interpolation based on polar cuts, Computational Intelligence, Theory and Applications*. Springer Verlag, Berlin, Heidelberg, New York. Bernd Reusch (Ed.), pp. 499-511.

Johanyák, Z.C. and S. Kovács (2007). Sparse fuzzy system generation by rule base extension. In: *Proceedings of 11th IEEE International Conference of Intelligent Engineering Systems (INES 2007), Budapest, Hungary*. pp. 99–104.

Johanyák, Z.C., D. Tikk, S. Kovács and K. K. Wong (2006). Fuzzy rule interpolation Matlab toolbox - FRI toolbox. In: *Proceedings of the IEEE World Congress on Computational Intelligence (WCCI'06), 15th International Conference on Fuzzy Systems (FUZZ-IEEE'06), Vancouver, BC, Canada,*. pp. 1427–1433.

Khalil, H.K. (2001). *Nonlinear Systems*. Prentice-Hall. Englewood Cliffs, NJ. 3rd Edition.

Kruszewski, A., R. Wang and T.M. Guerra (2008). Nonquadratic stabilization conditions for a class of uncertain nonlinear discrete time ts fuzzy models: A new approach. *IEEE Transactions on Automatic Control* **53**(2), 606–611.

Kurnaz, S., O. Cetin and O. Kaynak (2010). Adaptive neuro-fuzzy inference system based autonomous flight control of unmanned air vehicles. *Expert Systems with Applications* **37**(2), 1229–1234.

Lee, T.-C. and Z.-P.A. Jiang (2005). A generalization of Krasovskii-LaSalle theorem for nonlinear time-varying systems: Converse results and applications. *IEEE Transactions on Automatic Control* **50**(8), 1147–1163.

Lendek, Z., J. Lauber, T.M. Guerra, R. Babuška and B. De Schutter (2010). Adaptive observers for TS fuzzy systems with unknown polynomial inputs. *Fuzzy Sets and Systems* **161**(15), 2043–20965.

Lima, R. and M. Pettini (1990). Suppression of chaos by resonant parametric perturbations. *Physical Review A* **41**(2), 726–733.

Lorenz, E.N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences* **20**(2), 130–141.

Lorenz, E.N. (1993). *The Essence of Chaos*. University of Washington Press. Seattle, WA.

Michels, K., F. Klawonn, R. Kruse and A. Nrnberger (2006). *Fuzzy Control: Fundamentals, Stability and Design of Fuzzy Controllers*. Springer Verlag. Berlin, Heidelberg, New York.

Mozelli, L.A., R.M. Palhares, F.O. Souza and E.M.A.M. Mendes (2009). Reducing conservativeness in recent stability conditions of TS fuzzy systems. *Automatica* **45**(6), 1580–1583.

Neerhoff, F.L. and P. van der Kloet (2001). On the factorization of the system operator of scalar linear time-varying systems. In: *Proceedings of the ProRISC / IEEE Workshop on Semiconductors, Circuits, Systems and Signal Processing, Veldhoven, The Netherlands*. pp. 516–519.

Ott, E., C. Grebogi and J.A. Yorke (1990). Controlling chaos. *Physical Review Letters* **64**(11), 1196–1199.

Precup, R.-E. (2007). *Computer Assisted Mathematics. Algorithms*. Editura Orizonturi Universitare. Timişoara. in Romanian: Matematici asistate de calculator. Algoritmuri.

Precup, R.-E., M.-L. Tomescu, St. Preitl and E.M. Petriu (2009). Fuzzy logic-based stabilization of nonlinear time-varying systems. *International Journal of Artificial Intelligence* **3**(A09), 24–36.

Precup, R.-E., M.-L. Tomescu, St. Preitl and E.M. Petriu (2010). Fuzzy logic-based stabilization of a magnetic ball suspension system. *International Journal of Artificial Intelligence* **5**(10), 56–66.

Precup, R.-E., S. Preitl, J.K. Tar, M. Takács M.L. Tomescu, P. Korondi and P. Baranyi (2008). Fuzzy control system performance enhancement by iterative learning control. *IEEE Transactions on Industrial Electronics* **55**(9), 3461–3475.

Pyragas, K. (1992). Continuous control of chaos by self-controlling feedback. *Physics Letters A* **170**(6), 421–427.

Škrjanc, I., S. Blažič and O.E. Agamennoni (2005). Identification of dynamical systems with a robust interval fuzzy model. *Automatica* **41**, 327–332.

Škrjanc, I., S. Blažič, S. Oblak and J. Richalet (2004). An approach to predictive control of multivariable time-delayed plant: Stability and design issues. *ISA Transactions* **43**(4), 585–595.

Slotine, J.-J. and W. Li (1991). *Applied Nonlinear Control*. Prentice-Hall. Englewood Cliffs, NJ.

Tomescu, M.L., R.-E. Precup, S. Preitl and S. Blažič (2008). Stable fuzzy control of a category of nonlinear time-varying systems. In: *Proceedings of the international Symposium. Research and Education in Innovation Era, 2nd Edition, Romania, Arad, 20-21 November 2008 Section Mathematics and Computer Science*. pp. 221–233.

van der Kloet, P. and F.L. Neerhoff (2002). Dynamic eigenvalues for scalar linear time-varying systems. In: *Proceedings of the 15th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2002), August 12-16, Notre Dame, IN, 2002*. pp. CD–ROM, paper index 14423, 8 pp.

Vaščák, J. (2008). Fuzzy cognitive maps in path planning. *Acta Technica Jaurinensis, Series Intelligentia Computatorica, Széchenyi István University, Györ, Hungary* **1**(3), 467–479.

Vaščák, J. (2009). Using neural gas networks in traffic navigation. *Acta Technica Jaurinensis, Series Intelligentia Computatorica, Széchenyi István University, Györ, Hungary* **2**(2), 203–215.

Yoneyama, J. (2007). Robust stability and stabilization for uncertain takagi-sugeno fuzzy time-delay systems. *Fuzzy Sets and Systems* **158**(2), 115–134.

Zhao, Z.-Y., W.-F. Xie and H. Hong (2010). Hybrid optimization method of evolutionary parallel gradient search. *International Journal of Artificial Intelligence* **5**(A08), 1–16.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 11–16

# Classification Accuracy of Neural Networks with PCA in Emotion Recognition

Jasmina Novakovic[a,*], Milomir Minic[a], Alempije Veljovic[b]

[a] *Faculty of Public Administration, Megatrend University, Belgrade, Serbia.*
[b] *Technical Faculty Cacak, University in Kragujevac, Cacak, Serbia.*

## Abstract

This paper presents classification accuracy of neural network with principal component analysis (PCA) for feature selections in emotion recognition using facial expressions. Dimensionality reduction of a feature set is a common preprocessing step used for pattern recognition and classification applications. PCA is one of the popular methods used, and can be shown to be optimal using different optimality criteria. Experiment results, in which we achieved a recognition rate of approximately 85% when testing six emotions on benchmark image data set, show that neural networks with PCA is effective in emotion recognition using facial expressions.

*Keywords:* emotion recognition, feature selection, neural network, PCA.
*2000 MSC:* 68T45, 97P20, 97R40, 68T45.

## 1. Introduction

Feature selection is an active field in computer science. It has been a fertile field of research and development since 1970's in statistical pattern recognition (Wyse *et al.*, 1980), (Ben-Bassat, 1982), (Siedlecki & Sklansky, 1988), machine learning and data mining (Blum & Langley, 1997), (Dash & Liu, 1997), (Dy & Brodley, 2000), (Kim *et al.*, 2000), (Das, 2001), (Mitra *et al.*, 2002). Feature selection is a fundamental problem in many different areas, especially in forecasting, document classification, bioinformatics, and object recognition or in modeling of complex technological processes. Data sets with thousands of features are common in such applications. For some problems, all features may be important, but for some target concept, only a small subset of features is usually relevant.

Feature selection reduces the dimensionality of feature space, removes redundant, irrelevant, or noisy data. It brings the immediate effects for application: speeding up a data mining algorithm, improving the data quality and thereof the performance of data mining, and increasing the comprehensibility of the mining results.

Feature selection can be defined as a process that chooses a minimum subset of M features from the original set of N features, so that the feature space is optimally reduced according to a certain evaluation criterion. As the dimensionality of a domain expands, the number of feature N increases. Finding the best feature subset is usually intractable (Kohavi & John, 1997) and many problem related to feature selection have been shown to be NP-hard (Blum & Rivest, 1992).

In pattern recognition and general classification problems, methods such as PCA, independent component analysis (ICA) and Fisher linear discriminate analysis have been extensively used. These methods find a mapping between the original feature space to a lower dimensional feature space.

*Corresponding author
*Email addresses:* jnovakovic@megatrend.edu.rs (Jasmina Novakovic), mminic@megatrend.edu.rs (Milomir Minic), alempije@beotel.rs (Alempije Veljovic)

The main aim of this paper is to experimentally verify, on benchmark image data set, the classification accuracy of neural network with PCA in emotion recognition. This paper is organized as follows. Section 2 contains the general approach to automatic facial expression analysis. Neural network as classifier in emotion recognition is presented in section 3. Section 4 contains general issues concerning PCA as feature selection and reduction method. Section 5 presents experimental evaluation. Final section contains discussion of the obtained results and some closing remarks.

## 2. Facial Expression Analysis

Facial expression analysis includes both measurement of facial motion and recognition of expression. The general approach to automatic facial expression analysis consists of three steps: face acquisition, facial data extraction and representation, and facial expression recognition.

The first step, face acquisition is a processing stage to automatically find the face region for the input images or sequences. It can be a detector to detect face for each frame or just detect face in the first frame and then track the face in the remainder of the video sequence. To handle large head motion, the head finder, head tracking, and pose estimation can be applied to a facial expression analysis system.

The next step is to extract and represent the facial changes caused by facial expressions. There are mainly two types of approaches in facial feature extraction for expression analysis: geometric feature-based methods and appearance-based methods. The geometric facial features present the shape and locations of facial components (including mouth, eyes, brows, and nose). The facial components or facial feature points are extracted to form a feature vector that represents the face geometry. With appearance-based methods, image filters, such as Gabor wavelets, are applied to either the whole-face or specific regions in a face image to extract a feature vector. Depending on the different facial feature extraction methods, the effects of in-plane head rotation and different scales of the faces can be eliminated by face normalization before the feature extraction or by feature representation before the step of expression recognition.

In the last step, the facial changes can be identified as facial action units or prototypic emotional expressions. The detection and processing of facial expression is achieved through various methods such as optical flow, hidden Markov model, neural network processing or active appearance model. More than one modalities can be combined or fused (multimodal recognition, e.g. facial expressions and speech prosody (Caridakis *et al.*, 2006) or facial expressions and hand gestures (Balomenos *et al.*, 2005)) to provide a more robust estimation of the subject's emotional state.

Table 1. Properties of an ideal facial expression analysis system (Tian *et al.*, 2001).

| **Robustness** |
|---|
| Deal with subjects of different age, gender, ethnicity |
| Handle lighting changes |
| Handle large head motion |
| Handle occlusion |
| Handle different image resolution |
| Recognize all possible expressions |
| Recognize expressions with different intensity |
| Recognize asymmetrical expressions |
| Recognize spontaneous expressions |
| **Automatic process** |
| Automatic face acquisition |
| Automatic facial feature extraction |
| Automatic expression recognition |
| **Real-time process** |
| Real-time face acquisition |
| Real-time facial feature extraction |
| Real-time expression recognition |
| **Autonomic Process** |
| Output recognition with confidence |
| Adaptive to different level outputs based on input images |

The properties of an ideal facial expression analysis system are summarized in Table 1.

## 3. Neural Networks

Neural networks have been studied for more than four decades since Rosenblatt first applied the *single-layer perceptrons* to pattern-classification learning in the late 1950's. A network structure consisting of a number of nodes connected through directional links is a neural network. In this network each node represents a processing unit, and the links between nodes specify the causal relationship between connected nodes. The outputs of these nodes depend on modifiable parameters pertaining to these nodes.

A neural network is a massive parallel distributed processor made up of simple processing units. This network has the ability to learn from experiential knowledge expressed through interunit connection strengths, and can make such knowledge available for use.

A neural network derives its computing power through its massive parallel distributed structure and its ability to learn and therefore to generalize. Generalization refers to the neural network producing reasonable outputs for new inputs not encountered during a learning process.

Fundamental to the operation of a neural network is a neuron as an information-processing unit. Figure 1 shows that a neuron consists of three basic elements: a set of connecting links, an adder and an activation function.

The first basic elements, a set of connecting links from different inputs $x_i$ (or synapses), is characterized by a weight or strength $w_{ki}$. The weights of a neuron may lie in a range that includes negative as well as positive values. The first index refers to the neuron in question and the second index refers to the input of the synapse to which the weight refers. The second basic elements, an adder, summing the input signals $x_i$ weighted by the respective synaptic strengths $w_{ki}$. An activation function $f$, as the third basic elements in neuron, limiting the amplitude of the output $y_k$ of a neuron.

An externally applied bias, denoted by $b_k$, is also includes in the model of the neuron shown in Figure 1. The effect of bias is increasing or lowering the net input of the activation function, depending on whether it is positive or negative.

The architecture of a neural network is defined by the characteristics of a node and the characteristics of the node's connectivity in the network. Network architecture is specified by the number of inputs to the network, the number of outputs, the total number of elementary nodes that are usually equal processing elements for the entire network, and their organization and interconnections. Generally, neural networks are classified, on the basis of the type of interconnections, into two categories: feedforward and recurrent.

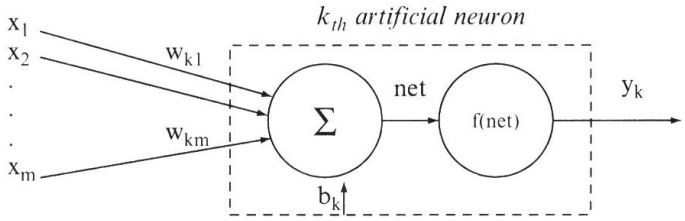


Figure 1. Model of a neuron.

If the processing propagates from the input side to the output side unanimously, without any loops or feedbacks, the network is feedforward. In a layered representation of the feedforward neural network, there are no links between nodes in the same layer; outputs of nodes in a specific layer are always connected as inputs to nodes in succeeding layers. This representation is preferred because of its modularity, i.e., nodes in the same layer have the same functionality or generate the same level of abstraction about input vectors. The network is recurrent if there is a feedback link that forms a circular path in a network (usually with a delay element as a synchronization component). In Figure 2 are shown examples of neural network belonging to both classes.

In both classes many neural-network models have been proposed, but the multilayer feedforward network with a backpropagation-learning mechanism, is the most widely used model in terms of practical applications.

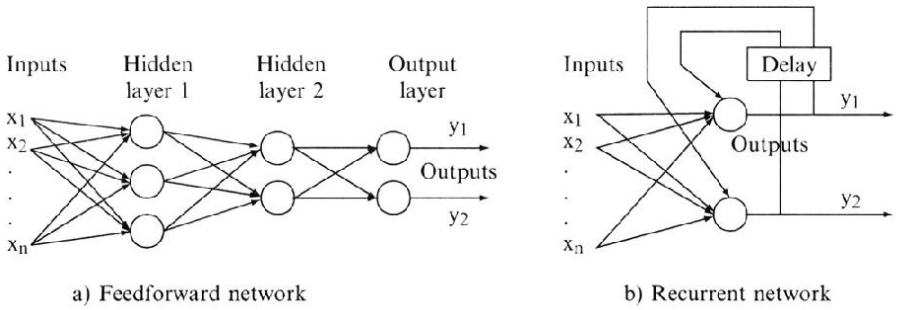


Figure 2. Typical architectures of neural networks.

## 4. PCA for Dimensionality Reduction

PCA, also known as Karhunen-Loeve transform, is one of the most popular techniques for dimensionality reduction. It is a standard statistical technique that can be used to reduce the dimensionality of a data set. PCA is useful tool for dimensionality reduction of multivariate data in image analysis, pattern recognition and appearance-based visual recognition, data compression, time series prediction, and analysis of biological data.

The strength of PCA for data analysis comes from its efficient computational mechanism, the fact that it is well understood, and from its general applicability. For example, a sample of applications in computer vision includes the representation and recognition of faces, recognition of 3D objects under varying pose, tracking of deformable objects, and for representations of 3D range data of heads.

PCA is a method of transforming the initial data set represented by vector samples into a new set of vector samples with derived dimensions. The basic idea can be described as follows: a set of $n$-dimensional vector samples $\mathrm{X} = \{x_1, x_2, x_3, \ldots, x_m\}$ should be transformed into another set $\mathrm{Y} = \{y_1, y_2, \ldots, y_m\}$ of the same dimensionality, but y-s has the property that most of their information content is stored in the first few dimensions. So, we can reduce the data set to a smaller number of dimensions with low information loss.

In implementation, the transformation from the original attributes to principal components is carried out through a process by first computing the covariance matrix of the original attributes and then, by extracting its eigenvectors to act as the principal components. The eigenvectors specify a linear mapping from the original attribute space of dimensionality $N$ to a new space of size $M$ in which attributes are uncorrelated. The resulting eigenvectors can be ranked according to the amount of variation in the original data that they account for. Typically, the first few transformed attributes account for most of the variation in the data set and are retained, while the remainder are discarded.

PCA is an unsupervised method which makes no use of information embodied within the class variable. Because, the PCA returns linear combinations of the original features, the meaning of the original features is not preserved.

However, PCA models have several shortcomings. One is that naive methods for finding the principal component directions have trouble with high dimensional data or large numbers of data points. Difficulties can arise in the form of computational complexity and also data scarcity. Another shortcoming of standard approaches to PCA is that it is not obvious how to deal properly with incomplete data set, in which some of the points are missing. To solve these drawbacks of standard PCA, a lot of methods were proposed in the field of statistics, computer engineering, neural networks etc.

Over the years there have been many extensions to conventional PCA. For example, Independent Component Analysis (ICA) is the attempt to extend PCA to go beyond decorrelation and to perform a dimension reduction onto a feature space with statistically independent variables. Other extensions address the situation where the sample data live in a low-dimensional (non-linear) manifold in an effort to retain a greater proportion of the variance using fewer components and yet other (related) extensions derive PCA from the perspective of density estimation (which facilitate modeling non-linearity in the sample data) and the use of Bayesian formulation for modeling the complexity of the sample data manifold.

## 5. Experimental Results

JAFFE database of facial expression images was used (available at http://www.kasrl.org/jaffe.html). Ten expressors posed 3 or 4 examples of each of the six basic facial expressions (happiness, sadness, surprise, anger, disgust, fear) and a neutral face for a total of 219 images of facial expressions. For simplicity of experimental design only Japanese female expressors were used. Sample images are shown in Figure 3.

Many classifiers have been applied to expression recognition such as neural network, support vector machines, linear discriminant analysis, k-nearest neighbor, multinomial logistic ridge regression, hidden Markov models, tree augmented naive Bayes, and others.

Some systems use only a rule-based classification based on the definition of the facial actions. Also, there are the expression recognition methods to frame-based and sequence-based expression recognition methods. The frame-based recognition method uses only the current frame with or without a reference image (mainly it is a neutral face image) to recognize the expressions of the frame. The sequence-based recognition method uses the temporal information of the sequences to recognize the expressions for one or more frames.

Figure 3. Sample images of JAFFE database.

For emotion recognition we used neural networks and PCA-based dimensionality reduction. Algorithm for facial expression recognition classify the given image into one of the seven basic facial expression categories (happiness, sadness, fear, surprise, anger, disgust and neutral). PCA is used for dimensionality reduction in input data. It retaining those characteristics of the data set that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. Because, low-order components contain the ”most important” aspects of the data. The extracted feature vectors in the reduced space are used to train the supervised neural network classifier. This approach does not require the detection of any reference point or node grid. The proposed method is fast and can be used for real-time applications.

Table 2. Confusion matrix for classification of facial expressions.
HAP - Happiness, SAD - Sadness, SUR - Surprise, ANG - Anger, DIS - Disgust, FEA - Fear.

| ANG | DIS | FEA | HAP | SAD | SUR | I/O |
|---|---|---|---|---|---|---|
| 9 | 1 | 0 | 0 | 0 | 0 | ANG |
| 0 | 11 | 1 | 0 | 1 | 0 | DIS |
| 0 | 0 | 9 | 1 | 0 | 0 | FEA |
| 0 | 0 | 0 | 9 | 2 | 0 | HAP |
| 0 | 0 | 1 | 2 | 8 | 0 | SAD |
| 0 | 0 | 1 | 0 | 0 | 10 | SUR |
| 0 | 0 | 0 | 0 | 0 | 0 | NEU |

The expression classifier was first tested using a set of images of expressions posed by ten Japanese females expressor initials: KA, KL, KM, KR, MK, NA, NM, TM, UY and YM. Each expresser posed three or four examples

of each of the six fundamental facial expressions and a neutral face. The image set was partitioned into ten segments, each corresponding to one expressor. Two facial expression images of each expression of each subject were randomly selected as training samples, while the remaining samples were used as test data. Not all expressions were equally well recognized by the system. Table 2 shows a confusion matrix showing misclassification rates for expressors. Our simulation experiment results show that neural networks is effective in emotion recognition using facial expressions, and we achieved a recognition rate of approximately 85% when testing six emotions.

It is not so convenient to compare categorization performance, because the problem that posed expressions is not always pure examples of a single expression category. It is important to realize that expression is never pure expressions of one emotion, but always admixtures of different emotions. The expression labels on the images in JAFFE database just represent the predominant expression in that image - the expression that the subject was asked to pose.

## 6. Conclusions and Ongoing Research

Experiment results with recognition rate of approximately 85% when testing six emotions on benchmark image data set, show that neural networks with PCA is effective in emotion recognition using facial expressions.

This research could help in future works, like capturing non-static images in real time and simultaneously analyzing these images according to affective computing techniques. By making these analyses some of the user's emotional states could be seen like joy, fear, angry, and with these probable results, assistants and computer optimizers could help users in the most different applications.

There are many questions and issues that remain to be addressed and that we intend to investigate in future work. Some improvements of the selecting methods presented here are possible. The algorithms and data sets will be selected according to precise criteria: classify algorithms and several data sets. These conclusions and recommendations will be tested on larger data sets using various classification algorithms in the near future.

## References

Balomenos, T., A. Raouzaiou, S. Ioannou, A. Drosopoulos, K. Karpouzis and S. Kollias (2005). Emotion analysis in man-machine interaction systems. In: *Machine Learning for Multimodal Interaction* (Samy Bengio and Hervé Bourlard, Eds.). Vol. 3361 of *Lecture Notes in Computer Science*. pp. 318–328. Springer Berlin, Heidelberg.

Ben-Bassat, M. (1982). Pattern recognition and reduction of dimensionality. In: *Handbook of statistics-II* (P. R. Krishnaiah and L. N. Kanal, Eds.). pp. 773–791. North Holland.

Blum, A. I. and P. Langley (1997). Selection of relevant features and examples in machine learning. *Artificial Intelligence* **97**(1-2), 245–271.

Blum, A. L. and R. L. Rivest (1992). Training a 3-node neural network is np-complete. *Neural Networks* (5), 117–127.

Caridakis, G., L. Malatesta, L. Kessous, N. Amir, A. Raouzaiou and K. Karpouzis (2006). Modeling naturalistic affective states via facial and vocal expressions recognition. In: *Proceedings of International Conference on Multimodal Interfaces (ICMI06), Banff, Alberta, Canada, November 2-4, 2006*. pp. 146–154.

Das, S. (2001). Filters, wrappers and a boosting-based hybrid for feature selection. In: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*. Morgan Kaufmann. pp. 74–81.

Dash, M. and H. Liu (1997). Feature selection for classification. *Intelligent Data Analysis* **1**, 131–156.

Dy, J. G. and C. E. Brodley (2000). Feature subset selection and order identification for unsupervised learning. In: *Proceedings of the Seventeenth International Conference on Machine Learning, June 29-July 2, 2000, Stanford University, CA*. Morgan Kaufmann. pp. 247–254.

Kim, Yongseog, W. Nick Street and Filippo Menczer (2000). Feature selection in unsupervised learning via evolutionary search. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press. pp. 365–369.

Kohavi, Ron and George H. John (1997). Wrappers for feature subset selection. *Artificial Intelligence* **97**(1), 273–324.

Mitra, P., C. A. Murthy and S. K. Pal (2002). Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **24**(3), 301–312.

Siedlecki, W. and J. Sklansky (1988). On automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence* **2**(2), 197–220.

Tian, Ying-Li, Takeo Kanade and Jeffrey Cohn (2001). Recognizing action units for facial expression analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(1), 97 – 115.

Wyse, N., R. Dubes and A.K. Jain (1980). A critical evaluation of intrinsic dimensionality algorithms. In: *Pattern Recognition in Practice* (E.S. Gelsema and L.N. Kanal, Eds.). pp. 415–425. Morgan Kaufmann Publishers, Inc.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 17–33

# SimTwo Realistic Simulator: A Tool for the Development and Validation of Robot Software

Paulo Costa[a,c,*], José Gonçalves[b,c], José Lima[b,c], Paulo Malheiros[a,c]

[a] *Faculty of Engineering of the University of Porto, DEEC, Porto, Portugal.*
[b] *Polytechnic Institute of Bragança, Department of Electrical Engineering, Bragança, Portugal.*
[c] *Robotics and Intelligent Systems research group from INESC PORTO, Portugal.*

**Abstract**

Many times the simulation environment is an ad hoc implementation done with the single purpose of testing the author's algorithm or methodology. It is not difficult to find that the accuracy of the simulation is low and mostly unable to reflect the real world. While, under certain circumstances, that can be a valid approach there is the need of a simulation environment where the focus is on the physical realism while retaining a lot of configurability so that it can be adapted to a lot of different uses. In this paper it is presented a simulation environment where multiple kinds of robots can be modeled. The robots can be modeled based on a network of physical bodies interconnected by joints that can be powered, or not, by electrical motors. The corresponding low level controllers and the high level decision or IA can also be implemented in the simulation environment. To achieve that, some established open source libraries like the Open Dynamics Engine are used. The parameters for the physical simulation are all accessible when defining the robot model and a very accurate motor model can also be implemented. There is also the ability to use external modules to implement the high level controllers.

*Keywords:* Robotics, Simulation, Sensors, Actuators.

## 1. Introduction

Simulation has established itself as an important tool in the mobile robotics field. The ability to test and develop robotic solutions in a virtual environment is widely used in research as well as in education, allowing the rapid development of robot software. In this context, having in mind the current needs of the authors, concerning both education and research it was developed SimTwo, being a versatile robot simulation environment that allows rapid test and design of differential, omnidirectional, industrial, humanoid robots, etc., developed in Object Pascal. SimTwo has a set of predefined components such as sensors and motors where specified models are inputed. The Open Dynamics Engine is used for the simulation of the rigid body dynamics. The robots look and behavior are defined in XML format files. The virtual world is represented using GLScene components (GLScene, 2010), these provide a simple implementation of OpenGL (OpenGL, 2010). The main motivation for the authors to develop a robot simulator was to have full control of the simulation features and its accuracy. The authors develop robot software and teach courses that use this tool as a teaching aid. They can add new features at any time and also improve the existing ones, never becoming limited by the features of the existing simulators (Gonçalves *et al.*, 2010).

For the simulator development it was used ODE (Open Dynamic Engine) (ODE, 2010), that provides an excellent real time time simulation, embedding it in a 3D application. ODE is an open source, high performance library for

*Corresponding author
*Email addresses:* paco@fe.up.pt (Paulo Costa), goncalves@ipb.pt (José Gonçalves), jllima@ipb.pt (José Lima), paulo.malheiros@fe.up.pt (Paulo Malheiros)

simulating rigid body dynamics. It is fully featured, stable, mature and platform independent with an easy to use C/C++ API. It has advanced joint types and integrated collision detection with friction. ODE is useful for simulating vehicles, objects in virtual reality environments and virtual creatures. It is currently used in many computer games, 3D authoring tools and simulation tools. There are other alternatives for physics Engines such as Box2D, Bullet, Chipmunk physics engine, SOFA (Simulation Open Framework Architecture), Tokamak physics engine and Jinngine (Java Physics Engine). It has also been made an effort to promote the use of different physics engines, for example using the Physics Abstraction Layer™(PAL), this provides a unified interface to a number of different physics engines. This enables the use of multiple physics engines within one application.

Some of the 3D simulators that are free and can be added new features are Gazebo Open Simulator and Simulator BOB. Gazebo is not open source although new Plugins can be uploaded with the user sensor and actuator modules. Open Simulator is open source and it can also be expandable through loadable modules to build complete custom configurations. Simulator BOB: 3D simulation environment for mobile robots is open source, with the goal to speed up its developing time, it is being developed for further research with artificial intelligence and autonomous mobile robots. Some of the commercial softwares available are Microsoft Robotics Studio, Webots by Cyberbotics and Marilou by Anycode (Michel, 2004).

There are also 2D simulation software suitable to develop software, mainly used in the area of artificial intelligence, not being important, for the robot software development, the interaction of the robot with a complex 3D world. Two examples of 2D simulators are Stage and Khepera simulator. Stage allows the simulation of a population of robots and the Khepera Simulator allows to develop controllers for the Khepera robot (Michel, 1996) (Wolfer & Rababaah, 2005) (Hassanzadeh *et al.*, 2008).

SimTwo is a free software that is not open source. Although it is not possible to upload new modules it is very flexible, providing a great variety of sensors and actuators and allowing to prototype and rapidly reconfigure different commercial and non commercial robots. One of the SimTwo important features is the ability of the use of remote clients that allow the development of robot software in different programming languages and commercial softwares such as MATLAB and LabVIEW. This feature is also available in both commercial and non commercial softwares, such Webots and Open Simulator. SimTwo has also a built in script controller that can communicate through serial port with micro-controllers and through Network with other applications. In section 2 is presented a description of the SimTwo simulator. In section 3 it is presented the modeling and simulation of a robot based on the Lego Mindstorms NXT Kit, where is given special focus to the actuator modeling and simulation. In section 4 some SimTwo applications examples are presented: an omnidirectional wheeled robot, a humanoid robot, and lighter-than-air vehicle. In the simulation of the wheeled robot it is presented how to develop the robot software with the purpose of its localization and navigation in a structured environment. The localization algorithm is achieved resorting to an Extended Kalman Filter, merging data provided from encoders (relative measurements) and distance data provided by infrared distance sensors. Finally some conclusions and future work are presented.

## 2. SimTwo – Realistic Simulator

SimTwo is a realistic simulation system that can support several types of robots. Its main purpose is the simulation of mobile robots that can have wheels or legs, although industrial robots, conveyor belts and lighter-than-air vehicles can also be defined. Basically any type of terrestrial robot definable with rotative joints and/or wheels can be simulated in this software. Figure 1 shows the software with all its main windows.

The dynamics realism in SimTwo is obtained by decomposing a robot in rigid bodies and electric motors. Each body behaviour is numerically simulated using its physical charatheristics: shape, mass and moments of inertia, surface friction and elasticity. It is also possible to define standard joints such as socket, hinge and slider which can be coupled with an actuator or a sensor.

SimTwo is an application with a *multiple document interface* (MDI) where all windows are under the "world view" window control, shown in Figure 1, exiting the simulator is done by closing this window. The "configuration" window offers control over several elements of the virtual scene. It is possible to define the controller timestamp and to configure the 3D world view (camera position, shadow visibility, etc.). Robots information is also displayed in this window, like its position and speed.

The "code editor" offers an *integrated development environment* (IDE) for high-level programming based in Pascal language, this is the main tool in this simulator. The control algorithms are directly compiled in this window, a message

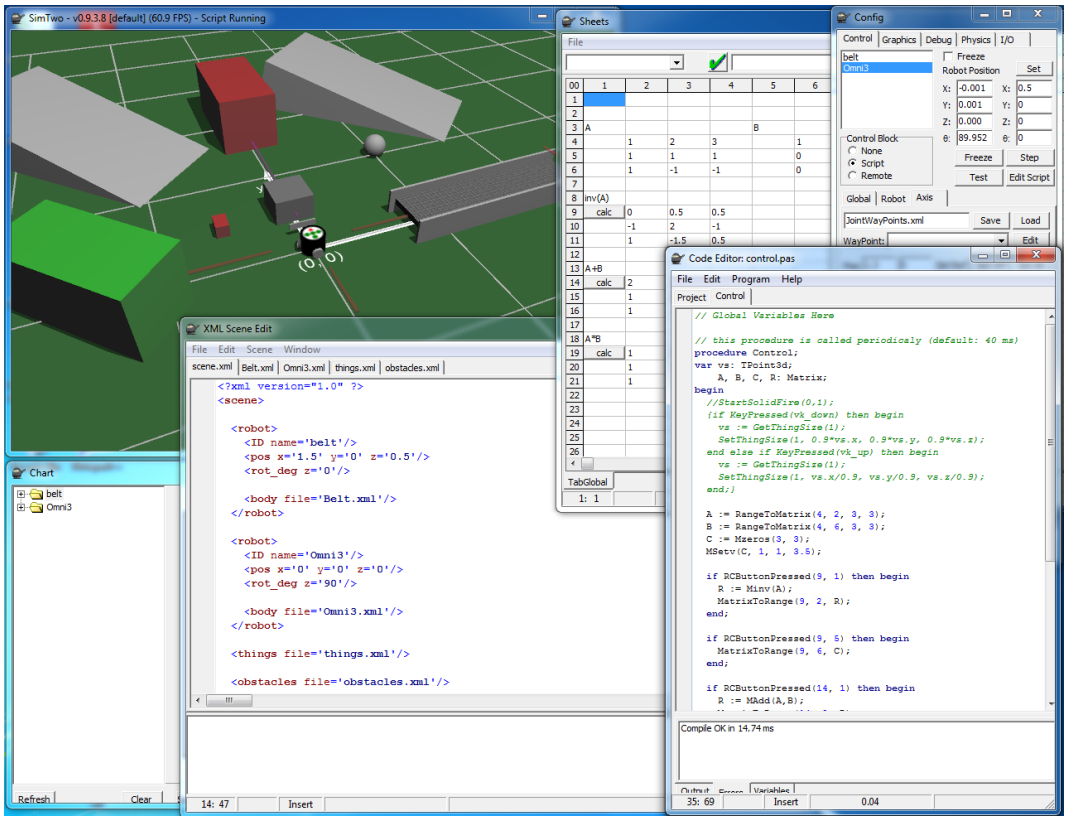


Figure 1: SimTwo software. Application windows clockwise from top left corner: world view, spreadsheet, configuration, code editor, scene editor and chart.

appears in the bottom of the page informing of a successful compilation or the existence of coding errors. The control script is started from this window where the resulting robot movement is visible in the main window, any changes to the control script requires this to be stopped and recompiled.

Debugging the control algorithm is possible using the “chart” and the “spreadsheet” window. In the “chart” it is possible to plot all variables available for every robot, such as its position, motors speed and current, etc. In the “spreadsheet” window it is possible to define “edit cells” as well as “button cells” for specific operations. This window becomes a customizable form window, this is the equivalent to a graphic application.

The scene implementation is done by editing several XML format files, these files are definable in the “scene editor” window. A scene in SimTwo can have “robots”, “obstacles” and “things”, as shown in Figure 2.

A main scene file (scene.xml) defines the robots in use and their specific construction file. Each robot is defined by various solids (cuboid, cylinder and sphere) connected through joints (slider, socket and hinge). The shell elements are solids without mass, these do not modified the robot physical properties but are an essential part for collision simulation. A robot can also have sensors, these provide information from the environment surrounding the robot.

The scene objects are the “obstacles” and “things”, these are very similar in definition (both are defined solely by solids) but while the “obstacles” are imovable in case of collision the “things” are not. If a robot colides with a “thing” these will react accordingly to their mass definition. A scene can also have sensors, these are static relative to the world as opposed to a robot sensor which gives information relative to its corresponding robot.

In Figure 3 it is presented the flux of information of a SimTwo controller. The controller has different levels that are updated at different rates, being presented its default values. The presented default values can be changed by the user depending on the application that is being simulated. The artificial intelligence controller is updated by default at 40 ms, being a common rate to accomplish real time requisites in mobile robotics challenges. The motor controller is updated by default at a 10 ms rate and its model output is updated by default at 1 ms rate. The default values

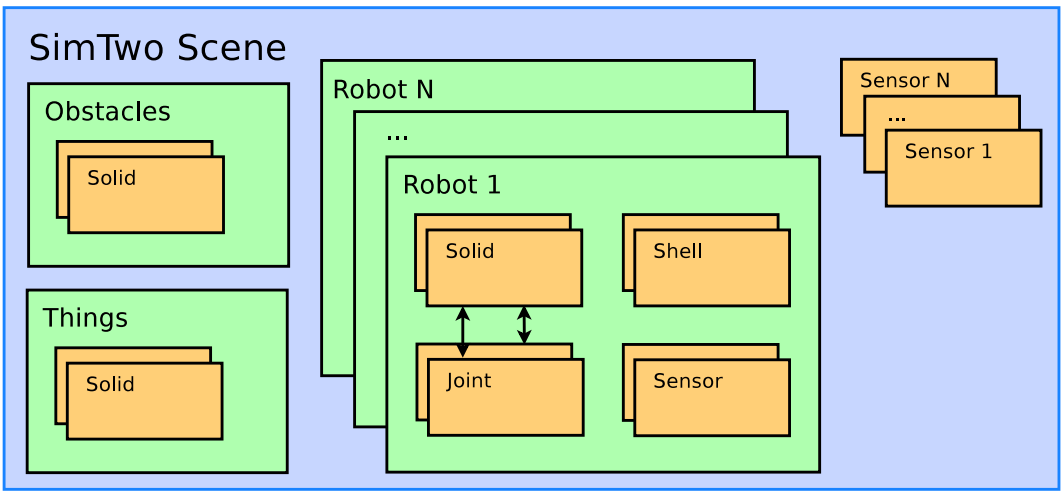


Figure 2. SimTwo Scene structure

are typical values that were chosen having in mind the dynamics of the world and motor models in mobile robotics, although user can change this values depending on the specif dynamics and real time requisites of its simulation.

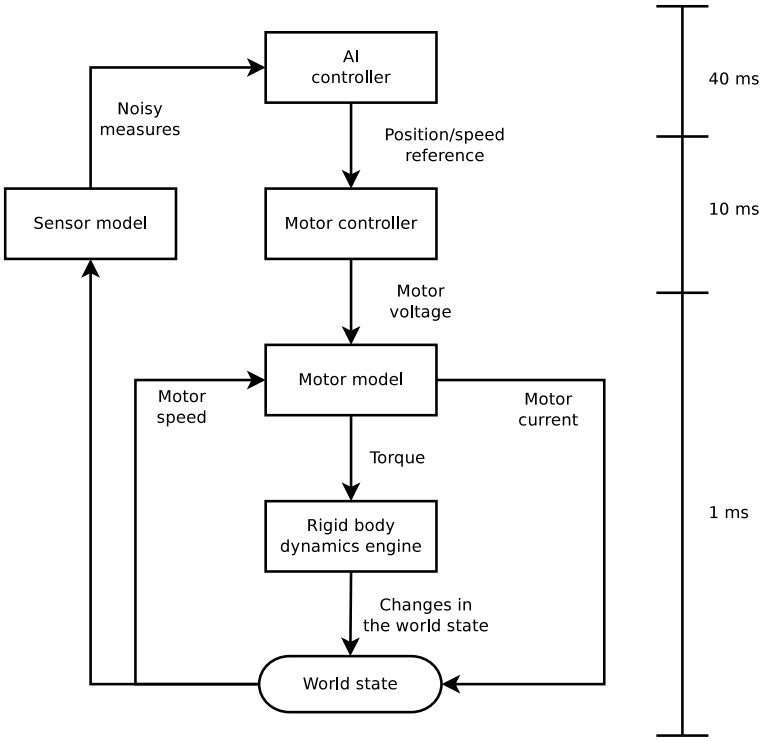


Figure 3. SimTwo controller cycle

The artificial intelligence level is provided with sensor information extracted from the world state. After some decisions and calculations related with control, localization and navigation the controller provides to the motor controller its inputs. The motor model provides the torque information, forcing the Rigid Dynamics Engine to react and consequently forcing the World state (Sensor, obstacles and robot positions) to change.

## 3. Modeling and simulation of a Lego Mindstorms Robot

Lego Mindstorms is a powerful educational tool (Gawthrop & McGookin, 2006) (Lund & Pagliarinis, 2000), being used by the authors in many activities. It has been used to teach mobile robots introductory concepts to undergraduate and graduate students and in several demonstrations. The undergraduate students while attending summer courses prototype their own robots and develop robot software based on the Lego Mindstorms educational software. The graduate students while attending robotics classes also prototype their own robots, with the difference that the developed robot software is done resorting to high level programming languages. It has also been used in demonstrations with the goal of captivate and inform undergraduate students, concerning the areas that involve technology.

In this section it is described the modeling of an NXT based robot and how it can be simulated using SimTwo, giving a special attention to the actuator model and simulation (Campion *et al.*, 1996) (Khosla, 1989) (Olsen & Petersen, 2001). It is not presented the sensor modeling and it is not presented the comparative between the simulated and the real robot, this results can be found in (Gonçalves *et al.*, 2009*b*). The presented approach does not replace the training with hardware but is an important complement. Students can develop and test robot controllers even at their homes, without the need of having access to hardware. This situation is important because usually students have access to hardware only when they are attending classes or when they have tutorial classes at the robotics laboratory. The simulation is also important to monitor some interesting variables, that are much more difficult to access in the real world. As a direct benefit the robot behavior can be monitorized in a more accurately way.

The Lego modularity makes the rapid prototyping of different robot configurations easier. This easiness presents itself as an extra motivation for the persons who are taking their first steps in the world of mobile robotics. The Lego parts allow connectivity, suppressing the need of using glue or screws. It is also an ecological tool because although it is not easy to recycle plastic, the Lego parts can be reused. It is a modular tool, it is possible for the same part to be a different thing depending on the application, motivating those who are using it and it is presented at a relatively low cost. Added to all the presented advantages of the Kit Lego Mindstorms, simulation can also be a very important teaching aid.

The modeled and simulated prototype, presented in Figure 4, is a differential robot. Its movement is controlled by varying the velocity of each wheel independently.


Figure 4. Driving Base of the Lego Mindstorms NXT

Design behavior without real hardware is possible due to a physics-based simulator implementation. The dynamic behavior of the simulated robot is computed by ODE Open Dynamics Engine, a free library for simulating rigid body

dynamics. The simulator architecture is based on the real Lego NXT robot. The body masses and dimensions are followed in order to build a simulated robot like the real one.

### 3.1. Actuator modeling

The Lego Mindstorms NXT kit provides three servomotors with built-in rotation sensor and a gear ratio of 1:48. A Lego Mindstorms NXT servomotor is shown in Figure 5.

Figure 5. Lego Mindstorms NXT servomotor.

The servomotor can be resumed to a DC motor model, presented in Figure 6, where $U_a$ is the converter output, $R_a$ is the equivalent resistor, $L_a$ is the equivalent inductance and $e$ is the back emf (electromotive force) voltage as expressed by equation (3.1).

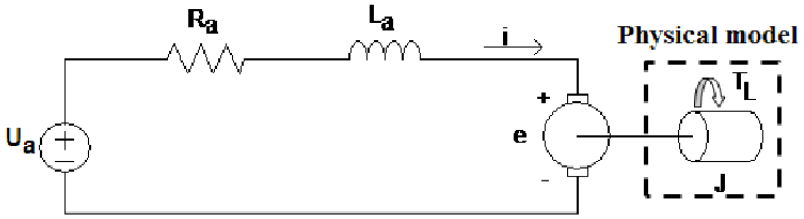


Figure 6. DC motor electric model.

$$U_a = e + R_a i_a + L_a \frac{\partial i_a}{\partial t} \tag{3.1}$$

The motor can supply a torque $T_L$ and the load has a moment of inertia $J$ that will be computed by the physical model ODE. Current $i_a$ can be correlated with the developed torque $T_d$ through equation (3.2) and the back emf voltage can be correlated with angular speed through equation (3.3), where $K_s$ is a motor parameter that can be found by an experimental setup as presented in subsection 3.1.1 (Bishop, 2002).

$$T_d(t) = K_s i(t) \tag{3.2}$$

$$e(t) = K_s \omega(t) \tag{3.3}$$

In fact, the real developed torque (useful) that will be applied to the load ($T_L$) is the developed torque subtracted by the friction torque ($T_c$).

$$T_L = T_d - T_c \tag{3.4}$$

*3.1.1. DC motor model measurements:*

It was used the NXT Lego Mindstorms motor as the base of the simulator. The $R_a$ and $L_a$ values can be directly measured ($R_a$=7.6 Ω and $L_a$= 4.88 $mH$). The $K_s$ motor parameter can be found by an indirect measure. For several angular speeds, it can be measured the emf voltage while the motor is in open circuit. Table 1 presents the data for 7 measures.

Table 1. Speed and emf voltage.

| $\omega$ (rotations/min) | $\omega$ (rad/s) | e (V) | k=e/$\omega$ |
|---|---|---|---|
| 0 | 0 | 0 | |
| 66 | 6.91 | 3.4 | 0.491 |
| 115 | 12.04 | 6.0 | 0.498 |
| 155 | 16.23 | 7.9 | 0.487 |
| 195 | 20.42 | 9.8 | 0.479 |
| 233 | 24.39 | 11.9 | 0.487 |
| 265 | 27.75 | 13.85 | 0.499 |

Figure 7 shows graphical data of the $K_s$ line and its trend line. The average value for $K_s$ (line slope) is about 0.4919 V.rad/s. It is possible to observe that the error in the origin for the presented approximation it is negligible, being only 0.0218 V. Despite being a small error it is possible to increase the precision in the $K_s$ parameter estimation, forcing the trend line to pass in the origin. The obtained value for $K_s$ is nearly 0.4908 V.rad/s.

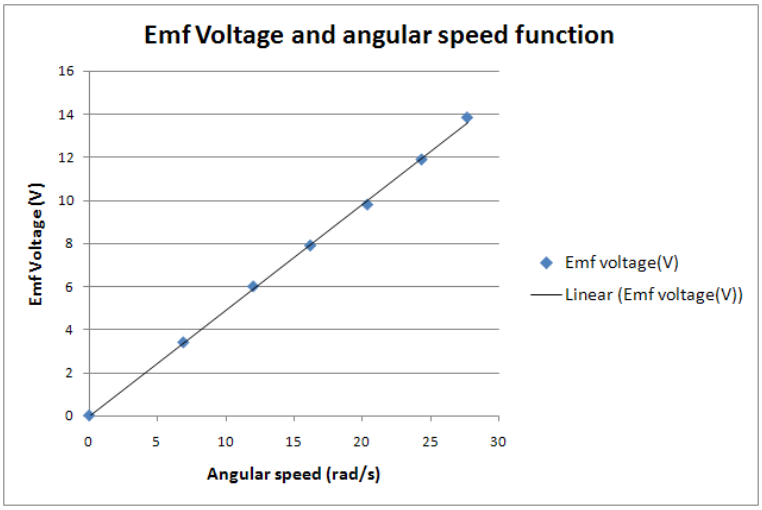


Figure 7. $K_s$ value for DC motor model.

In order to calculate the friction it was obtained the motor angular velocity for different voltages at its terminals. After a linear regression it was obtained equation (3.5).

$$\omega = 1.9248v - 0.2519 \tag{3.5}$$

Where $\omega$ is the angular velocity in rad/s and $v$ is the voltage at the motor terminals.

Assuming that the developed torque ($T_d$) is equal to the torque generated by the static friction ($T_c$) it is possible, resorting to the linear regression presented in equation (3.5), to obtain the necessary voltage to equal the static friction ($v_c$), by replacing by zero. For this situation it was not necessary to include in the model the viscous friction, because it only exists for angular velocities different from zero. As there is no movement there is no voltage generated by the emf and as the current has no variation it is possible to obtain the produced torque by the static friction resorting to

equations (3.1) and (3.2), resulting in equation (3.6). The torque due to the static friction has a numerical value of 3.558E-4 Nm.

$$T_c = K_s \frac{v_c}{R_a} \tag{3.6}$$

After obtaining the static friction there is only left, to complete the motor model, to estimate $B$, the parameter that relates the viscous friction with the motor angular velocity, as shown in equation (3.7).

$$T_\omega = B\omega \tag{3.7}$$

Placing the motor spinning without load the developed torque will given by equation (3.8).

$$T_d = T_c + B\omega \tag{3.8}$$

As the developed torque is proportional to the current as shown in equation (3.2) it is possible to conclude, from equation (3.8), that the current can be given by equation (3.9).

$$i_a = \frac{T_c + B\omega}{K_s} \tag{3.9}$$

The voltage applied to the motor terminals can be given by equation (3.1), but as in steady state the current variations are small, it can be obtained equation (3.10) for the motor terminal voltages.

$$U_a = K_s\omega + R_a i_a \tag{3.10}$$

where $e$ was replaced by $K_s\omega$ as exemplified in equation (3.3).
From equations (3.9) and (3.10) it is obtained equation (3.11).

$$\omega = \frac{U_a - \frac{R_a T_c}{K_s}}{\frac{R_a B}{K_s} + K_s} \tag{3.11}$$

Minimizing the sum of the absolute error between the real angular velocity and the obtained by the model (equation (3.11)), it is obtained a numerical value for $B$ (1.92E−3).
The estimated motor parameters are shown in Table 2.

Table 2. Motor parameters

| Parameters | Value SI units |
|---|---|
| $K_s$ | 0.4908 |
| $T_c$ | 3.558E-4 |
| $B$ | 1.92E−3 |
| $R_a$ | 7.6 |
| $L_a$ | 4.88E−3 |

The different obtained motor models can be observed in the graphics shown in Figure 8, where *rot* is $\omega$ (the motor angular velocity).

*3.1.2. DC motor nonlinearities:*

In a way to map the reality closer, where variables cannot assume all values, the model must have some limitations. The first one, the voltage applied to the supply terminals $U_a$. This voltage should be limited to the batteries voltage. Further, current $i$ should be limited once it is related to the torque through equation (3.2). Figure 9 shows the limitations presented in the servomotor model.

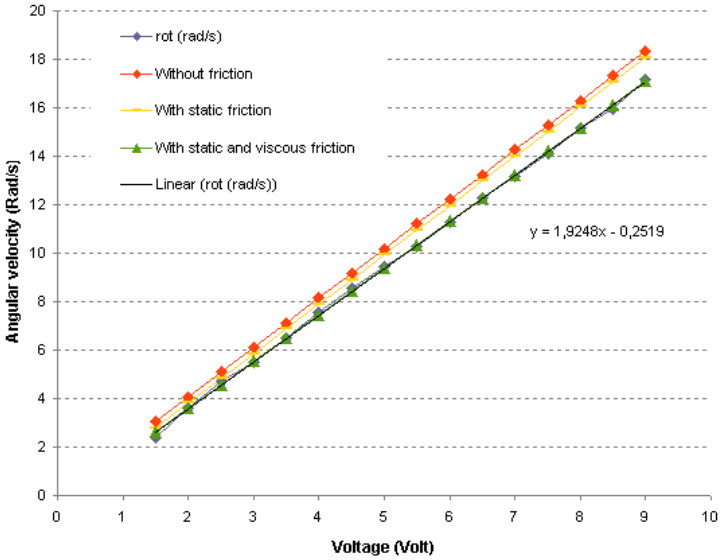


Figure 8. Different obtained models.

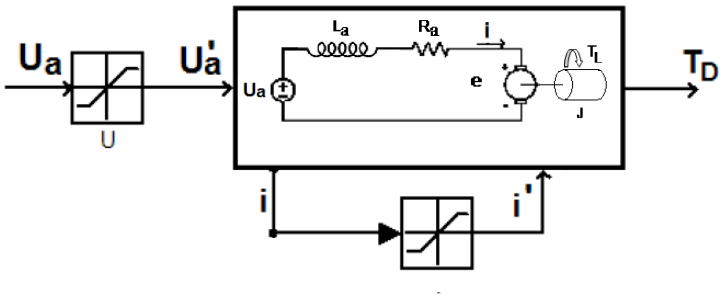


Figure 9. Servomotor model nonlinearities.

### *3.2. Robot simulation based on the SimTwo*

The robots look and behavior are defined in XML format files. The virtual world is represented using GLScene components, these provide a simple implementation of OpenGL. The NXT robot was defined using boxes and cilynders as shown in Figure 10. These are the objects used by the ODE to determine the collisions and friction. SimTwo allows the replacement of the solids by any 3DS model which gives a realistic look of the robot as shown in Figure 11.

## 4. SimTwo Application Examples

The presented examples are in the field of edutainment robotics. Edutainment robotics plays an important role in education due to the inherent multi-disciplinary concepts that are involved, motivating students to technological areas. It also plays an important role in research and development, because it is expected that the outcomes that will emerge here, will later be transfered to other application areas, such as service robots and manufacturing.

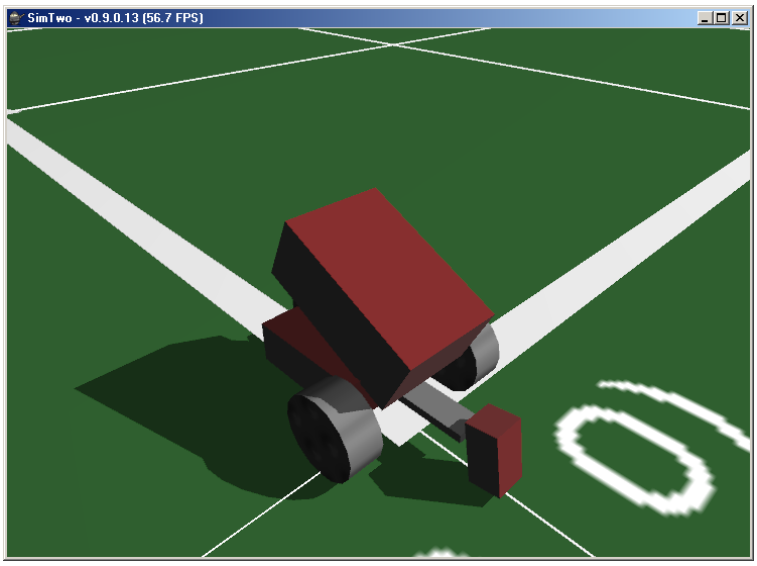


Figure 10. NXT robot built in the SimTwo

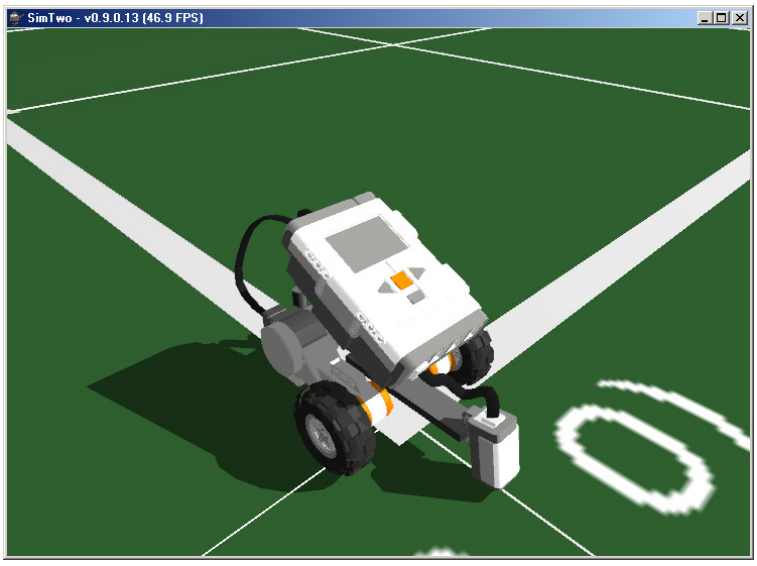


Figure 11. NXT robot built in the SimTwo with 3DS models

### *4.1. Simulation of an omnidirectional wheeled robot*

This section presents the modeling and simulation of a wheeled mobile robot designed to operate in structured environments and how to develop robot software based on its simulation. It is presented the localization and navigation in a Fire Fight Robot Contest arena of an omnidirectional mobile robot equipped with brushless motors and infra-red distance sensors (Williams *et al.*, 2002) (Leow *et al.*, 2002) (Loh *et al.*, 2003) (Muir & Neuman, 1987). The used control architecture is generic and can be applied to several control challenges other than mobile robotics. The robot controller was implemented resorting to the simulator, using a remote client that communicates with the simulator using the UDP protocol, as shown in Figure 12. It is also possible to control robots using the embedded script functionality. The developed code can be migrated to the real robot, making the simulation useful, as presented previously in (Gonçalves *et al.*, 2009*a*).

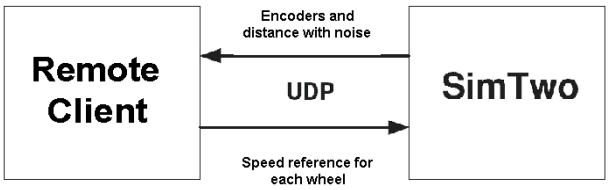


Figure 12. Simulator communicating with remote client.

The localization and navigation software is developed resorting to the simulation shown in (Figure 13). The simulator provides to a remote client the distance sensors data with noise as modeled in (Gonçalves *et al.*, 2008) and the encoders data. The remote client executes the localization and navigation algorithms and returns the speed references for each wheel to the simulator.

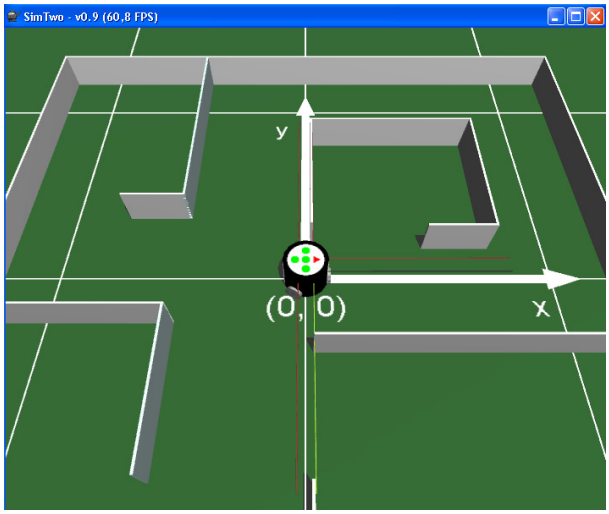


Figure 13. Robot simulator snapshot.

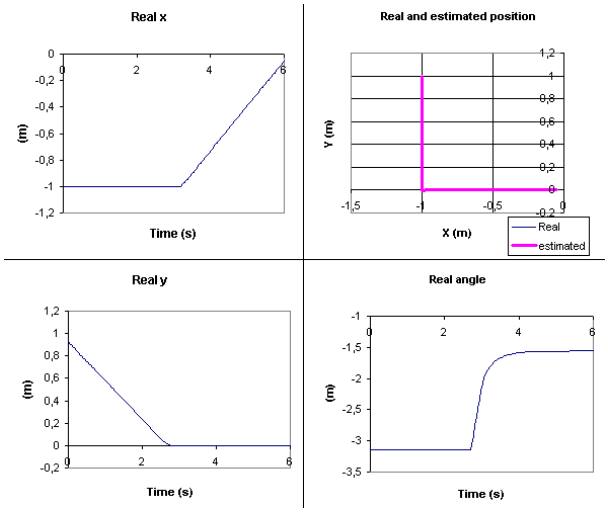


Figure 14. Robot trajectory.

Odometry and infra-red distance sensors data fusion was achieved applying an extended Kalman filter. This method was chosen because the robot motion equations are nonlinear and also because the measurements error probability distributions can be approximated to Gaussian distributions (Choset *et al.*, 2004) (Thrun *et al.*, 2005). As an example, a robot trajectory produced in the simulator is shown in Figure 14. The pose estimate error and its variance are shown in Figure 15.

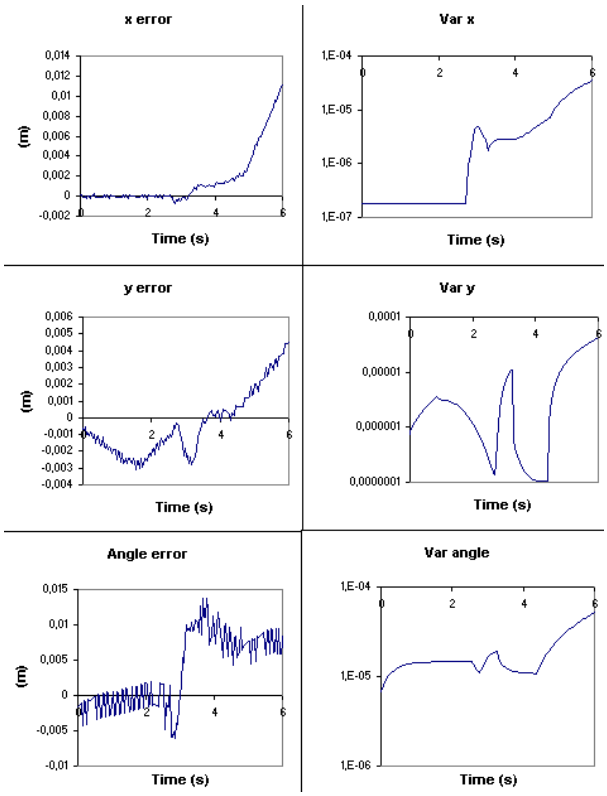


Figure 15. Pose estimate error and variance.

### *4.2. Simulation of a Humanoid Robot*

In last years, research studies in biped robots evolved rapidly and resulted in a variety of prototypes that resemble the biological systems. Legged robots have the ability to choose optional landing points, an advantage to move in rugged terrains, and two legged robots are also able to move in human environments. So, studies about biped robots are very important (Suzuki & Ohnishi, 2006) (Zagal *et al.*, 2009). A humanoid robot dynamic model is a complex mathematical matter, however, it can become easy if it is used a simulator.

Nowadays, there are several simulators that own the humanoid robot simulation capability. Meanwhile, this simulator allows to simulate different types of robots and allows the access to the low level behaviour, such as dynamical model, friction model and servomotor model in a way that can be mapped to the real robot, with a minimal overhead. The main purpose of Simtwo support humanoid simulations capability is to allow to develop new methods of control the robot (i.e. walk, get-up movements) and implement new issues that further can be applied to the real robot. There are several humanoid robots kits available. The commercially available Bioloid robot kit, from Robotis, served as the basis for the humanoid robot and the proposed biped robot is shown in Figure 16 and Figure 17 shows the humanoid robot in the Simtwo simulation environment. The presented humanoid robot is driven by 19 servo motors (AX-12): six per leg, three in each arm and one in the head. Three orthogonal servos set up the 3DOF (degree of freedom) hip joint. Two orthogonal servos form the 2DOF ankle joint. One servo drives the head (a vision camera holder). The

shoulder is based on two orthogonal servos allowing a 2DOF joint and elbow has one servo allowing 1DOF. The total weight of the robot (without camera and on board computer) is about 2 kg and its height is 38 cm.

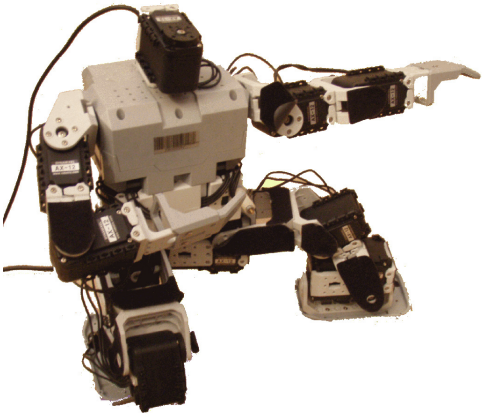

Figure 16. Real humanoid robot (Robotis).

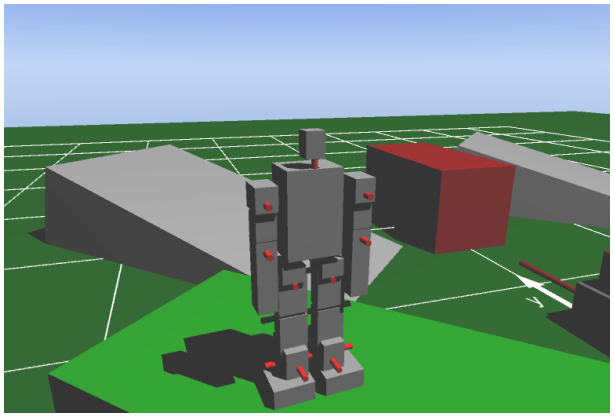

Figure 17. Simtwo Humanoid simulation.

The humanoid robot can be described through bodies and joints. Each body mass simulates the servo motors and connection pieces weights from the real robot. ODE joints, imitate the servo motors axis movements and must be defined its types, angles and torques limits. Joints types can be classified as hinges or universal joints: a hinge that allows both bodies to be connected and roll such as arms and forearms, femur and leg; a universal joint must be introduced when there are two or more degrees of freedom between two bodies. It happens when two servo motors are physically combined. A universal joint allows two bodies to roll on both axes. As example, presented in the simulator, these joints connect trunk and arms, trunk and legs, legs and feet.

The humanoid servomotors were modeled and results presents a comparison between real and simulated servos. The friction constants can be found resorting to a free fall of an arm (Figure 19) and actuators constants (state-space controller gains) allows to obtain a comparison between real and simulated humanoid servos step response, as presented in Figure 20.

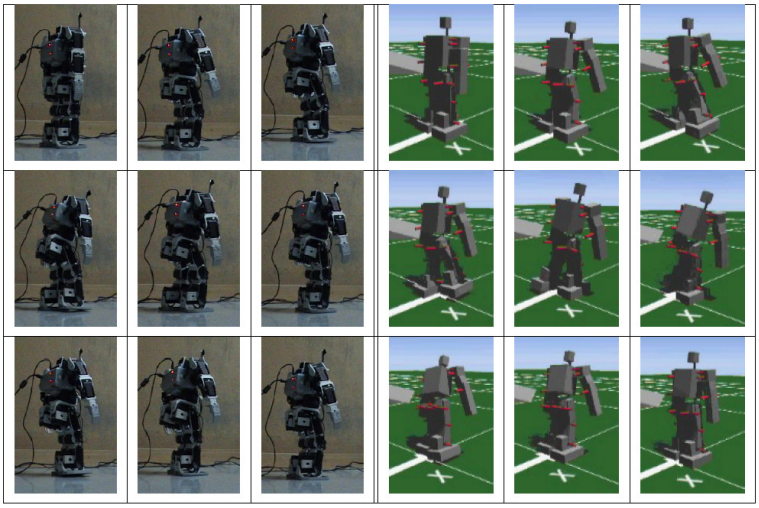

Figure 18. Real and Simtwo humanoid walk comparison.

A way to validate the humanoid simulation model is to apply the same control signal to both robots and to analyze the behavior. Predefined trajectory states, that allow robot to walk, are based on the Zero Moment Point (ZMP) method

and are well described in literature (Kajita *et al.*, 2006) (Zhang *et al.*, 2008) (Meghdari *et al.*, 2008). Figure 18 shows the sequence during walk movements for both robots (real at left and simulator at right).

It is possible to observe that both robots exhibit a very similar behavior. Previous works present the validation in other way: energy consumption on get-up movements (Lima *et al.*, 2008) and angles from real and simulator joints are compared (Lima *et al.*, 2009). These facts prove that humanoid simulator environment acts as a real robot.

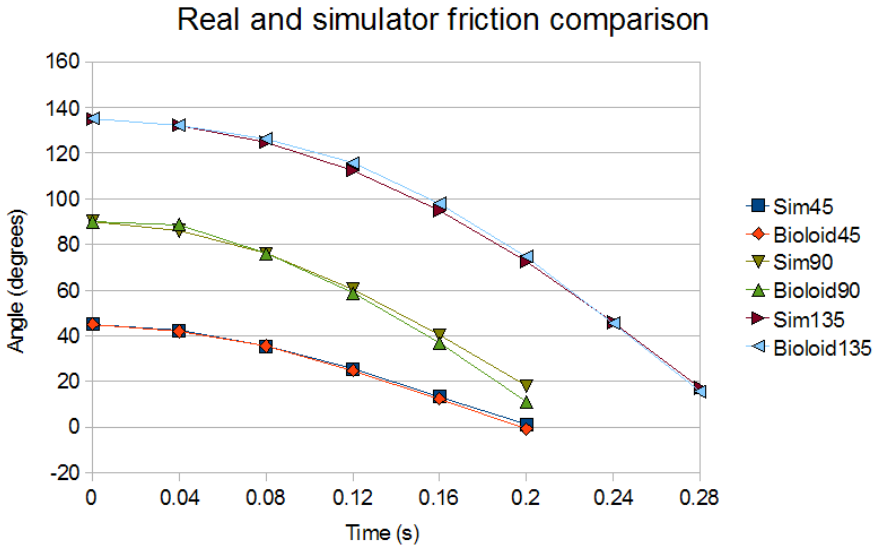


Figure 19. Real and simulator friction comparison.

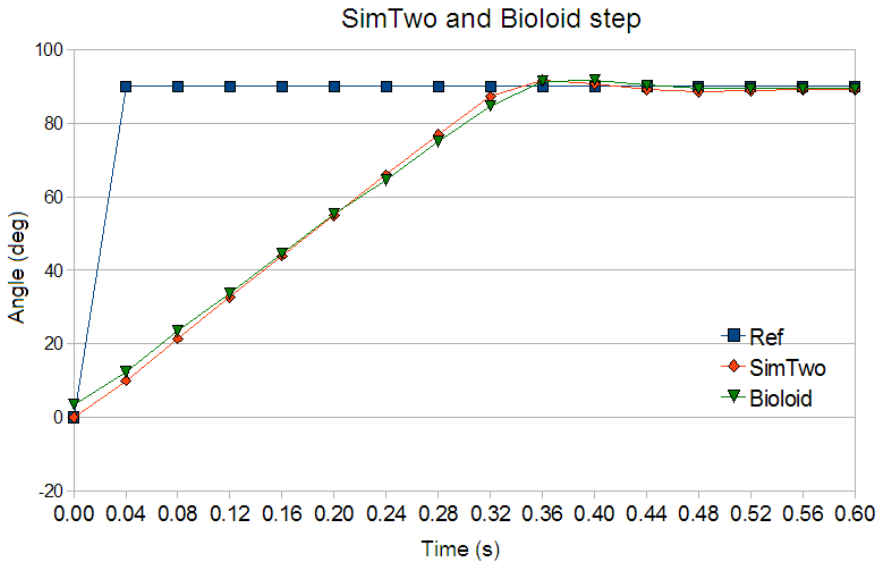


Figure 20. Simtwo and Bioloid step.

### *4.3. Simulation of a Lighter-than-air vehicle*

One of the most interesting items in this simulator is an object that can elevate itself in the air. It is possible to create balloons and airships, as show in Figure 21.

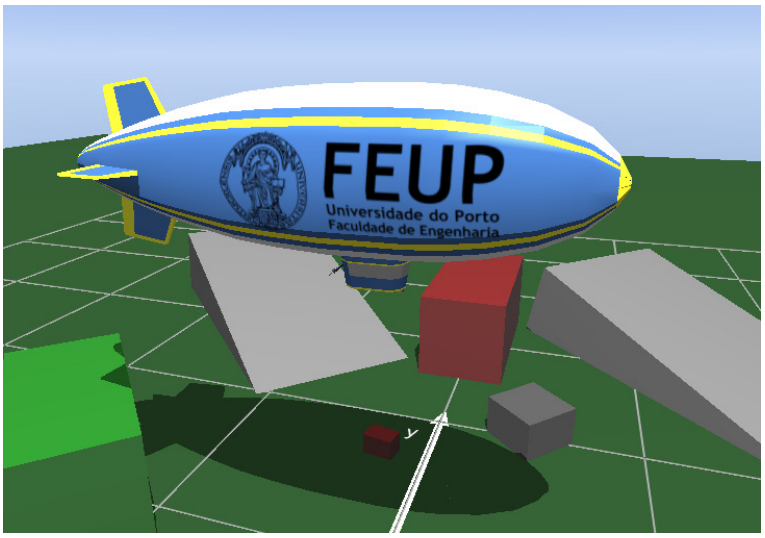


Figure 21. Blimp in SimTwo.

This type of vehicles is centered in one solid with a "buoyant" property, this is an upwards force that simulates the properties of the lifting gas. For more realism it is possible to define drag coefficient that is used to quantify the air resistance suffered by the vehicle. For movement this airborne vehicles require the use of propellers which was added to the SimTwo simulator.

This lighter-than-air application was specifically used for educational purposes in a microcontroller course unit. The students were required to model and control a real lighter-than-air vehicle which was a model of the famous "Goodyear Blimp", shown in Figure 22. A block diagram of educational experiment is shown in Figure 23.


Figure 22. Lighter-than-air vehicle class presentation.

A model was defined using a lift solid as the "balloon" and a second solid defining the control gondola in the correct size and position. The students were required to weigh these model elements for tuning the model and set the correct buoyancy parameter for air balance. Two propellers define the forward/backward movements as well as

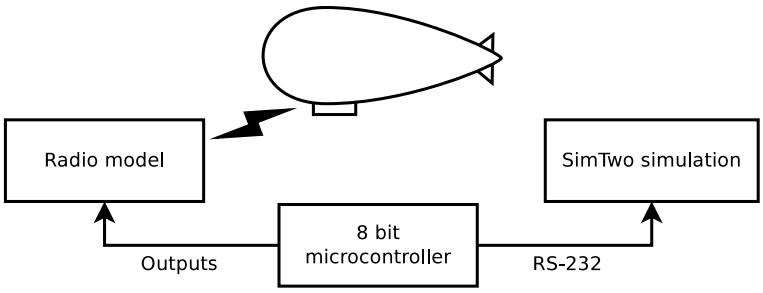


Figure 23. Zeppelin educational experiment.

steer the airship, a third propeller, placed horizontally below the gondola, allows the upwards/downwards lift. The horizontal and vertical speed can be defined in SimTwo tuning the motor parameters after the measurement of the same speeds in the real vehicle. A 3DS skin was added to make its appearance more realistic.

The main objective of the proposed educational experiment consisted in developing a 8 bit microcontroller application. The trajectory was defined by an open loop sequence of movements such as front, back, rotate left, rotate right, up and down which should be tested in SimTwo prior to using the real vehicle. The microcontroller sent the movement commands to SimTwo using a RS-232 serial connection thus validating the inputed trajectory. The same commands were sent to a radio module which relayed them to the airborne vehicle.

## 5. Conclusions and future work

The presented simulation environment has already been used on some educational settings where the students tested their control algorithms on a model before porting them to real robots. That avoided costly mistakes and increased the students exposure to the robots, even if some of that time was on a virtual robot. The SimTwo capabilities where also stressed on some research on humanoid robots that lead to a phd and some master thesis. That drove the development of SimTwo as the current research is still doing. While the initial focus was on mobile robots with omnidirectional wheels and humanoid robots, the capabilities where extended so that modeling robotic manipulators, car like robots, lighter than air vessels and machines with conveyor belts is already possible. There is an effort underway to implement more sensors with careful modeling from real ones. Also some physical devices like a mechanical differential are planned. The present library of already modeled robots is being expanded. While SimTwo is Wine friendly and able to run on a Linux x86 PC some effort is planned to make it truly cross platform.

## References

Bishop, R. (2002). *The Mechatronics Handbook*. CRC Press, New York.

Campion, G., G. Bastin and B. Dandrea-Novel (1996). Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation* **12**(1), 47–62. 1042-296X.

Choset, H., K. Lynch, W. Burgard, L. Kavraki and S. Thrun (2004). *Principles of robot motion*. MIT Press.

Gawthrop, P. and E. McGookin (2006). Using lego in control education. In: *Proceedings of the 7th Ifac Symposium on Advances in control education*.

GLScene (2010). http://glscene.sourceforge.net/wikka/HomePage.

Gonçalves, J., J. Lima, H. Oliveira and P. Costa (2008). Sensor and actuator modeling of an realistic wheeled mobile robot simulator. In: *Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2009*a*). Code migration from a realistic simulator to a real wheeled mobile robot. In: *9th Conference on Autonomous Robot Systems and Competitions, Castelo Branco, Portugal*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2009*b*). Realistic simulation of a lego mindstorms nxt based robot. In: *Scheduled for presentation during the Invited CCA Session "LEGO based Control Education and Prototyping in Robotics, Mechatronics and Embedded Systems, IEEE Multi-conference on Systems and Control", Saint Petersburg, Russia*.

Gonçalves, J., J. Lima, P. Malheiros and P. Costa (2010). Fostering advances in mechatronics and robotics resorting to simulation. In: *Proceedings of the 10th IFAC Workshop on Intelligent Manufacturing Systems, Lisabon*.

Hassanzadeh, I., H. Ghadiri and R. Dalayimilan (2008). Design and implemention of a simple fuzzy algorithm for obstacle avoidance navigation of a mobile robot in dynamic environment. In: *Proceeding of the 5th IEEE International Symposium on Mechatronics and its Applications (ISMA08)*.

Kajita, S., M. Morisawa, K. Harada, K. Kaneko, F. Kanehiro, K. Fujiwara and H. Hirukawa (2006). Biped walking pattern generator allowing auxiliary zmp control. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2994-2999, Beijing, China*.

Khosla, P. (1989). Categorization of parameters in the dynamic robot model. *IEEE Transactions on Robotics and Automation* **5**(3), 261–268. 1042-296X.

Leow, Y. P., Low K. H. and Loh W. K. (2002). Kinematic modelling and analysis of mobile robots with omni-directional wheels. In: *Seventh lnternational Conference on Control, Automation, Robotics And Vision (lCARCV'O2)*. Singapore.

Lima, J., J. Gonçalves, P. Costa and A. Moreira (2008). Realistic behaviour simulation of a humanoid robot. In: *8th Conference on Autonomous Robot Systems and Competitions, Aveiro, Portugal*.

Lima, J., J. Gonçalves, P. Costa and A. Moreira (2009). Humanoid realistic simulator: The servomotor joint modeling. In: *Proceedings of International Conference on Informatics in Control, Automation and Robotics, Milan, Italy*.

Loh, W. K., K. H. Low and Y. P. Leow (2003). Mechatronics design and kinematic modelling of a singularityless omni-directional wheeled mobile robot. In: *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*. Vol. 3. pp. 3237–3242.

Lund, H. and L. Pagliarinis (2000). Robocup jr. with lego mindstorms. In: *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA, IEEE*.

Meghdari, A., S. Sohrabpour, D. Naderi, S. Tamaddoni, F. Jafari and H. Salarieh (2008). A novel method of gait synthesis for bipedal fast locomotion. *Journal of Intelligent and Robotic Systems* **53**(2), 99–202.

Michel, O. (1996). *Khepera Simulator version 2.0*. User Manual.

Michel, O. (2004). Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems* **1**(1), 39–42.

Muir, P. and C. Neuman (1987). Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. In: *Proceedings 1987 IEEE International Conference on Robotics and Automation*. Vol. 4. pp. 1772–1778.

ODE (2010). http://www.ode.org/.

Olsen, M. M. and H. G. Petersen (2001). A new method for estimating parameters of a dynamic robot model. *IEEE Transactions on Robotics and Automation* **17**(1), 95–100. 1042-296X.

OpenGL (2010). http://www.opengl.org/.

Suzuki, T. and K. Ohnishi (2006). Trajectory planning of biped robot with two kinds of inverted pendulums.. In: *Proceedings of 12th International Power Electronics and Motion Control Conference, Portoroz, Slovenia*.

Thrun, S., W. Burgard and D. Fox (2005). *Probabilistic robotics*. MIT Press.

Williams, R., B. Carter, P. Gallina and G. Rosati (2002). Dynamic model with slip for wheeled omnidirectional robots. *IEEE Transactions on Robotics and Automation* **18**(3), 285–293. 1042-296X.

Wolfer, J. and H. Rababaah (2005). An integrated khepera and sumo-bot development environment for assembly language programming. In: *Proceeding of the International Conference on Engineering and Computer Education*.

Zagal, J., J.Delpiano and J. Solar (2009). Self-modeling in humanoid soccer robots. *Robotics and Autonomous Systems*.

Zhang, L., C. Zhou and R. Xiong (2008). A lie group formulation for realtime zmp detection using force/torque sensor. In: *Proceedings of the 11th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines, pp. 1250-1257, Coimbra, Portugal*.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 34–44

# Sensitivity Investigation of Fault Tree Analysis with Matrix-Algebraic Method

László Pokorádi[a,*]

[a] *University of Debrecen, Debrecen, Hungary.*

## Abstract

The Fault Tree Analysis (FTA) is a systematic, deductive (top-down type) and probabilistic risk assessment tool which shows the causal relations leading to a given undesired event, referred to as the "Top Event" (TE). The events, which cannot be subdivided, are called the Basic Events. Fault Tree diagram displays the undesired state of the investigated system (top event) in terms of the states of its components (basic events). The Fault Tree Analysis is a graphical design technique main result of which is a tree, a dendritic graph. Probabilistic Fault Tree Analysis (PFTA) is a quantitative analysis method used to calculate the probability of Top Event from given failure probabilities of system components. The objective of the sensitivity analysis is to show how the change in any system parameter influences the resultant reliability value of the whole system.

The main aim of this study is to elaborate an easy-used algorithm for setting-up of Linear Fault Tree Sensitivity Model (LFTSM). This modular approach process uses matrix-algebraic method based upon the mathematical diagnostic modeling of aircraft systems and gas turbine engines. The paper shows the adaptation of linear mathematical diagnostic modeling methodology for setting-up of LFTSM and its possibility of use to investigate Fault Tree sensitivity by a demonstrative example.

*Keywords:* Reliability, Availability, Maintenance, Sensitivity Analysis.
*2000 MSC:* 68M15, 49Q12, 93B35.

## 1. Introduction

Several analytical methods of dependability and reliability analysis are available, of which Fault Tree Analysis (FTA) is one. The purpose of each method and their individual or combined applicability in evaluating the reliability and availability of a given system or component should be examined by the analysts before starting the FTA. Consideration should also be given to the results available from each method. Data required performing the analysis, complexity of analysis, and other factors identified in the Standard lEC 1025 (IEC, 1990).

The Bell Telephone Laboratories developed the concept of FTA at the beginning of the 1960s for the U.S. Air Force for use with the Minuteman system. It was later adopted and extensively applied by the Boeing Company. Fault Tree Analysis is one of many symbolic "analytical logic techniques" found in operations research and in system reliability.

Probabilistic Fault Tree Analysis (PFTA) is a quantitative analysis method used to calculate the probability of Top Event from given failure probabilities of system components. For PFTA we have to know probability data of component failures (basic events), which can be hunted by:

- technical books;
- data delivered by manufacturer;
- laboratory evidences

*Corresponding author
*Email address:* pokoradi@mk.unideb.hu (László Pokorádi)

- experts' opinions;
- statistical analysis of maintenance data.

It is easy to see, that these data have non-negligible uncertainties. Therefore, the sensitivity and uncertainty analysis of FTA is an important task. The objective of the sensitivity analysis is to show how the change in any system parameter influences the resultant reliability value of the whole system.

The paper of Tchorzewska-Cieslak and Boryczko contains the methodology of the FTA and an example of its application in order to analyze different failure scenarios in water distribution subsystem. They concluded that the FTA is particularly useful for the analysis of complex technical systems in which analysis of failure scenarios is a difficult process because it requires to examine a high number of cause-effect relationship. Undoubtedly the water distribution subsystem belongs to such systems. The FTA involves "thinking back", which allows the identification of failure events that cause the occurrence of the Top Event. In the case of very large fault trees it is advisable to use the computer methods (Tchorzewska-Cieslak & Boryczko, 2010).

Siontorou, Batzias, and Tsakiri investigated the causes of biosensors' malfunction by means of FTA and proposed a computer-aided method for diagnosing biosensor failure during operation through an algorithmic procedure that is based on a nested loop mechanism. The tree (dendritic graph) structure serves as the knowledge base decision mechanism is the inference engine for fault detection and isolation (Siontorou *et al.*, 2010).

In article of Ekaette et al., authors described the application of probabilistic Fault Tree Methods to assess the probability of radiation misadministration to patients at a large cancer treatment center (Ekaette *et al.*, 2007). To populate the FT they used subjective probabilities from experts and compared results with incident report data. Both the Fault Tree and the incident report analysis revealed simulation tasks to be most prone to incidents, and the treatment prescription task to be least prone to incidents. Ekaette et al. have demonstrated that the Fault Tree Method is useful in modeling the probability of incidents in complex medical systems. They were able to evaluate the reliability of the FTA using incident reports. The FTA helps them to understand the type of incidents that could occur and therefore supports proactive risk analysis. The discussions and analysis of possible incident pathways throughout the process of building the FT provided the medical staff better insight of the treatment system as a whole, how their individual areas of expertise and duties interrelate, the vulnerable aspects of the tasks for which they are responsible, and possible systematic interventions for better provision of care.

An algorithm of vague Fault Tree Analysis is proposed in Chang et al.'s paper to calculate fault interval of system components from integrating expert's knowledge and experience in terms of providing the possibility of failure of bottom events. This paper also modifies Tanaka's definition on fault-tree analysis and integrates vague set arithmetic for implementing fault-tree analysis on weapon system fault diagnosis (Chang *et al.*, 2006).

The Author has studied mathematical modeling of deterministic and stochastic technical systems, types of mathematical models' uncertainties and the parametrical uncertainties' investigation methods in mathematical modeling, engineering simulation and maintenance management's decision making (Pokorádi & Szabolcsi, 1999), (Pokorádi, 2008), (Pokorádi, 2009*a*), (Pokorádi, 2009*b*), (Pokorádi, 2010).

A method of FT sensitivity analysis is shown in paper of Csiba by an example of the complete railway vehicle. The determination of resultant reliability is indispensable for the investigation. The determination of the reliability and the fulfillment of sensitivity analysis will be carried out by using the failure tree method. In the investigation of Csiba, the reliability of a railway carriage has been determined. The reliability model of the railway carriage has been built by 16 main constructional units (Csiba, 2008).

One of the disadvantages of Csiba's work, that only sensitivity of Top Event's probability can be investigated in case of changing of basic events' probabilities. His method cannot analyze sensitivities of intermediate events, which can represent sensitivity of element groups' or subsystems' reliability. Another disadvantage is that calculation of sensitivity coefficient can be difficult in a real, complicate situation. (It will be shown in Chapter 3 by demonstrative example of this paper)

The main aim of this study is to elaborate an easy-used algorithm for setting-up of Linear Fault Tree Sensitivity Model (LFTSM). This modular approach process uses matrix-algebraic method that is based upon the mathematical diagnostics methodology of aircraft systems and gas turbine engines by the Author (Pokorádi, 2008). This paper will show adaptation of linear mathematical diagnostic modeling methodology for setting-up of LFTSM and its possibility of use to investigate Fault Tree sensitivity by a demonstrative example.

The outline of the paper is as follows: Section 2 shows the Fault Tree Analysis shortly. Section 3 presents a modular approach algorithm for setting-up of Linear Fault Tree Sensitivity Model theoretically and by a practical demonstration. Section 4 interprets a simply application of the LFTSM. Section 5 summaries the paper, outlines the prospective scientific work of the Author.

## 2. The Fault Tree Analysis

The Fault Tree Analysis (FTA) is a systematic, deductive (top-down type) and probabilistic risk assessment tool which shows the causal relations leading to a given undesired event, referred to as the "Top Event" (TE). The events, which cannot be subdivided, are called the Basic Events. Fault Tree diagram displays the undesired state of the investigated system (top event) in terms of the states of its components (basic events). The Fault Tree Analysis is a graphical design technique. The main result of FTA is a graph that has a dendritic structure.

The FTA can be used:

- to determine faults, fault-combinations that can occur the TE and their causes;
- to detect especially critical events and/or event-chains;
- to perform reliability and dependability investigations;
- to demonstrate failure-mechanisms illustratively.

The first step in a Fault Tree Analysis is the selection of the Top Event that is a specific undesirable system's state or failure. Then the experts should analyze the system or process to discover logical dependencies between TE and all Basic Events. To represent logical dependencies, basically the AND or OR logical gates and so-called intermediate events can be used. The intermediate events can denote subsystem's faults.

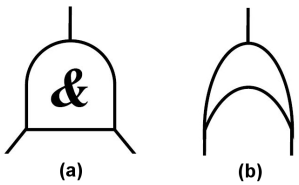


Figure 1. Logical Gates of Fault Tree.

The AND logical gate (figure 1.a) should be used if output event occurs only if all input events occur simultaneously. If output event occurs if any of the input events occur, either alone or in any combination, the OR logical gate (figure 1.b) should be used.

The Figure 2. shows a demonstrative Fault Tree. In the figure events 1; 2; 11 and 22 are intermediate events. The events 12; 21; 111; 112; 221 and 222 are Basic Events.

Probabilistic Fault Tree Analysis (PFTA) is a quantitative analysis method used to calculate the probability of TE from given failure probabilities of system components. The probability of occurrence of a (non-basic) event can be determined by probabilities of input events and the knowledge of logical gate describing connection.

In case of **AND** logical gate (Figure 1.a):

$$P = \prod_{i=1}^{n} P_i \tag{2.1}$$

where:

$P_i$, $\quad P_i \in [0,1] \subset \Re$ - probability of occurrence of $i$-th input event;

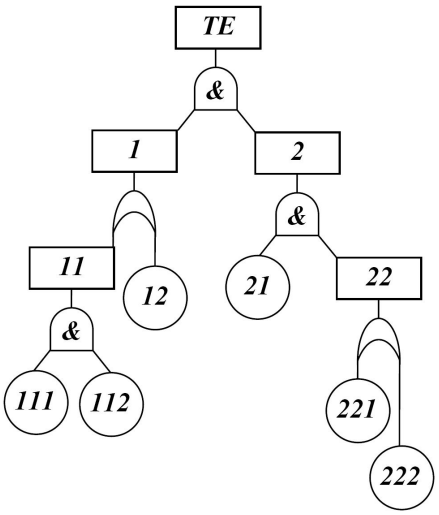


Figure 2. Fault Tree (Example).

$n$, $n \in \aleph$ - number of input events.

In case of OR logical gate (Figure 1.b):

$$P = 1 - \prod_{i=1}^{n} (1 - P_i), \tag{2.2}$$

but for two-input **OR** gate logical gate:

$$P = 1 - ((1 - P_C)(1 - P_D)) = P_C + P_D - P_C P_D. \tag{2.3}$$

Using equations (2.1), (2.2) and (2.3) the model of investigated fault tree (Figure 2):

$$P_{TE} = P_1 P_2 \tag{2.4}$$

$$P_1 = P_{11} + P_{12} - P_{11} P_{12} \tag{2.5}$$

$$P_2 = P_{21} P_{22} \tag{2.6}$$

$$P_{11} = P_{111} P_{112} \tag{2.7}$$

$$P_{22} = P_{221} + P_{222} - P_{221} P_{222}. \tag{2.8}$$

For further investigation, firstly the nominal (average, typical) probabilities of occurrence of Basic Events should be recorded. The Table 1. shows nominal values of basic events' probabilities of occurrence. Then - using equations (2.4) - (2.8) - the probabilities of intermediate events and at last probabilities of the Top Event should be determined. The Table 2. shows the results of result of the used model.

Table 1. Primary Data.

| | | |
|---|---|---|
| $P_{12} = 0,01$ | $P_{21} = 0,02$ | $P_{111} = 0,05$ |
| $P_{112} = 0,06$ | $P_{221} = 0,04$ | $P_{222} = 0,03$ |

Table 2. Probabilities Calculated by Equations (2.4) - (2.8).

| | |
|---|---|
| $P_{22} = 0,0688$ | $P_{11} = 0,003$ |
| $P_2 = 0,001376$ | $P_1 = 0,01297$ |
| $P_{TE} = 1,1784610^{-5}$ | |

For demonstration of main essence of shown modular approach methodology, the probability of TE can be determined by equation

$$P_{TE} = (P_{111}P_{112} + P_{12} - P_{111}P_{112}P_{12})(P_{21}(P_{221} + P_{222} - P_{221}P_{222})). \tag{2.9}$$

## 3. Setting-up of Sensitivity Model

For sensitivity investigation, the sensitivity model of the discussed Fault Tree should be set-up. In this chapter the method of modular approach sensitivity model setting up will be depicted theoretically and demonstrated practically by the example of the sample FT mentioned above (Figure 2).

### *3.1. Theoretical Solution*

To determine the sensitivity coefficient as a first step, the total differential of both sides of the initial equation

$$y = f(x_1, x_2, ..., x_n), f : \Re^n \to \Re, \tag{3.1}$$

should be formed:

$$dy = \frac{\partial f(x_1; x_2; \dots x_n)}{\partial x_1} dx_1 + \dots + \frac{\partial f(x_1; x_2; \dots x_n)}{\partial x_n} dx_n. \tag{3.2}$$

Then both sides of the last equation should be multiplied by same sides of the general equation and all elements should be multiplied by $\frac{x_i}{x_i}$

$$\frac{dy}{y} = \frac{\partial f(x_1; x_2; \dots x_n)}{\partial x_1} \frac{x_1}{f(x_1; x_2; \dots x_n)x_1} dx_1 + \dots + \frac{\partial f(x_1; x_2; \dots x_n)}{\partial x_n} \frac{x_n}{f(x_1; x_2; \dots x_n)x_n} dx_n \tag{3.3}$$

Introducing the sensitivity coefficients:

$$K_{y;x_i} = \frac{\partial f(x_1; x_2; \dots x_n)}{\partial x_i} \frac{x_i}{f(x_1; x_2; \dots x_n)} = \frac{\partial y}{\partial x_i} \frac{x_i}{y}, \tag{3.4}$$

a short sign is $K_i$, and

$$\frac{d\eta}{\eta} \approx \frac{\Delta\eta}{\eta} = \delta\eta \tag{3.5}$$

equation, the following linear system can be achieved:

$$\delta y = K_{y;x_1}\delta x_{y;x_2} + \dots + K_{y;x_n}\delta x_n. \tag{3.6}$$

Using equation, mentioned above, how sensitive dependent system output parameters will be to uncertainties of input ones.

The sensitivity coefficients of Fault Tree gates can be determined by the following way:

Using equation (2.1), in case of **AND** logical gate:

$$K_i = 1, \forall i \in \{1, 2, ..., n\} \tag{3.7}$$

where $n$ is the number of causer events

Using equations (2.2) and (2.3) in case of **OR** logical gate:

$$K_j = \frac{P_j}{P} \prod_{\substack{i=1 \\ i \neq j}}^{n} (1 - P_i). \tag{3.8}$$

or

$$K_C = \frac{\partial\left(1-\left(\left(1-P_C\right)\left(1-P_D\right)\right)\right)}{\partial P_C}\frac{P_C}{P} = \left(1-P_D\right)\frac{P_C}{P}. \tag{3.9}$$

Next task is to separate events of Fault Tree into basic events and non-basic (top and intermediate) ones. The probabilities of basic and non-basic events should be arranged into vectors $\mathbf{x}$ and $\mathbf{y}$. Then, the connection between probabilities of basic and non-basic events can be described by

$$\mathbf{A}\delta\mathbf{y} = \mathbf{B}\delta\mathbf{x}, \tag{3.10}$$

where $\mathbf{A} \in \Re^{n\times n}$ and $\mathbf{B} \in \Re^{n\times m}$ are coefficient matrices of basic and non-basic events of the investigated Fault Tree, and $\delta\mathbf{y} \in \Re^{n\times 1}$, $\delta\mathbf{x} \in \Re^{m\times 1}$, $\mathbf{n}$ is the number of non-basic events, $\mathbf{m}$ is the number of basic events.

Using the

$$\mathbf{D} = \mathbf{A}^{-1}\mathbf{B} \in \Re^{n\times m} \tag{3.11}$$

relative sensitivity coefficient matrix of investigated Fault Tree, the equation

$$\delta\mathbf{y} = \mathbf{D}\delta\mathbf{x} \tag{3.12}$$

can be used for sensitivity investigations.

Using equation (3.5), vector of relative changing of basic events' probabilities of occurrences can be determined by

$$\delta\mathbf{x} = \mathbf{X}_{nom}^{-1}\Delta\mathbf{x}, \tag{3.13}$$

where:

$\mathbf{X}_{nom} \in \Re^{m\times m}$ - nominal values matrix of basis events' probabilities of occurrence:

$$\mathbf{X_{nom}} = \begin{bmatrix} P_{1nom} & 0 & \dots & 0 \\ 0 & P_{2nom} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & P_{nnom} \end{bmatrix} \tag{3.14}$$

$\Delta\mathbf{x} \in \Re^{m\times 1}$ - vector of measured changing of basis events' probabilities. Knowing the vector measured changing of basis events' probabilities, by relative sensitivity coefficient matrix - equation (3.12) - the vector of relative changing of non-basic event' probabilities

$$\delta\mathbf{y} = \mathbf{D}\delta\mathbf{x} = \mathbf{D}\mathbf{X}_{nom}^{-1}\Delta\mathbf{x} \tag{3.15}$$

can be determined. Applying the nominal values matrix of non-basis events' probabilities of occurrence $\mathbf{Y}_{nom} \in \Re^{n\times n}$, the vector of measured changing of non-basis events' probabilities:

$$\Delta\mathbf{y} = \mathbf{Y}_{nom}\delta\mathbf{y} = \mathbf{Y}_{nom}\mathbf{D}\mathbf{X}_{nom}^{-1}\Delta\mathbf{x} \tag{3.16}$$

Introducing measured sensitivity coefficient matrix of investigated Fault Tree

$$\mathbf{S} = \mathbf{Y}_{nom}\mathbf{D}\mathbf{X}_{nom}^{-1}, \tag{3.17}$$

the equation (3.16) can be simplified

$$\Delta\mathbf{y} = \mathbf{S}\Delta\mathbf{x}, \tag{3.18}$$

where:

$\mathbf{Y}_{nom} \in \Re^{n\times n}$ - nominal values matrix of basis events' probabilities of occurrence:

$$\mathbf{Y}_{nom} = \begin{bmatrix} P_{1nom} & 0 & \dots & 0 \\ 0 & P_{2nom} & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & 0 & P_{mnom} \end{bmatrix}. \tag{3.19}$$

The vector of measured changing of basic events' probabilities can be determined by

$$\Delta \mathbf{x} = \mathbf{x} - \mathbf{x}_{nom} \tag{3.20}$$

where: $\mathbf{x}_{nom}$ - nominal values vector of basis events' probabilities of occurrence. The vector of measured changing of non-basic events' probabilities can be determined by

$$\Delta \mathbf{y} = \mathbf{y} - \mathbf{y}_{nom} \tag{3.21}$$

where: $\mathbf{y}_{nom}$ - nominal values vector of non-basic events' probabilities of occurrence. Using equations (3.18); (3.20) and (3.21) vector of measured values of non-basic events' probabilities can be determined by

$$\mathbf{y} = \mathbf{y}_{nom} + \mathbf{S}\left(\mathbf{x} - \mathbf{x}_{nom}\right). \tag{3.22}$$

If during a Fault Tree Analysis only the probability of TE and its sensitivities are investigated only the first row of matrix $\mathbf{S}$ will be used as vector $\mathbf{s} \in \Re^{m\times 1}$. So the changing of TE probability of occurrence can be calculated by

$$\Delta P_{TE} = \mathbf{s}^T \Delta \mathbf{x}, \tag{3.23}$$

and

$$P_{TE} = P_{TEnom} + \mathbf{s}^T \left(\mathbf{x} - \mathbf{x}_{nom}\right). \tag{3.24}$$

### *3.2. Practical Demonstration*

To demonstrate setting-up methodology mentioned above, lets' study the Fault Tree shown by Figure 2. The sensitivity coefficients of the investigated Fault Tree - using equations (2.4) - (2.8) are:

$$\delta P_{TE} = K_1 \delta P_1 + K_2 \delta P_2 \tag{3.25}$$

$$K_1 = 1, K_2 = 1$$

$$\delta P_1 = K_{11}\delta P_{11} + K_{12}\delta P_{12} \tag{3.26}$$

$$K_{11} = (1 - P_{12})\frac{P_{11}}{P_1}, K_{12} = (1 - P_{11})\frac{P_{12}}{P_1}$$

$$\delta P_2 = K_{21}\delta P_{21} + K_{22}\delta P_{22} \tag{3.27}$$

$$K_{22} = 1, K_{21} = 1$$

$$\delta P_{11} = K_{111}\delta P_{111} + K_{112}\delta P_{112} \tag{3.28}$$

$$K_{111} = 1, K_{112} = 1$$

$$\delta P_{22} = K_{221}\delta P_{221} + K_{222}\delta P_{222} \tag{3.29}$$

$$K_{221} = (1 - P_{222})\frac{P_{221}}{P_{22}}, K_{222} = (1 - P_{221})\frac{P_{222}}{P_{22}}$$

The vectors of probabilities of basic, and non- basic events are:

$$\mathbf{x}^T = [P_{12}; P_{21}; P_{111}; P_{112}; P_{221}; P_{222}], \tag{3.30}$$

$$\mathbf{y}^T = [P_{TE}; P_1; P_2; P_{11}; P_{22}]. \tag{3.31}$$

The coefficient matrices of basic, and non-basic events are:

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -0.229 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}; \tag{3.32}$$

$$\mathbf{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0,767 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,564 & 0,419 \end{bmatrix}. \tag{3.33}$$

The relative sensitivity matrix:

$$\mathbf{D} = \begin{bmatrix} 0,769 & 1 & 0,229 & 0,229 & 0,564 & 0,419 \\ 0,769 & 0 & 0,229 & 0,229 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0,564 & 0,419 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,564 & 0,419 \end{bmatrix}. \tag{3.34}$$

By tables 1. and 2. the matrices of nominal values of basic, and non-basic events are:

$$\mathbf{X}_{nom} = \begin{bmatrix} 0,01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0,02 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0,05 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,06 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,04 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0,03 \end{bmatrix}; \tag{3.35}$$

$$\mathbf{Y}_{nom} = \begin{bmatrix} 1,178 \cdot 10^{-5} & 0 & 0 & 0 & 0 \\ 0 & 1,297 \cdot 10^{-2} & 0 & 0 & 0 \\ 0 & 0 & 1,376 \cdot 10^{-3} & 0 & 0 \\ 0 & 0 & 0 & 3 \cdot 10^{-3} & 0 \\ 0 & 0 & 0 & 0 & 6,88 \cdot 10^{-2} \end{bmatrix}. \tag{3.36}$$

The vectors of nominal values of basic, and non- basic events are:

$$\mathbf{x}_{nom}^T = \begin{bmatrix} 0,01 & 0,02 & 0,05 & 0,06 & 0,04 & 0,03 \end{bmatrix}, \tag{3.37}$$

$$\mathbf{y}_{nom}^T = \begin{bmatrix} 1,178 \cdot 10^{-5} & 1,297 \cdot 10^{-2} & 1,376 \cdot 10^{-3} & 3 \cdot 10^{-3} & 6,88 \cdot 10^{-2} \end{bmatrix}. \tag{3.38}$$

The measured sensitivity matrix of the demonstrative FT (Figure 2):

$$\mathbf{S} = \begin{bmatrix} 1,372 \cdot 10^{-3} & 8,923 \cdot 10^{-4} & 8,173 \cdot 10^{-5} & 6,811 \cdot 10^{-5} & 2,516 \cdot 10^{-4} & 2.490 \cdot 10^{-4} \\ 0,997 & 0 & 5,940 \cdot 10^{-2} & 4,950 \cdot 10^{-2} & 0 & 0 \\ 0 & 6,880 \cdot 10^{-2} & 0 & 0 & 1,940 \cdot 10^{-2} & 1,940 \cdot 10^{-2} \\ 0 & 0 & 5,999 \cdot 10^{-2} & 0,500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0,970 & 0,960 \end{bmatrix}, \tag{3.39}$$

and measured sensitivity vector of TE:

$$\mathbf{s}^T = \begin{bmatrix} 1,372 \cdot 10^{-3} & 8,923 \cdot 10^{-4} & 8,173 \cdot 10^{-5} & 6,811 \cdot 10^{-5} & 2,516 \cdot 10^{-4} & 2.490 \cdot 10^{-4} \end{bmatrix}. \tag{3.40}$$

### 3.3. Discussion

Using method shown by reference (Csiba, 2008), the sensitivity coefficients can be determined by derivations - see equations (3.1) - (3.4)) - of equation (2.9). It is easy to see that this task can be difficult or error-risky for engineers or technical expert (who is not mathematician) in a real, complicate Fault Tree Analysis. Because in cases of real system reliability analysis, the Fault Tree can has more then 5 ~ 6 levels of intermediate events. In addition the result of sensitivity investigation shows only the Top Event's sensitivity. The analyzers cannot gather information about the sensitivities of intermediate events that is reliability and dependability of element groups or subsystems. This result can be got using shown matrix algebraic method and equation (3.24).

On the other hand, advantages of the modular approach method shown above are followings:

The connections between input and output event of all logical gates can be written easily - using equations (2.1), (2.2) or (2.3). Using these connections, the elements of vectors and matrices of nominal values of basic ($\mathbf{x}_{nom}$ and $\mathbf{X}_{nom}$) and non-basic ($\mathbf{y}_{nom}$ and $\mathbf{Y}_{nom}$) events' probabilities can be determined.

Therefore, employing equations (3.6) or (3.7), the sensitivity coefficients of all logical gates as modules of investigated Fault Tree that is elements of coefficient matrices of basic ($\mathbf{A}$) and non-basic ($\mathbf{B}$) events can be determined easily.

After determination of required vectors and matrices, the experts can use matrix-algebraic method to analyze sensitivity characters of given Fault Tree from given investigational point of view. (The next Chapter will show a simply example of possibility of use Linear Fault Tree Sensitivity Model.)

## 4. Application of Linear Fault Tree Sensitivity Model

The Linear Fault Tree Sensitivity Model (LFTSM) elaborated and set up above can be used to investigate reliability and dependability of a technical system analyzed by Fault Tree.

Knowing the sensitivity model of Fault Tree, sensitivity of the Top Event and intermediate events can be investigated by modification of vector of basic events' probabilities (as independent variables)$\Delta\mathbf{x}$. Results of sensitivity analysis can be used for conclusions to come about features of the given system and its behavior in case of simulated basic events' probability changing. This changing of probability can be produced by constructional modification of given element that occur the modeled probability changing, of investigated technical system.

The essence of the sensitivity analysis is that the anomalies and variations of dependent system parameters are simulated by changing of its independent (input and inner) variables. On the basis of the mathematical model of the investigated system can be determined how sensitive dependent system variables will be for simulated changes. If only one of independent variables is changed, the investigation will be called one-parameter sensitivity analysis. If the number of the changed independent variables is more than one, the several-parameter sensitivity analysis is used.

It is important to mention that changes of independent variables cannot be more than about 1 or 5 %, depending on the intensity of the original model's nonlinearity. Depending on the nonlinearity of the original model, results of the sensitivity analysis can have difference from real influences of simulated changes. But these results show the direction and order of the magnitude of real simulated changes.

Six one-parameter sensitivity investigations of the Fault Tree shown by figure 2. - using equations (3.24) and (3.40) - were performed. During investigations probabilities of every basic events were decreased (in other words reliability of all components were improved) by 0,005.

The Figure 3. and Table 3. show the results. It is seeable, that effects of basic events' probabilities 12 and 21 are the largest to the probability of Top Event reliability that is reliability of the system.

Table 3. Results of one-parameter sensitivity investigations of Top Events.

| Event | 12 | 21 | 111 | 112 | 221 | 222 |
|---|---|---|---|---|---|---|
| Nominal | 1,17846 $10^{-5}$ | | | | | |
| Modeled | 4,925 $10^{-06}$ | 7,323 $10^{-06}$ | 1,138 $10^{-05}$ | 1,144 $10^{-05}$ | 1,053 $10^{-05}$ | 1,054 $10^{-05}$ |

The Figure 4 and Table 4 show the results of increasing by 0,005 of basic event 12 probability (from 0,01 to 0,015). In the diagram can be seen that only probability of Top Event and intermediate event 1 will change on account of increasing of probability of basic event 12. It is important to mention that axe calibration of probability is a logarithmic one.

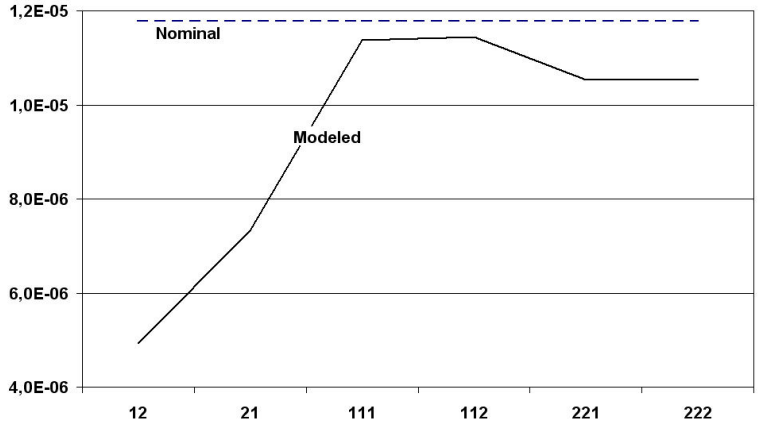


Figure 3. Results of one-parameter sensitivity investigation of Top Events.

Table 4. Results of one-parameter sensitivity investigation ($\Delta P_{12} = 0,005$).

| Event | TE | 1 | 2 | 11 | 22 |
|---|---|---|---|---|---|
| Nominal | 1,17846 $10^{-5}$ | 0,01297 | 0,001376 | 0,003 | 0,0688 |
| Modeled | 8,64403 $10^{-05}$ | 0,017955 | 0,001376 | 0,003 | 0,0688 |

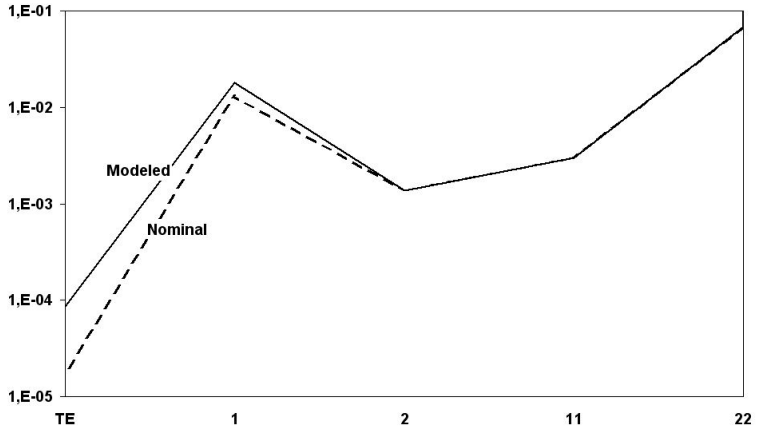


Figure 4. Results of one-parameter sensitivity investigation ($\Delta P_{12} = 0,005$).

## 5. Conclusions

This paper discussed a new sensitivity investigation method of Fault Tree Analysis elaborated by the Author. The paper showed the adaptation of linear mathematical diagnostic modeling methodology for setting-up of Linear Fault Tree Sensitivity Model (LFTSM) The LFTSM is a modular approach tool that uses matrix-algebraic method based upon the mathematical diagnostics methodology of aircraft systems and gas turbine engines. In this paper the possibility of use of LFTSM was demonstrated to investigate Fault Tree sensitivity by a simply example. Using demonstrated mathematical connections and procedure, the technical experts can get a easy-used methodology to build up sensitivity model of the given FT. This Linear Fault Tree Sensitivity Model (LFTSM) can be used to investigate system reliability and dependability from required point of view.

During prospective scientific research related to this field of applied mathematics and maintenance management decision making, the Author would like to work out methodologies of Fault Tree uncertainty investigation using other

mathematical tools, for example linear interval equations, Monte-Carlo simulation and fuzzy set theory, on basis of Linear Fault Tree Sensitivity Model (LFTSM).

## References

Chang, J.-R., K.-H. Chang, S.-H. Liao and C.-H. Cheng (2006). The reliability of general vague fault-tree analysis on weapon systems fault diagnosis. *Soft Comput.* **10**, 531–542.

Csiba, J. (2008). Sensitivity analysis of the reliability computed by using the failure tree method. In: *Proceedings of Proc. Of the 11th Mini Conference on Vehicle System Dynamics, Identification and Anomalies, Budapest*. pp. 749–760.

Ekaette, E. E., R. C. Lee, D. L. Cooke, S. Iftody and P. Craighead (2007). Probabilistic fault tree analysis of a radiation treatment system. *Risk Analysis* **27**, 1395–1410.

IEC (1990). Standard IEC 1025 fault tree analysis (FTA). Technical Report 39. International Electrotechnical Commission.

Pokorádi, L. (2008). *Systems and Processes Modeling*. Campus Kiadó, Debrecen, 242. (in Hungarian).

Pokorádi, L. (2009*a*). Uncertainties of mathematical modeling. In: *Proceedings of the 12th Symposium of Mathematics and its Applications. Timişoara, Romania, 2009.11.05-2009.11.07. Politehnica, University of Timişoara*. pp. 471–476.

Pokorádi, L. (2009*b*). Uncertainty of manufacturing simulation. *Academic Journal of Manufacturing Engineering* (7), 54–59.

Pokorádi, L. (2010). Uncertainties of engineering simulation. In: *Proceedings of International Conference on Innovative Technologies IN-TECH 2010, Prague, Czech Republic*. pp. 121–124.

Pokorádi, L. and R. Szabolcsi (1999). *Mathematical Models Applied to Investigate Aircraft Systems. Monographical Booklets in Applied and Computer Mathematics, MB-12*. PAMM, Müegyetemi Kiadó, Budapest.

Siontorou, Ch.G., F.A. Batzias and V. Tsakiri (2010). A knowledge-based approach to online fault diagnosis of fet biosensors. *IEEE Transactions on Instrumentation and Measurement.* **59**(9), 2345–2364.

Tchorzewska-Cieslak, Barbara and Krzysztof Boryczko (2010). Analysis of undesirable events scenarios in water supply system by means of fault tree method. *Journal of Konbin* **14-15**, 309–320.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 45–62

# The design and scheduling of chemical batch processes: building heuristics and probabilistic analysis

João Miranda[a,*]

[a] *College of Technology and Management, Portalegre Polytechnics Institute, Lugar da Abadessa, Apt. 148; 7301-999 Portalegre, Portugal. Centre for Chemical Processes, Instituto Superior Técnico, Technical University of Lisbon.*

## Abstract

The number of industrial cases published in the design and scheduling of batch multiproduct plants is short, and the difficulties to solve large models of this kind are well known, since their modeling usually consider variables integrality and data uncertainty. One way to address such difficulties is to use analytical studies to obtain significant improvements in algorithms and problem structures. Several MILP models from the open literature are selected focusing the successive generalizations on the options set, namely: from single machine to multiple parallel machines (identified by ***S*** or ***M***) in each stage; and from single product campaigns to multiple products campaigns (***S*** or ***M*** too). Four models (hereby ***SS, MS, SM, MM***) that consider zero wait operations are thus analyzed and compared, and several heuristics are developed in order to produce good approximations to the objective function's value and to the binary solution. Then, the probabilistic analysis of the heuristics was performed: the deviations on the objective function values, the deviations on the binary solutions, and the computational times are evaluated. The analysis both allowed the certification of the modeling and the numerical implementation. The model MS, addressing multiple machines and SPC, is found to be the most promising model to further developments that aim the design and scheduling of batch processes in stochastic and robust frameworks.

*Keywords:* Design and scheduling, Batch processes, Heuristics, Probabilistic analysis.
*2000 MSC:* 93C42, 93C10, 37M05.

## 1. Introduction

The problem of scheduling batch processes presents a large number of industry applications, such as in manufacturing systems, mechanics, electronics, and in chemical industry. In addition, the batch processes in biotechnology, food industry, fine chemicals, and pharmaceutics usually apply restrictive conditions of temperature and pressure, quality, safety, and quite expensive equipments are thus required. Considering their cost, the maximization of the equipment utilization is important to the economic efficiency of the chemical batch plants, and this subject is commonly related to the minimization of makespan.

In the design and scheduling of batch chemical processes, it is widely accept that the design problem must include the scheduling subjects. It is well known that the production policies, the storage policies, and the sequencing of the production are quite important to the economic efficiency of the batch plants (Barbosa-Póvoa, 2007). Then, the design problem simultaneously considers the solution of detailed scheduling sub-problems, anticipating the operation modes early at the design stage.

It must be well defined if the model is focusing to support decisions of economic type, concerning the treatment of external elements (purchases, sales, investments), or if it is aiming the internal decisions related to the batch operations (setup and hold-on times, maintenance, control), in close relation with the given time horizon (Romero

*Corresponding author
*Email address:* jlmiranda@estgp.pt (João Miranda)

et. al., 2003; Moreno et. al., 2007). Usually, external decisions consider uncertainty and risk treatments through stochastic framework, while internal decisions are addressed by logical relations and applying integer and binary variables.

The simultaneous approach, considering both the external and stochastic subjects and the internal and integrality requirements, is very difficult to solve (Pekny, 2002). By other side, the implementation of MILP or MINLP deterministic models is common in the design of flowshop batch plants (Barbosa-Póvoa, 2007), with the purposes of selecting and sizing equipments. One way to address the difficulties is to use analytical studies and computational complexity to gain insight (Liu and Sahinidis, 1997; Ahmed(Ahmed & Sahinidis, 2000) and (Ahmed & Sahinidis, 2003), and to obtain significant improvements in algorithms and problem structures. Also, the application of heuristics, tabu search (Cavin *et al.*, 2004), ant colony procedures (Jayaraman *et al.*, 2000), or evolutionary algorithms (Tan & Mah, 1998), is spreading in the design and scheduling of batch chemical processes.

Consequently, several deterministic MILP models from the open literature are selected (Voudouris & Grossmann, 1992), they are analyzed and compared, and heuristics that can be useful in stochastic framework are then developed. The selected models of the design and scheduling of batch chemical processes are focusing successive generalizations in the options set, but all consider zero wait policy (ZW) and flowshop environment. The options set is considering variations in the number of processes implanted in each production stage (single machine vs. multiple machine, ***S*** or ***M***), and in the mode of production campaign, single versus multiple product campaign (SPC *vs*. MPC, ***S*** or ***M***). In a combinatory way, the four alternative models **(SS, MS, SM, MM)** are analyzed, (Miranda, 2007) showed that are occurring equivalence relations between specific instances of them, and these models thus belong to the same class of computational complexity.

Several heuristic procedures are developed, in order to produce good quality approximations to the objective function value, or to build the optimal or a near optimal binary solution. The probabilistic analysis of the heuristics is performed, evaluating the deviations to the objective value and to the binary solution, and also comparing the computational times. Numerical experiences are described, related not only to the heuristics analysis, but also to certify the modeling and the numerical implementation of the several models. This is an important point, due to some modeling inconsistencies that drives solutions incoherence, as described later.

The structure of the paper considers: in Section 2, the mathematical models at hand are presented; in Section 3, the heuristic procedures focusing model ***SS*** (single machine and SPC) are developed, and the results are analyzed; in Section 4, the heuristic procedures onto model MS (multiple machine and SPC) are developed and analyzed; in Section 5, the models ***SM***(single machine and MPC) and ***MM*** (multiple machine and MPC) are addressed; in Section 6, the main conclusions are presented.

## 2. The models for design and scheduling of batch chemical processes

The issue of scheduling batch processes is embedded in the process design, in a way to conjugate the short term decisions from the operational planning within the long term investment planning. The models are collected from the open literature (Voudouris & Grossmann, 1992), and they are selected based in their character of successive generalizations in the options set: *i)* from single machine to multiple machines working in parallel at each stage; *ii)* from SPC to MPC. The search space is thus enlarged, from one single option to many options in each of the subjects, and the solutions are expected to be qualitatively and quantitatively improved.

The four models in analysis are considering the flowshop and ZW contexts, in the following sense:

- The flowshop problem (Garey *et al.*, 1976) considers that N jobs are to be processed on M stages sequentially and it assumes: one machine is available at each stage; one job is processed on one machine at a time without preemption; and one machine processes no more than one job at a time.
- The zero wait (ZW) storage mode considers that the materials produced in one stage are directly entering in the next stage without any storage (unlimited intermediate storage, UIS), or even without waiting inside the equipment until the next stage is available to process them (no intermediate storage, NIS).

The models are described successively in this section, using a generalization approach that shows the successive enlargement of the options set. The MILP model ***SS*** considers one single machine in the several production stages,

SPC and ZW operations, and the discrete enumeration of equipments.

***Model SS***

**1-a**
$$[\min] \quad z = \sum_{j=1}^{M} \sum_{s=1}^{NS(j)} c_{js} y_{js}$$

subject to

**1-b**
$$S_{ij} Q_i CT_i \sum_{s=1}^{NS(j)} \frac{y_{js}}{dv_{js}} - t_i \leq 0\,, \ \forall i, j$$

**1-c**
$$\sum_{s=1}^{NS(j)} y_{js} = 1, \forall j$$

**1-d**
$$\sum_{i=1}^{N} t_i \leq H$$

**1-e, f**
$$t_i \geq 0\,, \ \forall i; y_{js} \in \{0; \ \ 1\}\,, \ \ \forall j, s$$

The MILP model ***MS*** is enlarging model ***SS*** by addressing multiple machines *NP(j)* in parallel in stages *j*, in a way to reduce the cycle time $CT_i$ of each product *i*. In SPC and ZW modes, the cycle time corresponds to the maximum of the processing times of each product considering all stages *j* (Biegler *et al.*, 1997):

**2**
$$CT_i = \max_{\substack{j=1..M \\ i=1..N}} \left\{ \tau_{ij} \middle/ NP(j) \right\}$$

Thus, when a second machine is introduced on a specific stage, the processing times in this stage are reduced to half, and so on. The discrete enumeration of equipments allows that the normalized sizes of processing equipments are included.

**Model *MS***

**3-a**
$$[\min] \quad z = \sum_{j=1}^{M} \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} c_{jsp} y_{jsp}$$

subject to

**3-b**
$$S_{ij} Q_i \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} \frac{y_{jsp}}{dv_{js}} - n_i \leq 0, \ \forall i, j$$

**3-c**
$$\sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} y_{jsp} = 1, \ \forall j$$

**3-d**
$$\sum_{i=1}^{N} t_i \leq H$$

**3-e**
$$\sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} \left( \frac{\tau_{ij}}{p} nc_{ijsp} \right) - t_i \leq 0, \forall i, j$$

**3-f**
$$nc_{ijsp} - nc_{ijsp}^{Upp} y_{jsp} \leq 0\,, \ \forall i, j, s, p$$

**3-g**
$$\sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} nc_{ijsp} - n_i = 0,\ \forall i, j$$

**3-h,i**
$$nc_{ijsp},\ n_i,\ t_i \geq 0,\ \forall i, j, s, p; y_{jsp} \in \{0;\ 1\},\ \forall j, s, p$$

When transition times (for clean-up, setup, or tuning) between different products are short, it is useful to consider a mix of products running in the same production campaign. The adoption of MPC considers the sequencing of different products with the purpose to reduce idle times, when the equipments are not in use. This way, the global cycle time can be minimized and the equipment utilization can be improved. With the same characteristics of flowshop and ZW, (Voudouris & Grossmann, 1992) presented the model considering one single machine at each stage and MPC production policy, hereby referred by ***SM***.

**Model *SM***

**4-a**
$$[\min]\quad z = \sum_{j=1}^{M} \sum_{s=1}^{NS(j)} c_{js} y_{js}$$

subject to

**4-b**
$$S_{ij} Q_i \sum_{s=1}^{NS(j)} \frac{y_{js}}{dv_{js}} - n_i \leq 0,\ \forall i, j$$

**4-c**
$$\sum_{s=1}^{NS(j)} y_{js} = 1, \forall j$$

**4-d**
$$\sum_{k=1}^{N} Nch_{ik} = n_i,\ \forall i$$

**4-e**
$$\sum_{i=1}^{N} Nch_{ik} = n_k,\ \forall k$$

**4-f**
$$Nch_{ii} \leq n_i - 1,\ \forall i$$

**4-g**
$$\sum_{i=1}^{N} \left[ n_i \tau_{ij} + \sum_{k=1}^{N} SL_{ikj} Nch_{ik} \right] \leq H,\ \forall j$$

**4-h,i**
$$n_i,\ Nch_{ik} \geq 0,\ \forall i,\ k; y_{js} \in \{0; 1\},\ \forall j, s$$

The model ***SM*** is then directly enlarged by considering the implementation of multiple machines in parallel at each stage and neglecting the transition times. Thus, one more dimension or group of variables is incorporated in the problem: the number of processes p(j) considered in each stage *j*. In substitution of the restrictions **4-g**, related to the satisfaction of the horizon time in each stage *j*, the following restrictions are introduced:

**5**
$$\sum_{i=1}^{N} n_i \tau_{ij} \leq H. \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} p(j) y_{jsp},\ \forall j$$

This substitution is developed in flowshop and ZW contexts, and the model ***MM*** (Voudouris & Grossmann, 1992) focused multiple machines and MPC.

**Model *MM***

**6-a**
$$[\min] \quad z = \sum_{j=1}^{M} \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} c_{jsn} y_{jsp}$$

subject to

**6-b**
$$S_{ij} Q_i \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} \frac{y_{jsp}}{dv_{js}} - n_i \leq 0, \ \forall i, j$$

**6-c**
$$\sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} y_{jsp} = 1, \forall j$$

**6-d**
$$\sum_{i=1}^{N} n_i \tau_{ij} - H. \sum_{s=1}^{NS(j)} \sum_{p=1}^{NP(j)} p(j) y_{jsp} \leq 0, \ \forall j$$

**6-e**
$$\sum_{k=1}^{N} Nch_{ik} = n_i, \ \forall i$$

**6-f**
$$\sum_{i=1}^{N} Nch_{ik} = n_k, \ \forall k$$

**6-g**
$$Nch_{ii} \leq n_i - 1, \ \forall i$$

**6-h**
$$\sum_{i=1}^{N} \left[ n_i \tau_{ij} + \sum_{k=1}^{N} SL_{ikj} Nch_{ik} \right] \leq H, \ \forall j$$

**6-i,j**
$$n_i, \ Nch_{ik} \geq 0, \ \forall i, \ k; y_{jsp} \in \{0; \ 1\}, \ \forall j, s, p$$

## 3. Model SS analysis

The model **SS** (single machine and SPC) searches how to choose the smallest discrete size (the cheapest) equipment to satisfy products demand $Q_i$, on a given time horizon H, knowing that the cycle time $CT_i$ of each product campaign (SPC) is size-dependent.

In Table 1, six numerical examples (A series) are specified in terms of the parameters of the model ***SS***, numbers of binary variables, continuous variables, and restrictions, and the mean of the solving times for the100 instances of each example. It was used the OSL solver provided within GAMS, running on an ASUS-F3JC (Intel Core2 T5500, 1.55GHz). In the example EX3A, the solver time (in *italic*) is related to a large number of unfeasible instances: when infeasibility occurs, the corresponding execution is stopping before the instance's optimum is reached.

In model ***SS***, some simplifications were important in the formulation of the horizon time $H$, the production times $t_i$, and in the cycle times $CT_i$. However, when comparing it with other models it must be noted that:

- The time slacks $SL_{ikj}$ that represent transitions between different products were neglected;
- In each production campaign, the head and tail are neglected;
- The overlapping between campaigns of different products is avoided, assuming that one product $k$ begins its SPC production only after the former product $i$ was finished.

The approximation procedures for the heuristic aiming the optimal solution of model ***SS*** are described in the following steps:

Table 1. Numbers of variables and restrictions in the numerical examples of model ***SS*** (A series).

| **Examples** | **Parameters** (N, M, NS) | **Binary variables** | **Continuous variables** | **Restrictions** | **Time *solver* (s)** |
|---|---|---|---|---|---|
| EX1A | 2, 3, 4 | 12 | 2 | 9 | 0.23 |
| EX2A | 3, 4, 5 | 20 | 3 | 17 | 0.36 |
| EX3A | 5, 4, 6 | 24 | 5 | 25 | 0.28 |
| EX4A | 6, 4, 10 | 40 | 6 | 29 | 0.76 |
| EX5A | 8, 5, 12 | 60 | 8 | 46 | 1.03 |
| EX6A | 10, 5, 15 | 75 | 10 | 56 | 2.17 |

1. Feasibility's previous step;
2. Initial guess;
3. Building feasible solutions:
   a) Using proportionality to demands; and
   b) Reducing production times, through *iterated local search (ILS)*

**Step 1) Feasibility's previous step**

In order to ensure solutions feasibility, a previous procedure is developed, *PREVIOUS_SS*. It assumes that the time constraints must be satisfied when the largest size *NS(j)* is selected in all the stages *j*, by assigning:

$$s'(j) = NS(j), \ \forall j \tag{7}$$

The assumption is checked: if positive, the execution continues; else the single machine instance is unfeasible.

**Step 2) Initial guess**

The procedure *APROX0_SS* assumes that the time horizon *H* must be satisfied by the sum of production times ti in all the stages *j*. Then, the minimum volume *V_h0* that is required in each stage *j* is estimated, and the enumeration of sizes is avoided:

$$H = \sum_{i=1}^{N} \frac{S_{ij} Q_i CT_i}{V\_h0(j)}, \ \forall j \ \Rightarrow \ V\_h0(j) = \sum_{i=1}^{N} \frac{S_{ij} Q_i CT_i}{H}, \ \forall j \tag{8}$$

The estimated values of volumes are adjusted to the discrete size immediately above, and the corresponding binary variables are assigned with the value 1:

$$y\_h0(j, s'(j)) = 1 \tag{9}$$

The other binary variables are assigned to 0, and the solution is checked using the procedure *VERIFY_SOLUTION_SS* to evaluate the costs, variables, and restrictions satisfaction. If the solution is unfeasible, then go to Step 3b.

**Step 3) Building feasible solutions**

The procedure *APROX1A_SS* builds feasible solutions by allocation of the production times $t_i$ in proportionality with the demand quantities, $Q_i$, for each product *i*. In plus, facing an unfeasible solution (e.g., obtained from *APROX0* in *Step 2*), the procedure *APROX1B_SS* performs a local search (ILS) to reduce the production times $t_i$ in close relation with the equipment costs $c_{js}$.

**Step 3a) Procedure *APROX1A_SS***

For each one of the products *i*, the production time $t_i$ is made proportional to its demand $Q_i$, and the time horizon *H* is partitioned among the various products in the same rate as the demand rate $\left(Q_i/Q_{total}\right)$ occurs:

**10**
$$\begin{cases} Q_{total} = \sum_{i=1}^{N} Q_i \\ t_{total} = \sum_{i=1}^{N} t_i \leq H \end{cases}$$

If no preference of products is specified in terms of the prices, profits, availabilities of raw materials, or other attributes, then,

**11**
$$\frac{t_i}{t_{total}} = \frac{Q_i}{Q_{total}} \Rightarrow \frac{t_i}{Q_i} = \frac{t_{total}}{Q_{total}} \leq \frac{H}{Q_{total}}.$$

The proportional times approach is driving feasible solutions, given that:

**12**
$$t_i \leq Q_i \frac{H}{Q_{total}} \Rightarrow \sum_{i=1}^{N} t_i \leq \sum_{i=1}^{N} \left(Q_i \frac{H}{Q_{total}}\right) \leq H.$$

The procedure checks all the products i, using the proportionality condition,

**13**
$$\frac{t_i}{Q_i} = \max_j \left\{ \frac{S_{ij} CT_i}{dv_{j,s'(j)}} \right\} \leq \frac{H}{Q_{total}}, \forall i$$

and, in each stage *j*, the shorter discrete volume $dv_{j,s(j)}$ that verifies the inequality is then selected.

However, unfeasibility can occur during the selection of the discrete sizes. If a very large ratio arises from the problem parameters, the time horizon H may be overpassed even selecting the largest size *NS(j)* of the equipment, namely, in case of:

**14**
$$\max_j \left\{ \frac{S_{ij} CT_i}{dv_{j,NS(j)}} \right\} > \frac{H}{Q_{total}}, \exists (i, j).$$

**Step 3b) Procedure *APROX1B_SS* (Local Search)**

Initiating with an unfeasible solution, the procedure *APROX1B_SS* searches to reduce the production times ti and then to satisfy the time horizon *H*. The reduction is performed in conformity with the equipment costs $c_{jS}$, since the selection of the equipment that increases in size is based in best ratio between the corresponding variations in production times and variations in costs.

The evaluation of the costs increase is direct, by direct subtraction between the corresponding costs.

To evaluate the impact of selecting larger equipment in production times, it must be noted that the cycle time $CT_i$ for each product *i* corresponds to the maximum of the processing times considering the stages *j* (the bottleneck stage). The stage where this maximum value occurs is the stage corresponding to the minimum value for the batches dimension $B_i$. The bottleneck stage is referred hereby by the *critical stage* $jcrit1_i$, and when larger equipment is selected on it, the bottleneck will then occur in other stage, $jcrit2_i$. The latter can be picked up in the descending order of the production times, from $tcrit1_i$ to $Bcrit2_i$, or in the ascending order of batches dimension, respectively, from $Bcrit1_i$ to $Bcrit2_i$. When larger equipment is selected in a *critical stage*, the total variation in the production times is obtained by the sum of all the variations in the production times, and these variations are estimated considering the variations in batches dimensions $B_i$.

It is known that

**15**
$$t_i = \max_j \left\{ \frac{S_{ij} Q_i CT_i}{dv_{j,s'(j)}} \right\} = Q_i CT_i \max_j \left\{ \frac{S_{ij}}{dv_{j,s'(j)}} \right\} = n_i CT_i, \forall i$$

and considering

$$n_i = Q_i \max_j \left\{ \frac{S_{ij}}{dv_{j,s'(j)}} \right\} = \frac{Q_i}{B_i},\ \forall i \Rightarrow B_i = \min_j \left\{ \frac{dv_{j,s'(j)}}{S_{ij}} \right\},\ \forall i \tag{16}$$

then the batches dimension, $B_i$, are obtained from the production times, $t_i$, and the number of batches, $n_i$, being these values defined accordingly with the present critical stage. The variation in production times is then expressed by:

$$\begin{aligned} \Delta t_i &= tcrit1_i - tcrit2_i \\ &= \frac{Q_i CT_i}{Bcrit1_i} - \frac{Q_i CT_i}{Bcrit2_i} = \frac{Q_i CT_i}{Bcrit1_i} - \frac{Q_i CT_i}{Bcrit1_i}\left(\frac{Bcrit1_i}{Bcrit2_i}\right) \\ &= \frac{Q_i CT_i}{Bcrit1_i}\left(1 - \frac{Bcrit1_i}{Bcrit2_i}\right) = tcrit1_i\left(1 - \frac{Bcrit1_i}{Bcrit2_i}\right),\ \forall i \end{aligned} \tag{17}$$

When an unfeasible solution is treated in the procedure *APROX1B_SS*, then for each product *i*:

i) One larger size is selected in a given critical stage for product i; the variation in costs is calculated;
ii) The sum of the variations in production times is evaluated;
iii) The variations in times and in equipment costs are compared, using a ratio;
iv) The best ratio is selected, and the corresponding discrete size is increasing, by swapping the related binary variables (assign 0 to the existing selection, and assign 1 to the next larger size);
v) To ensure feasibility, this binary solution is verified using the procedure *VERIFY_SOLUTION_SS*;
vi) If the solution is unfeasible and the iterations limit is not reached, then return to step i).

The procedure *APROX1B_SS* is executed iteratively until a feasible solution is achieved and it performs an *iterated local search* (ILS) considering:

- The searching neighborhood is defined by the increasing sizes of each of the critical stages, $jcrit1_i$;
- The evaluation function corresponds to the ratio between the variations in production times and the variations in costs;
- The alteration in the solution is defined by the neighbor solution with best (maximum) value for the described ratio;
- The termination criteria are stating the end of the procedure when a feasible solution is found or the number of iterations is reached.

The heuristic for model ***SS*** integrates the procedures from the feasibility test to the construction of feasible solutions, passing by the initial guess, the proportionality on times, and the solution verification. The values of several metrics (formulated in Appendix A.3) related to the deviations in the objective function values and in the binary solutions are presented in Table 2, namely:

- Objective function - percentage of feasible instances that are exactly estimated // percentage of unfeasible instances // percentage mean of the errors on the sub-optimal estimations // standard deviation on the percentage mean of the errors;
- Binary solutions - percentage of instances presenting exact binary solution // mean on the number of binary errors for the sub-optimal estimations.

The heuristic for model ***SS*** is satisfactory since:

- if the instance is feasible, then one solution is always obtained;
- optimal solutions are obtained in nearly 63% of the feasible instances;
- sub-optimal solutions present good quality, the errors are about 1%-2% in value, and on average nearly only one discrete size is switched.

Table 2. Computational times and deviations for heuristic of model ***SS***.

| Examples | Time *solver (s)* | Time heuristic (s) | Objective Function Deviations | Binary Solution Deviations |
|---|---|---|---|---|
| EX1A | 0.23 | 0.0006 | 91.// 0.// 6.8// 4.4 | 91.// 2.22 |
| EX2A | 0.36 | 0.0028 | 72.// 8.// 2.4// 1.7 | 72.// 1.96 |
| EX3A | 0.28 | 0.0005 | 73. // 56.// 1.2// 0.9 | 73.// 1.58 |
| EX4A | 0.76 | 0.0016 | 50. // 4. // 1.5// 1.4 | 50.// 1.79 |
| EX5A | 1.03 | 0.0020 | 30.// 0.// 1.5// 1.5 | 30.// 2.43 |
| EX6A | 2.17 | 0.0056 | 53.// 0.// 1.1// 0.9 | 53.// 2.00 |

Notwithstanding the model ***SS*** showed some weaknesses:

- Unfeasibility occurred on a large number of instances, namely, when the number of products N is increased, or for large values of the demands Q;
- The transition, setup, and cleanup times were not considered;
- The number of batches is neither specified, nor its integrality required.

As described, the model ***SS*** presents some weaknesses that limit its application onto industry's real cases and the design approach is enlarged onto multiple machines in each stage.

## 4. Model MS analysis

The model ***MS*** (multiple machine and SPC) searches how to choose the cheaper combination of discrete sizes to satisfy products demand Q, on the horizon H, but considering processes in parallel at each stage.

This model allows the implementation of equipments working in parallel in each production stage, supposing that the cycle times can be reduced in concordance with the reduction of the processing times in those stages with multiple machines.

The model ***SS***, with one single machine in each stage, may be addressed as a specific instance in the model MS if the number of processing machines is constrained to $NP = 1$ (Miranda, 2007), and equivalence relations thus occur between variables, objective function, and restrictions of the two models. This theoretic equivalence is also used to validate the computational runs: the machines number is supposed 1 in all the stages of MS and two series of 100 random samples (Appendix A) are treated. Then, the results are compared, and the equivalence between corresponding instances was numerically verified.

In Table 4, the parameters of the numerical examples for model ***MS*** are presented, so as the average of the solution times.

Table 3. Numbers of variables and restrictions in the numerical examples of model ***MS*** (A series).

| Examples | Parameters (N, M, NS, NP) | Binary variables | Continuous variables | Restrictions | Time *solver*(s) |
|---|---|---|---|---|---|
| EX1A | 2, 3, 4, 2 | 24 | 52 | 70 | 0.30 |
| EX2A | 3, 4, 5, 3 | 60 | 186 | 221 | 10.53 |
| EX3A | 5, 4, 6, 3 | 72 | 370 | 425 | 101.67 |
| EX4A | 6, 4, 10, 3 | 120 | 732 | 797 | 405.99 |
| EX5A | 8, 5, 12, 3 | 180 | 1456 | 1566 | (4098.31) |
| EX6A | 10, 5, 15, 3 | 225 | 2270 | 2406 | (15142.36) |

Given the fast increase of solving times, a time limit of 1 hour (3600 s) is stated. The examples EX5A and EX6A largely overpassed this time limit, and their mean times are taken from 30 instances and presented in-between round

brackets. In comparison with the solution times for model ***SS***, the cause of the increasing in MS times relies not only in the larger number of binary variables, but also in the much larger number of continuous variables.

These computational difficulties are strengthening the necessity of a heuristic aiming the model ***MS***. The heuristic is considering the following procedures:

1. Feasibility's previous step;
2. Initial guess;
3. Building feasible solutions;
   a) If feasible, solving by ***SS*** heuristic;
   b) Reducing production times, through ILS;
4. Tuning of feasible solutions;
   a) Reducing discrete sizes, through ILS.

**Step 1) Feasibility's previous step**

A previous procedure *PREVIOUS_MS* is developed to ensure the instance's feasibility. It assumes the satisfaction of the time constraints by selecting the largest number of machines *NP(j)* within the largest size *NS(j)*, in all the stages *j*:

**18**
$$\begin{cases} p'(j) = NP(j) \\ s'(j) = NS(j) \end{cases}, \forall j$$

If assumption is positively checked then execution continues. Else the instance with multiple machines is unfeasible. In real cases, the cardinality of machines and/or discrete sizes must be enlarged if feasible solutions are desired.

**Step 2) Initial guess**

The procedure *APROX0_MS* is similar to the initial guess for model ***SS***, since it equally assumes that the time horizon $H$ must be satisfied by the sum of production times $t_i$ in all the stages $j$. Then, the minima for the machines number *p_h0* and volume sizes *V_h0*, in each stage $j$, is estimated using the expressions:

**19**
$$H = \sum_{i=1}^{N} \frac{S_{ij} Q_i \tau_{ij}}{p\_h0 * V\_h0} \Rightarrow p\_h0 * V\_h0 = \sum_{i=1}^{N} \frac{S_{ij} Q_i \tau_{ij}}{H}, \forall j$$

The continuous value of machines and volumes are adjusted to the immediately above discrete numbers of machines $p'(j)$ and sizes $s'(j)$, and the corresponding binary variables are assigned with the value 1 while the other binary variables are set to 0:

**20**
$$y\_h0(j, s'(j), p'(j)) = 1$$

This initial guess is then checked using the procedure *VERIFY_SOLUTION_MS*, to evaluate costs, variables, and restrictions satisfaction.

**Step 3) Building feasible solutions**

Two approaches are used to build feasible solutions for model MS. The procedure *APROX1A_MS* is based in the single machine approach, in case of feasibility and since its heuristic is already tuned. In case of the solution at hand is unfeasible, the procedure *APROX1A_MS* performs a local search to increase the number and/or the sizes of equipments, by balancing the reduction in the production times with the increase in costs.

**Step 3a) Procedure *APROX1A_MS***

The procedure *APROX1A_MS* is based in the feasibility of the single machine approach. A feasible solution built this way is an upper bound for the multiple machine approach, and further developments can be applied to improve

it, *e.g.*, by reducing discrete sizes as described later in *Step 4*. The procedure *APROX1A_MS* is considering some procedures already tuned in the single machine approach: firstly, verification of single machine feasibility; if positive, the related procedures for the initial guess *(APROX0_SS)* and feasible solution search *(APROX1_SS)* are applied.

**Step 3b) Procedure *APROX1B_MS* (Local Search)**

The procedure *APROX1B_MS* constructs a feasible solution from an unfeasible one originated in prior procedures *APROX0_MS* or *APROX1A_MS*. It searches to reduce the production times $t_i$ in order to satisfy the time horizon $H$, and evaluating the related increase in costs $c_{jsp}$. The approach is similar to the one developed for single machine, since it provided satisfactory results. Increasing the discrete size or the number of processing machines, the variation in costs is calculated by a direct subtraction of values.

The variation in production times considers the cycle time of each product $i$, which depends on the critical stage that may be defined accordingly the size or the number of machines. This is driving the *critical number* of batches or the *critical cycle time*, respectively, $j\_nb1_i$ or $j\_ct1_i$. When the critical stage suffers an alteration in size or number, a *second critical stage* is specified for each product, $j\_nb2_i$ or $j\_ct2_i$, in descending order of the production times. The total variation in production times is thus evaluated; the comparison with the corresponding variation in costs is performed by a ratio; the best ratio is selected, that is, the maximum value of the variations ratio. In fact, note the relations,

$$\textbf{21} \qquad t_i = \max_j \left\{ \frac{\tau_{ij}}{p'(j)} n_i \right\} = CT_i.n_i, \text{ where } \begin{cases} n_i = \max_j \left\{ \frac{S_{ij} Q_i}{dv_{j,s'(j)}} \right\} \\ CT_i = \max_j \left\{ \frac{\tau_{ij}}{p'(j)} \right\} \end{cases}, \forall i$$

When the size is increased in a critical stage $j\_nb1_i$, the values of the related auxiliary variables $nbcrit1_i$ are modified onto $nbcrit2_i$, since a second critical stage $j\_nb2_i$ is then defined. The variation in production times is expressed in terms of the numbers of batches considering that for each product $i$,

$$\textbf{22} \qquad \begin{aligned} t_i &= tcrit1_i - tcrit2_i \\ &= CT_i.nbcrit1_i - CT_i.nbcrit2_i = CT_i.nbcrit1_i - CT_i.nbcrit1_i \left( \frac{nbcrit2_i}{nbcrit1_i} \right) \\ &= CT_i.nbcrit1_i \left( 1 - \frac{nbcrit2_i}{nbcrit1_i} \right) = tcrit1_i \left( 1 - \frac{nbcrit2_i}{nbcrit1_i} \right), \ \forall i \end{aligned}$$

Similarly, if the machines number is increased in a critical stage $j\_ct1_i$, then the values of the related variables $CTcrit1_i$ are modified onto $CTcrit2_i$, since a second critical stage $j\_ct2_i$ is then defined. For each product $i$, the variation in production times is expressed in terms of the cycle times,

$$\textbf{23} \qquad \begin{aligned} \Delta t_i &= tcrit1_i - tcrit2_i \\ &= n_i.CTcrit1_i - n_i.CTcrit2_i = n_i.CTcrit1_i - n_i.CTcrit1_i \left( \frac{CTcrit2_i}{CTcrit1_i} \right) \\ &= n_i.CTcrit1_i \left( 1 - \frac{CTcrit2_i}{CTcrit1_i} \right) = tcrit1_i \left( 1 - \frac{CTcrit2_i}{CTcrit1_i} \right), \ \forall i \end{aligned}$$

The procedure *APROX1B_MS* performs an approach that is similar to the prior ILS procedure in the single machine approach, namely, in the neighborhood search, evaluation function, alteration selection, and termination step. *APROX1B_MS* initiates with an unfeasible solution and for each product $i$:

i) One larger size (or number of machines) is selected in a critical stage for product $i$; the variation in costs is calculated;
ii) The sum of the variations in production times is evaluated;
iii) The variations in times and in equipment costs are compared by its ratio, and considering the numbers of batches (or the cycle times);
iv) The best ratio is selected, and the corresponding discrete size is increasing, by swapping the related binary variables (assign 0 to the existing selection, and assign 1 to the next larger size);
v) To ensure its feasibility, the present binary solution is verified using the procedure *VERIFY_SOLUTION_MS*.
vi) If the solution is unfeasible and the iterations limit is not reached, then return to step i).

**Step 4) Tuning of feasible solutions**

The procedure *APROX2_MS* searches to improve a feasible solution that was generated in the prior procedures *APROX1A_MS* or *APROX1B_MS*. It aims to reduce equipment costs $c_{jsp}$ for the *non-critical stages*: stages where the processing time for a given product is shorter than its cycle time. Note that the maximum of the processing times for a given product occurs in its critical stage, where a bottleneck exists for the product. Then, bottlenecks never occur in the *non-critical stages*. The reduction in costs is obtained from a reduction in the discrete size or in the number of machines of a *non-critical stage*. The production times may (or may not) increase, but it is ensured that the satisfaction of the time horizon $H$ remains. The type of reduction (in size or in number of machines) is selected in concordance with the best ratio between the corresponding variations in costs and in production times. This approach is evolving in opposite sense of the prior procedures in *Step3*, but the reasoning is based in the same kind of relations: *i)* reducing the sizes $dv(j, s'(j))$, then the related numbers of batches ni are increasing, or vice versa; or *ii)* reducing the numbers of processing machines *p(j)* then the related cycle times $CT_i$ are increasing, or vice versa.

**Step 4a) Procedure *APROX2_MS* (Local Search)**

The procedure *APROX2_MS* searches to diminish the equipment costs, but it proceeds in opposite sense of the procedures that aim to construct a feasible solution. When the equipment size is reduced, the variation in times is related with the increasing (or not) numbers of batches,

$$\begin{aligned} t_i &= tcrit2_i - tcrit1_i \\ &= CT_i.nbcrit2_i - CT_i.nbcrit1_i = CT_i.nbcrit1_i\left(\frac{nbcrit2_i}{nbcrit1_i}\right) - CT_i.nbcrit1_i \\ &= CT_i.nbcrit1_i\left(\frac{nbcrit2_i}{nbcrit1_i} - 1\right) = tcrit1_i\left(\frac{nbcrit2_i}{nbcrit1_i} - 1\right),\ \forall i \end{aligned} \tag{24}$$

If the number of processing machine is reduced, similarly, the variation in production times is related with the increasing (or not) of cycle times:

$$\begin{aligned} \Delta t_i &= tcrit2_i - tcrit1_i \\ &= n_i.CTcrit2_i - n_i.CTcrit1_i = n_i.CTcrit1_i\left(\frac{CTcrit2_i}{CTcrit1_i}\right) - n_i.CTcrit1_i \\ &= n_i.CTcrit1_i\left(\frac{CTcrit2_i}{CTcrit1_i} - 1\right) = tcrit1_i\left(\frac{CTcrit2_i}{CTcrit1_i} - 1\right),\ \forall i \end{aligned} \tag{25}$$

The procedure *APROX2_MS* initiates with a feasible solution. For a given non-critical stage, the alteration by reducing the discrete size (or the number of machines) is analyzed, and the variation in costs is compared with the total variation of the production times; the ratio between the referred variations is evaluated and the best ratio is selected; and the binary variables corresponding to the alteration are adjusted. Finally, the feasibility of the solution is verified in the procedure *VERIFY_SOLUTION_MS*. This procedure *APROX2_MS* also develops a local search, where the termination criteria are both the exhausting of feasible solutions and the iterations limit. The described procedures are integrated in the heuristic for model ***MS***, from *Step1* to *Step4*. The values of the metrics related to the errors in the objective function and in the binary solutions are presented in Table 4:

Table 4. Execution times and deviations for heuristic of model ***MS***.

| Examples | Time *solver*(s) | Time heuristic (s) | Objective Function Deviations | Binary Solution Deviations |
|---|---|---|---|---|
| EX1A | 0.30 | 0.0027 | 84.// 0.// 7.6// 4.9 | 84.// 2.19 |
| EX2A | 10.53 | 0.0038 | 68.// 0. // 3.4// 3.8 | 68.// 2.03 |
| EX3A | 101.67 | 0.0042 | 42.// 0.// 7.0// 6.4 | 42.// 2.48 |
| EX4A | 405.99 | 0.0072 | 48.// 0.// 1.7// 2.1 | 48.// 1.88 |
| EX5A | (4098.31) | 0.0166 | 37.// 0.// 1.5// 1.5 | 37.// 2.58 |
| EX6A | (15142.36) | 0.0250 | 53.// 0.// 0.9// 0.9 | 53.// 2.00 |

The heuristic developed for model ***MS*** is satisfactory since:

i) Feasible solutions are always obtained;
ii) Optimal solutions are obtained in nearly 56% of the feasible instances;
iii) Sub-optimal solutions are of good quality, namely, errors are less than 2% in value for large instances, and almost only one discrete size switched.

Anyway, the model MS presented strong and weak points:

- Feasible solutions are always obtained;
- For comparable instances, the ***MS*** solutions are better or equal to the solutions in ***SS***;
- For large instances, with numerous continuous variables and constraints, the bound (3600 s) in execution time is often exceeded;
- Transition, setup, and cleanup times were not considered;
- The numbers of batches are specified, but their integrality is not required or imposed.

The model ***MS*** presented some solving difficulties, related with the treatment of the large number of continuous variables and restrictions. Nevertheless the satisfactory results from the heuristic, this model presents some weak points in the modeling of real cases, and the design is enlarged to MPC policy in next section.

## 5. Analysis of models *SM* and *MM*

The model ***SM*** searches how to choose the smallest (cheapest) size to satisfy products demand Q, on a given time horizon H, but considering the scheduling of several products in each production campaign (MPC).

In addition to the data for the SPC models, it is necessary to specify the slack times $SL_{ikj}$ that occur in the transition from product $i$ to product $k$, in the processing machines of stage $j$. The slack times are calculated by a recursive algorithm (Biegler *et al.*, 1997) that requires the cleanup times $CL_{ikj}$ of the corresponding transitions.

However, the solving difficulties are harder than in SPC, because the integration of the transition times within the ZW policy is requiring the entirely definition of the cyclic scheduling. In the opposite sense, notice that for UIS and zero values for cleanup times ($CL_{ikj} = 0$), the cycle times of each product are analytically calculated. However, when addressing UIS and non-zero cleanup times, the cyclic scheduling is required too.

The model ***SS***, addressing SPC production policy, may be treated as a specific instance of the model ***SM*** (Miranda, 2007), and equivalence relations occur between variables, objective function, and restrictions of these two models. For that, the number of transitions for each product $i$ is constrained to:

$$Nch_{ii} = n_i - 1, \ \forall i \qquad \textbf{26}$$

The theoretic equivalence is again used to validate the computational efforts: in model ***SM***, the number of transitions is satisfying *equation 26* for all the products; and cleanup and slack times are considered in the B series of random instances. The results are compared, and the equivalence between corresponding values was verified. Thus, the optimal value of model ***SS*** represents an upper bound of model ***SM***, since the latter considers MPC policy. In fact, the feasible space of MPC solutions is enlarging the set of solutions when SPC is addressed.

In Table 5, the parameters of the B series of examples for model ***SM*** are presented, so as the mean values of execution times. The instances in EX5B and EX6B are unfeasible (* mark), due to the large number of products and thus their demands are not achievable in the horizon time.

The execution times are moderate when compared with those of model ***MS*** (multiple machines and SPC). The solution in ***SM*** is quite efficient due to the aggregated TSP's formulation (Birewar & Grossmann, 1989*a*) and (Birewar & Grossmann, 1989*b*) of the transition times: this formulation allows the specification of the transition times by LP. Since the execution times are moderate, the development of a dedicated heuristic for model ***SM*** is weighted, and it is found not necessary.

Comparing the models ***SS*** and ***SM*** that are supposing SPC and MPC policies, respectively, the metrics (defined in Appendix) in Table 6 allow the analysis of the objective function and the binary solutions in comparable instances:

Table 5. Numbers of variables and restrictions in ***SM*** examples (B series).

| Examples | Parameters (N, M, NS, NP) | Binary variables | Continuous variables | Restrictions | Time (s) |
|---|---|---|---|---|---|
| EX1B | 3, 4, 5 | 20 | 12 | 29 | 0.46 |
| EX2B | 6, 4, 10 | 40 | 42 | 50 | 1.42 |
| EX3B | 8, 5, 12 | 60 | 72 | 74 | 2.38 |
| EX4B | 12, 6, 15 | 90 | 156 | 120 | 2.89 |
| EX5B | 20, 7, 17 | 119 | 420 | 214 | * |
| EX6B | 30, 9, 20 | 180 | 930 | 378 | * |

- Objective function - percentage of improved instances (negative deviations in costs, for the feasible instances) // percentage of unfeasible instances // percentage mean of the cost improvements // standard deviation on the percentage mean of cost improvements;
- Binary solutions - percentage of instances presenting the same binary solution // mean of the binary deviations for the comparable instances.

Table 6. Comparison between models ***SM*** and ***SS*** (MPC *vs*. SPC).

| Examples | Parameters (N, M, NS, NP) | Improvements in Objective Function | Binary Solution |
|---|---|---|---|
| EX1B | 3, 4, 5 | 43. // 6.7// 4.9// 2.7 | 57.// 1.6 |
| EX2B | 6, 4, 10 | 97. // 0. // 5.7// 3.6 | 3.// 2.5 |
| EX3B | 8, 5, 12 | 100. // 13.3// 5.2// 3.1 | 0.// 3.1 |
| EX4B | 12, 6, 15 | 100. // 66.7// 5.8// 0.8 | 0.// 4.4 |
| EX5B | 20, 7, 17 | * // 100. // *// * | * |
| EX6B | 30, 9, 20 | * // 100. // *// * | * |

When the optimal solutions of model ***SM*** and ***SS*** are compared, the MPC model presents cost improvements in most of instances, and the reduction in costs is a stable value of about 5%. The MPC policy is more efficient and minimizes transition times: production times increase, then shorter sizes are selected and costs are reduced.

When the number of products N increases, the necessity in production capacities is also increasing. Unfeasibility can occur if the machine with the largest size is not satisfying the demands within the time horizon. In Table 6, for examples EX5B an EX6B that are presenting the number of parameters in conformity with the real cases of chemical industry, the unfeasibility rules.

For model ***SM***, the development of heuristic or approximation procedures is found not essential because the reformulation using a TSP aggregated structure leads to LP's type solution and computational time (Birewar & Grossmann, 1989*a*) and (Birewar & Grossmann, 1989*b*). In plus: *i)* the cyclic scheduling in MPC has been computationally developed (Miranda, 2007); *ii)* the setup or cleanup times are addressed; and iii) in comparison with model SS, the results from SM showed about 5% reduction in investment cost.

However, some weaknesses are remaining in model SM:

- Unfeasibility occurs in a large number of instances, namely, when large values for N or Q are given;
- Integrality is not required for ni and Nchik.

In despite of the MPC policy, model ***SM*** showed that the single machine approach is quite limitative. The application onto real cases requires that multiple machines can be selected in each stage, as discussed in next section for model ***MM***.

### 5.2) Discussion of model MM

The model ***MM*** (multiple machine and MPC) searches the cheaper combination of machines and their discrete sizes to satisfy products demand Q, on the horizon, H, but considering the scheduling of several products in each production campaign (MPC) and multiple processes in parallel at each stage.

The model ***MM*** does not treat the slacks corresponding to the production transitions. Implicitly, it considers unlimited intermediate storage (UIS) instead of zero wait (ZW) policy. Then, there is a contradiction in the conditions assumed: MPC vs. zero production transitions. If the slack times are null, there is no necessity to optimize them using MPC.

This contradiction drives inconsistency on numerical results. That is, facing the same instances as ***SM*** (single machine and MPC), the model ***MM*** equally selects configurations of single machine but with smaller sizes and cheaper equipments. This occurs because the slack times are made zero.

To compare the MPC models, **SM** (single machine and MPC) and ***MM***, the number of processing machines is constrained to $NP = 1$ and the transition times are made $SL_{ikj} = 0$, in the sense to adjust the instances in both models. Running two series of 30 random instances, the results obtained are equal. Beyond the accuracy of the computational implementation, the theoretic assumption that ***SM*** is a specific instance of ***MM*** (Miranda, 2007) is verified. Also, the optimal value of model ***SM*** drives an upper bound for the ***MM*** objective function, and it can be used as a cutting value.

Also the model MS (multiple machine and SPC) may be treated as a specific instance of the model MM, if the numbers of transitions Nchikj are satisfying equation 26. Then, equivalence relations occur between variables, objective function, and restrictions of these two models (Miranda, 2007).

Table 7 shows the number of binary and continuous variables, the number of restrictions, and the mean execution times for the ***MM*** numerical examples (B series).

Table 7. Numbers of variables and restrictions in the numerical examples of model ***MM*** (B series).

| Examples | Binary (N, M, NS, NP) | Continuous Variables | Restrictions Variables | Restrictions | Time solver (s) |
|---|---|---|---|---|---|
| EX1B | 3, 4, 5, 3 | 60 | 12 | 29 | 1.65 |
| EX2B | 6, 4, 10, 3 | 120 | 42 | 50 | 3.49 |

Table 7 presents only two examples, since results inconsistency is observed. The results from model ***MM*** were compared with the results from model ***SM***, and:

- The unfeasible instances in single machine are becoming feasible in the multiple machines, as expected;
- For the feasible instances in SM, the results from MM are better, and it was expected that results never be worse;
- However, when the solutions that require only one machine at each stage are compared, lesser costs are observed, since shorter sizes were selected for this single machine.

That is, facing the same instance, with equal values on parameters and data, when ***MM*** selects only one processing machine at each stage, it selects shorter sizes that are driving lesser costs. This incoherence can be explained since the available time for production is larger due to the nullification of transition times. In fact, ***MM*** will not treat the slack times in production transitions when it already assumed that these slacks are zero. Implicitly, model ***MM*** is considering unlimited intermediate storage (UIS) in contradiction with the ZW policy, as it was stated.

This contradiction is driven from a theoretic limitation, since the products sequencing in model SM (single machine and MPC) used the TSP aggregated structure that allows LP solving, as described by (Pekny & Miller, 1991). In ***MM***, it would be necessary to combine a TSP problem for each of the machines implemented in parallel, which seems not reasonable.

The model ***MM*** is the more enlarged formulation in analysis, since it generalizes both the machines number and products number in the production campaigns (MPC). The limited numerical runs are revealing concordance with the described theoretic considerations:

- In comparison with model SM, the costs are diminishing in MM;
- For all instances, feasibility rules;
- Integrality is not required for $n_i$ and $Nch_{ik}$;
- The setup or cleanup times are formulated, but not numerically treated;
- The implicit UIS costs and parameters are not considered.

Initially, ***MM*** was the model that presented better expectations, but in despite of the reduction in investment costs, the application of the corresponding solutions is not realistic. The inconsistence in the numerical results is due to the implicit contradiction in the operation mode, UIS vs. ZW, and then its application to real cases is not possible.

## 6. Conclusions

Several models aimed to simultaneously design and scheduling batch chemical processes are selected from literature, compared, and various heuristics are developed in order to obtain satisfactory solutions.

The selected models simultaneously address the scheduling of the production cycles embedded on the design problem of batch chemical processes, considering multiproduct environment (flowshop) and ZW storage policy. The models differ in the number of processes considered at each stage (single machine vs. multiple machine) and in the production policy (SPC *vs*. MPC).

In a combinatory way, four different models are studied (***SS, MS, SM,*** and ***MM***), their characteristics and limitations are referred, and the probabilistic analysis of the heuristics is performed. The main points of the analysis are:

i) For model ***SS***, several approximation procedures are built and integrated in a heuristic; a significant fraction of optimal solutions is obtained, with errors of about 1% in value;
ii) For ***MS*** (multiple machines and SPC), the solving difficulties are increasing; a heuristic is developed too, and a similar level of quality is obtained;
iii) For ***SM*** (single machine and MPC), the solver execution was efficient but it is assumed the integrality relaxation in the number of batches; when compared with the model ***SS***, the model ***SM*** predicts reduction of about 5% in the investment costs due to the MPC policy;
iv) The model ***MM*** allowed feasible solutions in realistic instances with high number of products or quantities that are not achievable through the previous model ***SM*** (single machine and MPC); however, theoretical contradictions are driving numerical incoherence; namely, using comparable data and constraining the number of machines to 1, the results from ***MM*** are significantly better than those from ***SM***, but they are not realistic.

From the comparative analysis performed, the combination of multiple machines with SPC was found to be the most promising from a computational standpoint. On a generalization approach, the model ***MS*** is thus selected toward a stochastic environment.

### Acknowledgement

This work was supported in part by the Centre for Chemical Processes (CPQ) at the IST/UTL. The Author also thanks the support at ESTG/IPP.

### Nomenclature

***Index and sets***

| | |
|---|---|
| *M* | - number of stages *j*; |
| *NC* | - number of components or products *i*; |
| *NP(j)* | - number of processes *p(j)* per stage; |
| *NS(j)* | - number of discrete dimensions *s(j)* in the process of stage *j*; |

***Parameters***

$\tau_{ij}$ - processing times (h), for each product *i* in stage *j*;
$c_{jsp}$ - equipment cost of process *p(j)* and size *s(j)* in stage *j*;
$dv_j$ - discrete volume (equipment size) in each stage *j*;
$H$ - time horizon;
$nc_{ijsp}^{Upp}$ - upper bound on the number of batches of product *i*, disaggregated by process *p(j)* and size *s(j)* in each stage *j*;
*p(j)* - (ordinal) number of processes in stage j;
$Q_i$ - demand quantities (uncertain) for each product *i*;
*s(j)* - (ordinal) number of process discrete dimensions in stage *j*;
$S_{ij}$ - dimension factor (L/kg), for each product *i* in stage *j*;
$V_j$ - equipment volume (continuous value) in each stage *j*;

***Variables***

$n_i$ - number of batches of product *i*;
$nc_{ispj}$ - number of batches of product *i*, disaggregated by process *p(j)* and size *s(j)* in each stage *j*;
$Nch_{ik}$ - number of transitions (changes) from product *i* to *k*;
$t_i$ - production times of each product *i*;
$y_{jsp}$ - binary decision toward process *p(j)* and size *s(j)* in stage *j*;

## Appendix - Estimators for analysis of the deviations

In the probabilistic analysis of heuristics, several estimators are evaluated and presented in Table 2 and Table 4. Namely, excluding unfeasible instances and feasible instances that are exactly estimated, the percentage errors between the corresponding values of the optimal objective function ("OPT" indexes) and the sub-optimal heuristic ("H" indexes) are evaluated using the expression:

**A.1** $$\%dsv\_z = \frac{|z_{OPT} - z_H|}{z_{OPT}}.100.$$

Again excluding unfeasible instances and instances presenting exact binary solution, the numbers of binary errors (Ndsv_y) for the heuristics sub-optimal solutions are evaluated by:

**A.2** $$Ndsv\_y = \sum_{j=1}^{M} \sum_{s=1}^{NS} \frac{|y_{OPT}(j,s) - y_H(j,s)|}{2}.$$

In Table 6, considering only the feasible and comparable instances, the percentage improvements in the corresponding values of the objective function in MPC ("M" indexes) and SPC ("S" indexes) are given by:

**A.3** $$\%dsv\_z = \frac{|z_M - z_S|}{z_M}.100.$$

Again excluding the instances presenting the same binary solution, the numbers of binary deviations for the improved MPC instances, in relation to the SPC ones, are obtained from:

**A.4** $$Ndsv\_y = \sum_{j=1}^{M} \sum_{s=1}^{NS} \frac{|y_M(j,s) - y_S(j,s)|}{2}.$$

## References

Ahmed, S. and N. V. Sahinidis (2000). Analytical investigations of the process planning problem. *Computers and Chemical Engineering* **23**, 1605–1625.

Ahmed, S. and N. V. Sahinidis (2003). An approximation scheme for stochastic integer programs arising in capacity expansion. *Operations Research* **51**(3), 461–471.

Barbosa-Póvoa, A. P. (2007). A critical review on the design and retrofit of batch plants. *Computers and Chemical Engineering* (31), 833–855.

Biegler, L. T., I. E. Grossmann and A. W. Westerberg (1997). *Systematic Methods of Chemical Process Design*. Prentice-Hall, New Jersey.

Birewar, D. B. and I. E. Grossmann (1989*a*). Efficient optimization algorithms for zero-wait scheduling of multiproduct batch plants. *Ind. Eng. Chem. Res.* **28**, 1333–1345.

Birewar, D. B. and I. E. Grossmann (1989*b*). Incorporating scheduling in the optimal design of multiproduct batch plants. *Computers and Chemical Engineering* **13**, 141–161.

Cavin, L., U. Fischer, F. Glover and K. Hungerbhler (2004). Multi-objective process design in multi-purpose batch plants using a tabu search optimization algorithm. *Computers and Chemical Engineering* (28), 459–478.

Garey, M. R., D. S. Johnson and R. Sethi (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research* **1**(2), 117–129.

Jayaraman, V. K., B. D. Kulkarni, S. Karale and P. Shelokar (2000). Ant colony framework for optimal design and scheduling of batch plants. *Computers and Chemical Engineering* **24**, 1901–1912.

Miranda, J. L. (2007). Optimização em Sistemas de Processos Químicos: Generalização de Modelos com Planeamento e Sequenciamento. PhD dissertation. Instituto Superior Técnico, Technical University of Lisbon, Lisboa.

Pekny, J. F. (2002). Algorithm architectures to support large-scale process systems engineering applications involving combinatorics, uncertainty, and risk management. *Computers and Chemical Engineering* **26**, 239–267.

Pekny, J. F. and D. L. Miller (1991). Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristic methods. *Computers and Chemical Engineering* **15**(11), 741–748.

Tan, S. and R. S. H. Mah (1998). Evolutionary design of noncontinuous plants. *Computers and Chemical Engineering* **22**(1/2), 69–85.

Voudouris, V. T. and I. E. Grossmann (1992). Industrial and engineering chemistry research. *Computers and Chemical Engineering* **31**, 1315–1325.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 63–67

# Inequalities Involving Noor Integral Operator

Mohamed Kamal Aouf[a], Daniel Breaz[b,*], Nicoleta Breaz[b]

[a]*Department of Mathematics, Faculty of Science, Mansoura University, Mansoura 35516, Egypt.*
[b]*Department of Mathematics, "1 Decembrie 1918" University of Alba Iulia, Alba, 510009, Str. N. Iorga, No. 11-13, Romania.*

**Abstract**

The object of the present paper is to give an application of Noor integral operator $I_{n+p-1}(n > -p; p \in N)$ to Miller and Mocanu's theorems.

*Keywords:* Analytic, $p$-valent, Noor integral operator.
*2000 MSC:* 30C45.

## 1. Introduction

Let $A(p)$ be the class of functions of the form :

$$f(z) = z^p + \sum_{k=p+1}^{\infty} a_k z^k \quad (p \in N = \{1, 2, ....\}), \tag{1.1}$$

which are analytic and $p$-valent in the unit disc $U = \{z : |z| < 1\}$. For functions $f_j(z)(j = 1, 2)$ defined by

$$f_j(z) = z^p + \sum_{k=p+1}^{\infty} a_{k,j} z^k, \tag{1.2}$$

we define the Hadamard product ( or convolution ) $(f_1 * f_2(z))$ of functions $f_1(z)$ and $f_2(z)$ by

$$(f_1 * f_2(z)) = z^p + \sum_{k=p+1}^{\infty} a_{k,1} a_{k,2} z^k \ . \tag{1.3}$$

The integral operator $I_{n+p-1} : A(p) \to A(p)$ is defined as follows (see (Liu & Noor, 2000)):

For any integer $n$ biger than $-p$, let $f_{n+p-1}(z) = \frac{z^p}{(1-z)^{n+p}}$ and let $f_{n+p-1}(z)$ be defined such that

$$f_{n+p-1}(z) * f^{(-1)}_{n+p-1}(z) = \frac{z^p}{(1-z)^{1+p}} \ . \tag{1.4}$$

Then

$$I_{n+p-1} f(z) = f^{(-1)}_{n+p-1} * f(z) = \left[\frac{z^p}{(1-z)^{n+p}}\right]^{(-1)} * f(z). \tag{1.5}$$

*Corresponding author
*Email addresses:* mkaouf127@yahoo.com (Mohamed Kamal Aouf), dbreaz@uab.ro (Daniel Breaz), nbreaz@uab.ro (Nicoleta Breaz)

From (1.4) and (1.5) and a well-known identity for the Ruscheweyh derivative (see (Goel & Sohi, 1980) and (Ruscheweyh, 1975)), it follows that

$$z(I_{n+p}f(z))' = (n+p)I_{n+p-1}f(z) - nI_{n+p}f(z). \tag{1.6}$$

For $p = 1$, the identity (1.6) is given by Noor and Noor (Noor & Noor, 2005). If $f(z)$ is given by (1.1), then from (1.4) and (1.5), we deduce that

$$I_{n+p-1}f(z) = \left[z^p {}_2F_1(1, 1+p; n+p; z)\right] * f(z)(n > -p), \tag{1.7}$$

where ${}_2F_1$ is the hypergeometric function. We also note that $I_{p-1}f(z) = \frac{zf'(z)}{p}$ and $I_p f(z) = f(z)$. Moreover, the operator $I_{n+p-1}f(z)$ defined by (1.5) is called as Noor integral operator of $(n+p-1)$-th order of $f(z)$ (see (Liu & Noor, 2000)). For $p = 1$, the operator $I_n f$ was introduced by Noor (Noor, 1999) and Noor and Noor (Noor & Noor, 2005). Several classes of analytic functions, defined by using the operator $I_{n+p-1}f(z)$, have been studied by many authors (see (Noor, 2004), (Noor, 2005) and (Patel & Cho, 2005)).

By using the operator $I_{n+p-1}$ we define :

**Definition 1.1.** Let $G_1$ be the set of complex- valued functions $g(r, s, t)$;

$$g(r, s, t) : C^3 \to C \text{ (C is the complex plane)}$$

such that

(i) $g(r, s, t)$ is continuous in a domain $D \subset C^3$;
(ii) $(0, 0, 0) \in D$ and $|g(0, 0, 0)| < 1$;
(iii) $\left|g(e^{i\theta}, \frac{n+\zeta}{n+p}e^{i\theta}, \frac{(n-1)(n+2\zeta)e^{i\theta}+M}{(n+p)(n+p-1)})\right| > 1$
whenever $(e^{i\theta}, \frac{n+\zeta}{n+p}e^{i\theta}, \frac{(n-1)(n+2\zeta)e^{i\theta}+M}{(n+p)(n+p-1)}) \in D$ with $Re\{e^{-i\theta}M\} \geq \zeta(\zeta-1)$ for all $\theta \in R$, and for all $\zeta \geq p \geq 1$.

**Definition 1.2.** Let $G_2$ be the set of complex- valued functions $h(r, s, t)$;

$$h(r, s, t) : C^3 \to C$$

such that

(i) $h(r, s, t)$ is continuous in a domain $D \subset C^3$;
(ii) $(1, 1, 1) \in D$ and $|h(1, 1, 1)| < J$ $(J > 1)$;
(iii) $\left|h(Je^{i\theta}, \frac{\zeta-1+(n+p)Je^{i\theta}}{n+p-1}, \frac{1}{n+p-2}\{\zeta - 1 + (n+p)Je^{i\theta} + \frac{\zeta-\zeta^2+(n+p)\zeta Je^{i\theta}+L}{\zeta-1+(n+p)Je^{i\theta}}\})\right| \geq J$,
whenever $(Je^{i\theta}, \frac{\zeta-1+(n+p)Je^{i\theta}}{n+p-1}, \frac{1}{n+p-2}\{\zeta - 1 + (n+p)Je^{i\theta} + \frac{\zeta-\zeta^2+(n+p)\zeta Je^{i\theta}+L}{\zeta-1+(n+p)Je^{i\theta}}\}) \in D$
with $Re\{L\} \geq \zeta(\zeta-1)$ for all $\theta \in R$ and for all $\zeta \geq \frac{J-1}{J+1}$.

## 2. Main Results

We begin with the statement of the following lemmas due to Miller and Mocanu (Miller & Mocanu, 1978).

**Lemma 2.1.** ***(Miller & Mocanu, 1978).*** *Let* $w(z) = b_p z^p + b_{p+1}z^{p+1} + .....(p \in N)$ *be regular in the unit disc* $U$ *with* $w(z) \neq 0$ $(z \in U)$. *If* $z_0 = r_0 e^{i\theta}(0 < r_0 < 1)$ *and* $|w(z_0)| = \max_{|z|\leq|z_0|} |w(z)|$ *then*

$$(i) \quad z_0 w'(z_0) = \zeta w(z_0) \tag{2.1}$$

and

$$(ii) \quad Re\left\{1 + \frac{z_0 w''(z_0)}{w'(z_0)}\right\} \geq \zeta \tag{2.2}$$

where $\zeta$ is real and $\zeta \geq p \geq 1$.

**Lemma 2.2.** *(Miller & Mocanu, 1978). Let $w(z) = a + w_k z^k + .....$be regular in $U$ with $w(z) \neq a$ and $k \geq 1$. If $z_0 = r_0 e^{i\theta} (0 < r_0 < 1)$ and $|w(z_0)| = \max_{|z| \leq |z_0|} |w(z)|$then*

$$(i) \quad z_0 w'(z_0) = \zeta w(z_0)$$

$$(ii) \quad Re\left\{1 + \frac{z_0 w''(z_0)}{w'(z_0)}\right\} \geq \zeta$$

*where $\zeta$ is a real number and*

$$\zeta \geq k \frac{|w(z_0) - a|^2}{|w(z_0)|^2 - |a|^2} \geq k \frac{|w(z_0)| - |a|}{|w(z_0)| + |a|}. \tag{2.3}$$

*Making use of the above lemmas, we prove*

**Theorem 2.3.** *Let $g(r, s, t)$ be in $G_1$, and let $f(z)$ belonging to the class $A(p)$, satisfy*

(i) $(I_{n+p} f(z), I_{n+p-1} f(z), I_{n+p-2} f(z)) \in D \subset C^3$
and
(ii) $\left|g(I_{n+p} f(z), I_{n+p-1} f(z), I_{n+p-2} f(z))\right| < 1$
where $n > -p, p \in N$ and $z \in U$ . Then we have

$$\left|I_{n+p} f(z)\right| < 1 \quad (z \in U). \tag{2.4}$$

We define the function $w(z)$ by

$$I_{n+p} f(z) = w(z) \qquad (n > -p; p \in N) \tag{2.5}$$

for $f(z)$ belonging to the class $A(p)$. Then, it follows that $w(z) \in A(p)$ and $w(z) \neq 0 \, (z \in U)$. With the aid of the identily (1.6), we have

$$I_{n+p-1} f(z) = \frac{1}{n+p} \left\{n w(z) + z w'(z)\right\} \tag{2.6}$$

and

$$I_{n+p-2} f(z) = \frac{1}{(n+p)(n+p-1)} \left\{n(n-1) w(z) + 2(n-1) z w'(z) + z^2 w''(z)\right\}. \tag{2.7}$$

Suppose that $z_0 = r_0 e^{i\theta} \quad (0 < r_0 < 1; \theta \in R)$ and

$$|w(z_0)| = \max_{|z| \leq |z_0|} |w(z)| = 1 \tag{2.8}$$

Letting $w(z_0) = e^{i\theta}$ and using (2.1) of Lemma 2.1, we see that

$$L_{n+p} f(z_0) = w(z_0) = e^{i\theta}, \tag{2.9}$$

$$L_{n+p-1} f(z_0) = \frac{n+\zeta}{n+p} w(z_0) = \frac{n+\zeta}{n+p} e^{i\theta}, \tag{2.10}$$

and

$$\begin{aligned} L_{n+p-2} f(z_0) &= \frac{1}{(n+p)(n+p-1)} \left\{(n-1)(n+2\zeta) w(z_0) + z_0^2 w''(z_0)\right\} \\ &= \frac{(n-1)(n+2\zeta) e^{i\theta} + M}{(n+p)(n+p-1)}, \end{aligned} \tag{2.11}$$

where $M = z_0^2 w''(z_0)$ and $\zeta \geq p \geq 1$.

Further, an application of (2.2) in Lemma 2.1 gives

$$Re\left\{\frac{z_0 w''(z_0)}{w'(z_0)}\right\} = Re\left\{\frac{z_0^2 w''(z_0)}{\zeta e^{i\theta}}\right\} \geq (\zeta - 1), \tag{2.12}$$

or

$$Re\left\{e^{-i\theta}M\right\} \geq \zeta(\zeta-1)(\theta \in R; \zeta \geq 1). \tag{2.13}$$

Since $g(r,s,t) \in G_1$, we have

$$\left|g(I_{n+p}f(z_0), I_{n+p-1}f(z_0), I_{n+p-2}f(z_0))\right|$$

$$= \left|g(e^{i\theta}, \frac{n+\zeta}{n+p}e^{i\theta}, \frac{(n-1)(n+2\zeta)e^{i\theta}+M}{(n+p)(n+p-1)})\right| > 1 \tag{2.14}$$

which contradicts the condition (ii) of Theorem 2.3. Therefore, we conclude that

$$|w(z)| = \left|I_{n+p}f(z)\right| < 1, \tag{2.15}$$

which $n > -p; p \in N$ and for all $z \in U$. This completes the proof of Theorem 2.3.

**Corollary 2.4.** *Let $g_0(r,s,t) = s$ and let $f(z)$ belonging to the class $A(p)$ satisfy the conditions in Theorem 2.3. Then*

$$\left|I_{n+p+i}f(z)\right| < 1 \quad (i = 0,1,2,....; n > -p; p \in N; z \in U). \tag{2.16}$$

*Proof.* Note that $g_0(r,s,t) = s$ is in $G_1$, with the aid of Theorem 2.3, we have

$$\left|I_{n+p-1}f(z)\right| < 1 \Longrightarrow \left|I_{n+p}f(z)\right| < 1 \quad (n > -p; p \in N)$$

$$\Longrightarrow \left|I_{n+p+i}f(z)\right| < 1 \quad (i = 0,1,2,....; n > -p; p \in N; z \in U).$$

□

**Theorem 2.5.** *.Let $h(r,s,t) \in G_2$ , let $f(z)$ belonging to $A(p)$ satisfying*

$$(i)\ (\frac{I_{n+p-1}f(z)}{I_{n+p}f(z)}, \frac{I_{n+p-2}f(z)}{I_{n+p-1}f(z)}, \frac{I_{n+p-3}f(z)}{I_{n+p-2}f(z)}) \in D \subset C^3$$

*and*

$$(ii)\ \left|h(\frac{I_{n+p-1}f(z)}{I_{n+p}f(z)}, \frac{I_{n+p-2}f(z)}{I_{n+p-1}f(z)}, \frac{I_{n+p-3}f(z)}{I_{n+p-2}f(z)})\right| < J$$

*for some $n, J$ $(n > -p; p \in N; J > 1)$ and for all $z \in U$. Then we have*

$$\left|\frac{I_{n+p-1}f(z)}{I_{n+p}f(z)}\right| < J \quad (z \in U). \tag{2.17}$$

*Proof.* We define the function $w(z)$ by

$$\frac{I_{n+p-1}f(z)}{I_{n+p}f(z)} = w(z) \qquad (n > -p; p \in N; z \in U) \tag{2.18}$$

for $f(z)$ belonging to the class $A(p)$. Then, it follows that $w(z)$ is either analytic or meromorphic in $U$, $w(0) = 1$, and $w(z) \neq 1$. With the aid of the identity (1.6), we have

$$\frac{I_{n+p-2}f(z)}{I_{n+p-1}f(z)} = \frac{1}{n+p-1}[(n+p)w(z) - 1 + \frac{zw'(z)}{w(z)}] \tag{2.19}$$

and

$$\frac{I_{n+p-3}f(z)}{I_{n+p-2}f(z)} = \frac{1}{n+p-2}\left\{(n+p)w(z) - 1 + \frac{zw'(z)}{w(z)} + \right.$$

$$\left. \frac{(n+p)zw'(z) + \frac{zw'(z)}{w(z)} + \frac{z^2w''(z)}{w(z)} - (\frac{zw'(z)}{w(z)})^2}{(n+p)w(z) - 1 + \frac{zw'(z)}{w(z)}}\right\}. \tag{2.20}$$

Suppose that $z_0 = r_0 e^{i\theta}(0 < r_0 < 1; \theta \in R)$ and $|w(z_0)| = \max_{|z|\le|z_0|} |w(z)| = J$. Letting $w(z_0) = Je^{i\theta}$ and using Lemma 2.2 with $a = k = 1$, we see that

$$\frac{I_{n+p-2}f(z_0)}{I_{n+p-1}f(z_0)} = \frac{1}{n+p-1}[\zeta - 1 + (n+p)Je^{i\theta}] \tag{2.21}$$

and

$$\frac{I_{n+p-3}f(z_0)}{I_{n+p-2}f(z_0)} = \frac{1}{n+p-2}\Big[\zeta - 1 + (n+p)Je^{i\theta} + \frac{\zeta - \zeta^2 + (n+p)\zeta Je^{i\theta} + L}{\zeta - 1 + (n+p)Je^{i\theta}}\Big], \tag{2.22}$$

where $L = \frac{z_0^2 w''(z_0)}{w(z_0)}$ and $\zeta \ge \frac{J-1}{J+1}$.
Further, an application of (ii) in Lemma 2.2 gives

$$Re\{L\} \ge \zeta(\zeta - 1).$$

Since $h(r, s, t) \in G_2$, we also have

$$\begin{aligned} &\Big|h\Big(\frac{I_{n+p-1}f(z_0)}{I_{n+p}f(z_0)}, \frac{I_{n+p-2}f(z_0)}{I_{n+p-1}f(z_0)}, \frac{I_{n+p-3}f(z_0)}{I_{n+p-2}f(z_0)}\Big)\Big| \\ &= \Big|h\Big(Je^{i\theta}, \frac{\zeta - 1 + (n+p)Je^{i\theta}}{n+p-1}, \frac{1}{n+p-2}\Big\{\zeta - 1 + (n+p)Je^{i\theta} + \frac{\zeta - \zeta^2 + (n+p)\zeta Je^{i\theta} + L}{\zeta - 1 + (n+p)Je^{i\theta}}\Big\}\Big)\Big| \ge J, \end{aligned} \tag{2.23}$$

which contradicts condition (ii) of Theorem 2.3. Therefore, we conclude that

$$|w(z)| = \Big|\frac{I_{n+p-1}f(z)}{I_{n+p}f(z)}\Big| < J \tag{2.24}$$

for $n > -p, p \in N$ and $z \in U$. This completes the proof of Theorem 2.3. □

## References


Goel, R.M. and N. S. Sohi (1980). A new criterion for p-valent functionsn. *Proc. Amer. Math. Soc.* (78), 353–357.

Liu, J.-L. and K. I. Noor (2000). Some properties of noor integral operator. *J. Nat. Geom.* (21), 81–90.

Miller, S. S. and P. T. Mocanu (1978). Second differential inequalities in the complex plane. *J. Math. Anal. Appl.* **65**, 289–305.

Noor, K. I. (1999). On new classes of integral operators. *J. Nat. Geom.* **16**, 71–80.

Noor, K. I. (2004). Some classes of p-valent analytic functions defined by certain integral integral operator. *Applied Math. Comput.* **157**, 835–840.

Noor, K. I. (2005). Generalized integral operator and multivalent functions. *J. Intequal. Pure Appl. Math.* **6**(2), 1–7.

Noor, K. I. and M. A. Noor (2005). On integral operators. *J. Math. Anal. Appl.* **238**, 341–352.

Patel, J. and N.E. Cho (2005). Some classes of analytic functions involving noor integral operator. *Journal of Mathematical Analysis and Applications* **312**(2), 564 – 575.

Ruscheweyh, St. (1975). New criteria for univalent functions. *Proc. Amer. Math. Soc.* **49**(1), 109–115.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 68–73

# A Short Note

# Certain New Classes of Analytic and Univalent Functions in the Unit Disk

Abiodun T. Oladipo[a,*]

[a]*Ladoke Akintola University of Technology, Ogbomoso, Department of Pure and Applied Mathematics, P. M. B. 4000, Ogbomoso, Nigeria.*

**Abstract**

We introduce the classes $H(\omega,\alpha)$ and $K(\omega,\alpha)$ of analytic functions with negative coefficients. In this work we give some properties of functions in these classes and we obtain coefficient estimates, neighborhood and integral means inequalities for function $f(z)$ belonging to these classes.

*Keywords:* Starlike, Convex, Coefficient Estimate, Neighbourhood, Integral Means Inequalities.
*2000 MSC:* Primary 30C45.

## 1. Introduction

Let $A$ be the class of function $f(z)$ of the form

$$f(z) = z + \sum_{k=2}^{\infty} a_k z^k \tag{1.1}$$

which are analytic in the unit disk $E = \{z : |z| < 1\}$. Let $S$ denote the subclass of $A$ consisting of univalent functions $f(z)$ in $E$.

It is necessary here to recall the definitions of the well-known classes of starlike and convex functions

$$S^* = \left\{ f \in A : Re\left(\frac{zf'(z)}{f(z)}\right) > 0, z \in E \right\}$$
$$S^c = \left\{ f \in A : Re\left(1 + \frac{zf''(z)}{f'(z)}\right) > 0, z \in E \right\}.$$

Let $A(\omega) \subset A$ denote the class of functions of the form

$$f(z) = (z-\omega) + \sum_{k=2}^{\infty} a_k (z-\omega)^k \tag{1.2}$$

which are analytic in the unit disk $E = \{z : |z| < 1\}$ and normalized with $f(\omega) = 0$ and $f'(\omega) - 1 = 0$, and $\omega$ is a fixed point in $E$.

*Corresponding author
*Email address:* atlab_3@yahoo.com (Abiodun T. Oladipo)

In (Kanas & Ronning, 1999), S. Kanas and F. Ronning introduced and studied the following classes of functions.

$$S(\omega) = \{f \in A(\omega) : f \text{ is univalent in } E\}$$
$$ST(\omega) = S^*(\omega) = \left\{f \in S(\omega) : Re\left(\frac{(z-\omega)f'(z)}{f(z)}\right) > 0, z \in E\right\}$$
$$CV(\omega) = S^c = \left\{f \in S(\omega) : Re\left(1 + \frac{(z-\omega)f''(z)}{f'(z)}\right) > 0, z \in E\right\}$$

which are respectively the classes of univalent, starlike and convex functions and $\omega$ is a fixed point in $E$. These classes were further studied in (Acu & Owa, 2005).

Let $T(\omega)$ denote subclass of $S(\omega)$ whose elements can be expressed in the form

$$f(z) = (z-\omega) - \sum_{k=2}^{\infty} a_k(z-\omega)^k. \tag{1.3}$$

Here we denote by $H(\omega,\alpha)$ and $K(\omega,\alpha)$ repectively the subfamilies of $S^*(\omega,\alpha)$ and $S^c(\omega,\alpha)$ obtained by taking intersection of $S^*(\omega,\alpha)$ and $S^c(\omega,\alpha)$ with $T(\omega)$ that is,

$$H(\omega,\alpha) = S^*(\omega,\alpha) \cap T(\omega)$$

and

$$K(\omega,\alpha) = S^c(\omega,\alpha) \cap T(\omega)$$

where $S^*(\omega,\alpha)$ and $S^c(\omega,\alpha)$ are respectively classes of starlike of order $\alpha$ and convex of order $\alpha$ (Oladipo, 2009).

Consequently, we have

$$H(\omega,0) = S^*(\omega,0) \cap T(\omega) \Rightarrow H(\omega) = S^*(\omega) \cap T(\omega)$$

and

$$K(\omega,0) = S^c(\omega,0) \cap T(\omega) \Rightarrow K(\omega) = S^c(\omega) \cap T(\omega).$$

Also let $P(\omega) \subset P$(class of Caratheodory functions) denote the class of functions of the form

$$p_\omega(z) = 1 + \sum_{k=2}^{\infty} B_k(z-\omega)^k \tag{1.4}$$

that are regular in $E$ and satisfy $p_\omega(\omega) = 1$, $Rep_\omega(z) > 0$ for $z \in E$ and $\omega$ is a fixed point in $E$ and

$$|B_k| \leq \frac{2}{(1+d)(1-d)^k}, \ k \geq 1, \ \text{and } d = |\omega|$$

(Kanas & Ronning, 1999), (Acu & Owa, 2005), (Wald, 1978).

## 2. Coefficient estimates

For our main results we first derive the following:

**Lemma 2.1.** *A function $f(z) \in T(\omega)$ is in the class $H(\omega,\alpha)$ if and only if*

$$\sum_{k=2}^{\infty}(k-\alpha)(1-d)^{k-1}a_k \leq 1-\alpha. \tag{2.1}$$

The result is sharp.

*Proof.* Assume that the inequality (2.1) holds and let $|z-\omega| = 1-d < 1$. Then we have

$$\left|\frac{(z-\omega)f'(z)}{f(z)} - 1\right| = \left|\frac{-\sum_{k=2}^{\infty}(k-1)a_k(z-\omega)^{k-1}}{1-\sum_{k=2}^{\infty}a_k(z-\omega)^{k-1}}\right|$$

$$\leq \frac{\sum_{k=2}^{\infty}(k-1)(1-d)^{k-1}a_k}{1-\sum_{k=2}^{\infty}(1-d)^{k-1}a_k} \leq 1-\alpha.$$

This shows that the values of $\frac{(z-\omega)f'(z)}{f(z)}$ lie in the circle centred at $\gamma = 1$ whose radius is $1-\alpha$. Hence $f(z)$ is in the class $H(\omega,\alpha)$. Then

$$Re\frac{(z-\omega)f'(z)}{f(z)} = Re\left\{\frac{1-\sum_{k=2}^{\infty}ka_k(z-\omega)^{k-1}}{1-\sum_{k=2}^{\infty}a_k(z-\omega)^{k-1}}\right\} > \alpha \tag{2.2}$$

for $z \in E$ and $\omega$ is a fixed point in $E$.
Choose values of $z$ on the real axis so that $\frac{(z-\omega)f'(z)}{f(z)}$ is real. Upon clearing the denominator in (6) and letting $z \to 1^-$ through real values, we have

$$\alpha\left(1-\sum_{k=2}^{\infty}(1-d)^{k-1}a_k\right) \leq 1-\sum_{k=2}^{\infty}k(1-d)^{k-1}a_k \tag{2.3}$$

which obviousely is the required result.
Finally, we note that the assertion (2.1) of Lemma 2.1 is sharp, with the extremal function being

$$f(z) = (z-\omega) - \frac{1-\alpha}{(k-\beta)(1-d)^{k-1}}(z-\omega)^k. \tag{2.4}$$

□

**Corollary 2.2.** *Let $f(z) \in T(\omega)$ be in the class $H(\omega,\alpha)$. Then we have*

$$a_k \leq \frac{1-\alpha}{(k-\beta)(1-d)^{k-1}} \tag{2.5}$$

Equality in (2.5) holds true for the function $f(z)$ given by (2.4).

**Lemma 2.3.** *A function $f(z) \in T(\omega)$ is in the class $K(\omega,\alpha)$ if and only if*

$$\sum_{k=2}^{\infty}k(k-\beta)(1-d)^{k-1}a_k \leq 1-\alpha. \tag{2.6}$$

The result is sharp.

*Proof.* The proof follows the same method as in Lemma 2.1 . The assertion of Lemma 2.2 is sharp with extremal function

$$f(z) = (z-\omega) - \frac{1-\alpha}{k(k-\beta)(1-d)^{k-1}}(z-\omega)^k. \tag{2.7}$$

□

**Corollary 2.4.** *Let $f(z) \in T(\omega)$ be in the class $K(\omega,\alpha)$. Then we have*

$$a_k \leq \frac{1-\alpha}{k(k-\beta)(1-d)^{k-1}} \tag{2.8}$$

Equality in (2.8) holds true for the function $f(z)$ given by (2.7).

**Theorem 2.5.** *Let $f(z) \in H(\omega, \alpha)$ and $f(z) = (z-\omega) - a_2(z-\omega)^2 - \ldots$ for $0 \leq \alpha < 1$, and $\omega$ is a fixed point in $E$. Then*

$$|a_2| \leq \frac{-2(1-\alpha)}{1-d^2} \tag{2.9}$$

$$|a_3| \leq -\left[\frac{(1-\alpha)}{(1-d^2)(1-d)} + \frac{2(1-\alpha)^2}{(1-d^2)^2}\right]$$

$$|a_4| \leq -\left[\frac{2(1-\alpha)}{3(1+d)(1-d)^3} + \frac{2(1-\alpha)^2}{(1-d)(1-d^2)^2} + \frac{4(1-\alpha)^3}{3(1-d^2)^3}\right].$$

*Proof.* Let us define

$$p_\omega(z) = \frac{\frac{(z-\omega)f'(z)}{f(z)} - \alpha}{1-\alpha} \tag{2.10}$$

That is,

$$(z-\omega)f'(z) = f(z)\left[1 + (1-\alpha)\sum_{k=2}^{\infty} B_k(z-\omega)^k\right] \tag{2.11}$$

On comparing the coefficient in (2.11) the results follow.

Following the earlier investigations of Goodman (Goodman, 1957) and Ruschweyh (Ruscheweyh, 1981), we define the $\delta$-neighborhood of function $f(z) \in T(\omega)$ by

$$N_\delta = \left\{g \in T(\omega) : g(z) = (z-\omega) - \sum_{k=2}^{\infty} b_k(z-\omega)^k,\ \sum_{k=2}^{\infty} k(1-d)^{k-1}|b_k| \leq \delta\right\} \tag{2.12}$$

and in particular, for the identity function

$$e(z) = (1 - \frac{\omega}{z})z \tag{2.13}$$

we immediately have

$$N_\delta(e) = \left\{g \in T(\omega) : g(z) = (z-\omega) - \sum_{k=2}^{\infty} b_k(z-\omega)^k,\ \sum_{k=2}^{\infty} k(1-d)^{k-1}|b_k| \leq \delta\right\}. \tag{2.14}$$

□

**Theorem 2.6.** *$H(\omega, \alpha) \subset N_\delta(e)$, where $\delta = \frac{2(1-\alpha)}{(2-\alpha)(1-d)}$.*

*Proof.* Let $f(z) \in H(\omega, \alpha)$. Then, in view of Lemma 2.1, since $(k-\alpha)(1-d)^{k-1}$ is an increasing function of $k(k \geq 2)$, we have

$$(2-\alpha)(1-d)\sum_{k=2}^{\infty} a_k \leq \sum_{k=2}^{\infty}(k-\alpha)(1-d)^{k-1}a_k \leq 1-\alpha \tag{2.15}$$

which immediately yields

$$\sum_{k=2}^{\infty} a_k \leq \frac{1-\alpha}{(2-\alpha)(1-d)}. \tag{2.16}$$

On the other hand, we also find from (2.3) that

$$(1-d)\sum_{k=2}^{\infty} ka_k - \alpha(1-d)\sum_{k=2}^{\infty} a_k \leq \sum_{k=2}^{\infty}(k-\alpha)(1-d)^{k-1} \leq 1-\alpha \tag{2.17}$$

from (2.16) and (2.17), we have

$$\begin{aligned}(1-d)\sum_{k=2}^{\infty} ka_k &\leq (1-\alpha)+\alpha(1-d)\sum_{k=2}^{\infty} a_k \\ &\leq (1-\alpha)+\frac{\alpha(1-\alpha)}{(2-\alpha)} \\ &\leq \frac{2(1-\alpha)}{2-\alpha}\end{aligned} \tag{2.18}$$

$$\sum_{k=2}^{\infty} ka_k \leq \frac{2(1-\alpha)}{(2-\alpha)(1-d)} \tag{2.19}$$

which proved the theorem. □

## 3. Integral mean inequality

**Lemma 3.1.** *if $f$ and $g$ are analytic in $E$ with $f \prec g$, then*

$$\int_0^{2\pi} \left|g(re^{i\theta})\right|^{\delta} d\theta \leq \int_0^{2\pi} \left|f(re^{i\theta}\right|^{\delta} d\theta \tag{3.1}$$

*where $\delta > 0, z = re^{i\theta}, \omega = de^{i\theta}$ and $0 < r + d < 1$.*

Applying Lemma 3.1 and (1.2) we prove the following

**Theorem 3.2.** *Let $\delta > 0$. if $f(z) \in H(\omega, \alpha)$, then $z = re^{i\theta}, \omega = de^{i\theta}$ and $0 \leq d < r < 1$, we have*

$$\int_0^{2\pi} \left|f(re^{i\theta})\right|^{\delta} d\theta \leq \int_0^{2\pi} \left|f_2(re^{i\theta})\right|^{\delta} d\theta \tag{3.2}$$

*where*

$$f_2(z) = (z-\omega) - \frac{1-\alpha}{(2-\alpha)(1-d)}(z-\omega)^2. \tag{3.3}$$

*Proof.* Let $f(z)$ defined by (1.3) and $f_2(z)$ be given by (3.3). We must show that

$$\int_0^{2\pi} \left|1 - \sum_{k=2}^{\infty} a_k(z-\omega)^{k-1}\right|^{\delta} d\theta \leq \int_0^{2\pi} \left|1 - \frac{1-\alpha}{(2-\alpha)(1-d)}(z-\omega)\right|^{\delta} d\theta. \tag{3.4}$$

By Lemm A, it suffices to show that

$$1 - \sum_{k=2}^{\infty} a_k(z-\omega)^{k-1} \prec 1 - \frac{1-\alpha}{(2-\alpha)(1-d)}(z-\omega). \tag{3.5}$$

Setting

$$1 - \sum_{k=2}^{\infty} a_k(z-\omega)^{k-1} = 1 - \frac{1-\alpha}{(2-\alpha)(1-d)}h(z) \tag{3.6}$$

From (3.6) and (2.1), we obtain

$$|h(z)| = \left| \sum_{k=2}^{\infty} \frac{(2-\alpha)(1-d)}{1-\alpha} a_k (z-\omega)^{k-1} \right| \tag{3.7}$$
$$\leq |z-\omega| \sum_{k=2}^{\infty} \frac{(k-\alpha)(1-d)^{k-1}}{1-\alpha} a_k$$
$$\leq |z-\omega| .$$

This complete the proof. □

## References


Acu, Mugur and Shigeyoshi Owa (2005). On some subclasses of univalent functions. *Journal of Inequalities in Pure and Applied Mathematics* **6**(3), 1–14.

Goodman, A.W. (1957). Univalent functions and analytic curves. *Proc. Amer. Math. Soc.* **8**(3), 598–601.

Kanas, S. and F. Ronning (1999). Uniformly starlike and convex functions and other related classes of univalent functions. *Ann. Univ. Mariae Curie-Sklowdowska Section A* **53**(53), 95–105.

Oladipo, A. T. (2009). On subclasses of analytic and univalent functions. *Advances in Applied Mathematical Analysis* **4**(1), 87–93.

Ruscheweyh, Stephan (1981). Neighborhoods of univalent functions. *Proc. Amer. Math. Soc.* **8**(3), 521–527.

Wald, J.K. (1978). *On starlike functions*. Ph.D Thesis, University of Delaware, New Ark Delaware.

**Theory and Applications of Mathematics & Computer Science**

(ISSN 2067-2764)
http://www.uav.ro/applications/se/journal/index.php/tamcs

Theory and Applications of Mathematics & Computer Science 1 (2011) 74–79

# An Overview on Software Reconfiguration

Robert Szepesi[a], Horia Ciocârlie[a,*]

[a]*Computer and Software Engineering Department "Politehnica" University of Timişoara*
*V. Pârvan street 2, 300223, Timişoara Romania.*

**Abstract**

Dynamical software reconfiguration represents a major direction in nowadays research due to its promise of providing faster solutions to ever changing problems by adding more flexibility to any given software solution at the cost of processing power, cost which given the relentless progress made by hardware manufacturers is becoming insignificant. This paper was designed as a complete overview of the software reconfiguration paradigm, looking at it from all the relevant angles - pros and cons, where to use and where not to use, challenges and solutions in implementation.

*Keywords:* Software reconfiguration, Reconfigurable software architecture, Design paradigms, Reconfiguration challanges.

## 1. Introduction

If anyone was to ever ask what was the most defining element of our times, the answer would be speed. The need for ever increasing speed is reflected in everything around us, especially in production processes, and even more so, in the software industry.

Throughout the years, the ever changing needs of the clients have prompted the software engineers to deliver software solutions at an always increasing rate, thus creating the need for the optimization of the entire creation process. First, we decided not to use chunks of code but mold those chunks into entities - classes - that could later be reused for other projects, then the classes became components, and eventually the question came what if we could reuse entire applications?

And so the idea of reconfigurable software solutions appeared - applications that could adapt themselves to a dynamic work environment without requiring the software manufacturer to change the source code in any way.

Obviously, the road from idea to actually implementing such an application is not without its bumps and that is why, throughout the length of this paper, we will discuss what exactly are these reconfigurable systems, why are they useful, where is it wise to use such a solution rather then implementing a standard application that could be patched up in time, what challenges are usually met when implementing such a solution, how they can be overcome and last, but not least, we will present a simple design that could serve for most common scenarios.

## 2. The What?

So what is a reconfigurable application? In short, it represents the next step on the evolutionary scale of software architectures, the key characteristic of a reconfigurable architecture being that it can adapt to the changes of either its

*Corresponding author
*Email address:* horia.ciocarlie@cs.upt.ro (Horia Ciocârlie)

environment, for instance the topology of a network, or the needs of its users, as long as these changes are within a predefined range, without there being any need for the system to be turned off for even a second.

However, just because a predefined range of accepted changes are allowed for the system to be configurable, it does not mean that the application in itself represents just a bundle of applications built into a single system, each configuration having assigned an internal application. The number of available configurations could, in theory, depending on the configurable parameters of the system, be infinite, but it would still require a precise set of allowed changes that the user can make - set of changes which are established through a set of parameters, each with its own set of accepted values.

In other words, a dynamically configurable software application is a generic application that can adapt its behavior, meaning data input, data processing and data presentation, at any given time , according to the values of a set of system parameters. Once having defined the system parameters, which should always be determined after carefully analyzing what is it that the system should be able to respond to, the reconfiguration process of the system can be easily defined through the change of the said parameters. However there are two ways in which the reconfiguration can be done: manual or automatic.

In the case of automatic reconfiguration, the system can determines the values of the parameters through the means of internal mechanisms that can range from sensors to simply broadcasting a message within a private network and waiting for a response. This is method of reconfiguration is used for systems that need to respond to changes occurred in the working environment of the application.

The manual reconfiguration is much simpler than the automatic one but implies the existence of a configuration panel of the system to which a (generally very limited) group of people can have access. The principle of the control panel is very simple: displays the system parameters and the allows the user to change their values, while making sure that the system can still act based on the new values.

## 3. The Why?

So why use dynamically configurable applications? On one hand there are systems that simply work in such dynamic environments that there is just no other solution, but for the most part, self configuration is implemented because of a number of benefits it brings to both the application per se and the software production process as well. The process of producing the software in itself is enriched by the fact that, while building a system that can later on be configured in various ways can take a longer while than building a normal application, future applications need not be created from scratch, they will simply require a proper configuration, thus saving great amount of time in the long run - this is the case of software product families, where an entire class of applications can be abstracted into a single configurable system; for more details on application families, see the following sections. When it comes to the application itself, due to the dynamic nature of the system, self configuration allows the users to perform online upgrades and even extend the application's functionalities with additional services. When talking about distributed systems, implementing self configuration capabilities into a system is almost a necessity due to the unreliable nature of a large span network which innately brings forth the need to make sure that the system survives the disconnection of one or more of the system nodes, in which case the responsibilities of one node need to be passed to others and as such, the very form of the system, as it would be represented on a graphical representation of the network, needs to be able to change while the system is running. Last, but not least, application reconfiguration becomes useful when the data that the software either produces or simply stores needs to be presented in an way that is either not unique, or simply variable in time. In these cases, the application usually allows its users to define and store, at runtime, both the ways in which the application can be presented to its users and the rules by which a certain presentation mode will be chosen over the other. Needless to say, while these are the main reasons for which reconfigurable applications are being created, they are not mutually exclusive. On the contrary, due to the ever expanding horizon of the software industry, it is expected that future generations of applications will include reconfiguration capabilities on all the levels described in this section.

## 4. The Where?

It should be obvious by this point that dynamic reconfiguration is not suited for any kind of application. For instance, one should not implement reconfiguration capabilities into a video game, where the software has one well

defined purpose, that does not change in time, does not operate on an unknown system configuration (every computer games has a list of system requirements attached to it, thus assuring the game developers that they know what kind of equipment will the game run on) and any other changes to the game can be done through an ulterior patch that need not be applied while the game is running. It is clear that, in such a case the application does not require reconfiguration features and implementing them would be not only pointless but also a great waste of time.

So where is it that it pays off to implement reconfiguration capabilities into the software? In general, at least one of the following questions needs to be answered with an affirmative answer before reconfiguration capabilities will be implemented into the software product:

- Is the targeted environment unstable, prone to frequent changes or simply unreliable?
- Is the way in which data is processed likely to change frequently or does data need to be processed in a different way depending on the system status?
- Is the way in which the application presents itself going to change in time?
- Will it save time in the future if the system is implemented as a general abstraction of a common problem? (to be read implemented as a family product)
- Will the system require self tuning based on self observation of execution, failure and recovery in order to provide the quality of service required?

As such, applications that usually make use of dynamic reconfiguration are either applications that run on highly dynamic equipment or applications that are part of a larger family of software products, or both. It is generally easy to determine when reconfiguration needs to be implemented due to the environment changes that the application needs to adapt to, but how does one decide if a certain application should be implemented as part of a more general family of products or an individual. In practice, the answer is always provided by the frequency with which the current problem to be solved is met. Truth be told, it is the old problem of when to create a new function/method for a piece of code versus just using that piece of code. As we all know, whenever a given piece of code, as small as it may be is expected to be used more than once, it should be factored out and implemented as a different method. If that piece of code appears in more than one places throughout the source of a component, but with some variations, the corresponding method should be parameterized so that calling the method can allow for the code to be used in all its original forms. So how do we translate this when dealing with entire applications? Let's take for instance the case of a simple management application. Most of this kind of applications use the same principle, they have a database in which they store various entities, usually products that are either being sold or produced, and an interface through which these entities are managed. The only difference between most of these applications is what the entities are, how many types of entities are there to be handled and how the data is presented to the user, and so, instead of creating a unique application each time, one would rather implement a family of applications: a template that could then be configured at run time to act as any of the applications that would've been implemented separately through the use of certain parameters.

However, it should be mentioned, that while dynamically reconfigurable software architectures are most commonly met in business and management environments, they can be successfully applied to almost all other domains. For instance, (Mun et al., 2006) describes the **Fr**actal **M**anufacturing System (FrMS) as a successful implementation of a reconfigurable software application designed specifically for the manufacturing environment.

## 5. The Why not?

However, the question emerges: if dynamically reconfigurable applications are so great how come the market isn't full of them yet?

The truth is that, while having reconfigurable features incorporated into an application does bring a great amount of benefit to both the software producer and user, the challenges that appear while trying to implement such a system are not few and they're also not slight.

One of the main issues is that there is no real formal method of implementing the reconfiguration of an application. In order to truly deal with reconfigurable software architectures a formal method to describe software architectures

and the changes that these need to go through has to come forth. It would be a lie to say that attempts to create such a formal method to describe the intricacies of software reconfiguration have not been made but the truth remains that most of these solutions are limited, especially when it comes to representing hierarchy and modeling context-aware systems (applications that behave differently depending on certain environment parameters, such as the geographical location of the user).

As such, without proper tools that will allow them to concretely express and document the idea upon which they intend to build, not many software engineers are inclined to take this path.

There is however hope for the future. In their paper, (Chang et al., 2008), Mao et al. propose a new formal method that could prove to fill this much needed role. Their idea is to use and extend bigraphs to describe reconfigurable software architecture, the basic idea being that bigraphs can easily survey both static and dynamic architectures through the use of graphic elements and term languages.

In addition, (Gomaa & Hussein, 2004) fully describes a few approaches for designing reconfiguration patterns, such as the *master-slave reconfiguration patterns or the centralized reconfiguration pattern*, for each of them providing modeling examples through the means of state diagrams.

Beyond the problem of having no precisely standardized way of modeling the application concept, implementing the reconfiguration capabilities of an application is not always as easy as it might appear.

Let us imagine the situation of a large distributed application, consisting of multiple processes running on different computers spread across a network. In such a case, reconfiguring the application becomes a problematic due to the concurrent nature of the system. One must always make sure that all the processes are ready to be reconfigured, which could mean that certain processes might need to be put on hold for a while or even shut down completely while the reconfiguration is performed, thus destroying the illusion of ”on the fly” reconfiguration (Mun et al., 2006). Same situation applies for when a system uses redundant servers to ensure high availability of service/data. In such cases, in order to assure the consistency of the system, one must make sure that all the system components are reconfigured at the same time, but what if one of the servers is performing an action while the server that receives the reconfiguration request is ready? Evidently one must, just like in the case of concurring processes, one must again make sure that all servers are available for reconfiguration.

A simple solution to this issue would be to simply appoint a ”reconfiguration center” somewhere within the system, perhaps even a dynamic one - the network node that received the reconfiguration request (or generated it itself after various parameter measurements) can become the reconfiguration server - which broadcasts a RECONF_ REQ message and waits until all other participants respond with a RECONF_ READY. However, while simple this solution is not always practical. What if the application is spread over a large network, is comprised of a large number of nodes, the whole request/acknowledge mechanism might generate too big of a traffic compared to how many times certain components of the system will not be ready for a reconfiguration.

A solution to this particular problem is proposed in (Whisnant et al., 2003), which suggests that dataflow dependencies among operations can be derived for any given configuration of the system if input and output signatures are created based upon the variable access patterns of the code blocks. Thus, any other proposed reconfiguration can be formally evaluated before applying it to the system, in order to determine if the new mapping of the operations to the code blocks disrupts the any dataflow dependencies currently existing within the system. Once these tests have been run on all possible configuration, using the results, either the system administrator or a part of a system itself (automated check) could use the data provided by the mechanism, which the designers named **ARMOR** (**A**daptive **R**econfigurable **M**obile **O**bjects of **R**eliability), and determine whether the system needs to use a synchronize routine before applying reconfiguration or not. Evidently, the issue risen by synchronicity is just an example, the principle presented in (Whisnant et al., 2003) can be whenever it needs to be determined if certain operations need to be executed before a configuration can be applied to the system and even to determine if the current system can actually support the configuration at hand.

## 6. The How?

And finally, how is the whole concept of software reconfiguration implemented in an application? While complicated in practice due to the complexity of any specific software architecture, the principle is relatively simple.

Unfortunately, changing the very blocks of an application in a transparent way at run time is a rather complicated matter, and as such, highly unlikely to be approached by many developers. One of the research directions receiving

a lot of attention regarding this matter is software dynamic translation, which represents a technology that allows the modification of an application's instructions while it is running, and strides are being made towards progress. For example, (Scott et al., 2003) describes Strata as a cross-platform infrastructure for building software dynamic translators, developed with the main goal of not only simplifying the task of initiating a new project in SDT, which the authors themselves describe as a rather difficult one, but also to promote the adoption of the SDT technology.

Imagine the components of a given application as floors of a special building. Now, the way in which the application works, is dictated by how these components are linked with one another, in other words any configuration of an application can be expressed by expressing the links between the application components. In the case of a building the floors are linked through stairs (or elevators, but, for the sake of the analogy, let's assume there are no such things as elevators). The layout of the building is expressed through the layout of the stairs. So how does a reconfigurable application work? Simple. Imagine that these stairs, like the ones presented in J.K. Rowlings books famous castle Hogwarts, can move! Every time these stairs change their location, linking different levels of the building among themselves, a person needs to follow a different path to get from point A to point B, even though points A and B have not really changed their location.

Similarly, every time a new configuration is applied to the software architecture, the links created between the software components are torn apart and new ones are created. Thus, the next time data will need to be processed by the application, like the person running through the floors of the building, it will have to follow a different path, being processed by different components and, as such, creating a different dataflow for each configuration.

Changes from one configuration to another do not have to alter the original dataflow to its very foundations, however. While it is true that the size of the components among which the links can be broken can be as small as a single variable or as huge as entire modules of the system, what it all boils down to is the fact that the way in which the basic elements of the software architecture combine changes for each configuration of the system.

But how exactly can these data and work flows actually be remodeled while the system is running? Given the extreme variety of the ways in which an application can be implemented (both in terms of coding language, each language having its own pros and cons, and in terms of methods used - for instance, an application can be implemented both by using servlets or javabeans) the ways in which the reconfiguration is just as diversified.

One of the oldest ways in which reconfiguration features were implemented in software systems is to implement the application as a set of self deployable modules that could be loaded at runtime into the main module of the application. Thus, the main module, usually representing nothing more than a figurative set of slots for the actual business modules to be inserted into, would determine once launched which are the modules are required in the current configuration of the application and then load only those modules. However, despite the amoeba-like look and feel of the application, this method of integrating reconfiguration into a software architecture proved to be inefficient due to the fact that the application is very limited when it comes to the number of valid configurations accepted. The fact is that this method represents nothing more than a fixed set of possible problems linked into a single interface, without allowing the set of problems that that system can solve ever expand without having the programmer go back into the code and creating a new module including a new solution.

The idea behind software reconfiguration is that once written it should be able to be used to solve a number of problems as large as possible, and most importantly, it should allow even its creator to be at some point surprised of what his application could be used for.

The truth is, that a reconfigurable software architecture is supposed to be nothing more, but nothing less either!, than the software equivalent of an **FPGA** (**F**ield **P**rogrammable **G**ate **A**rray) circuit, in the sense that it should obviously have a limited amount of resources available for the end user of the application, but, like an FPGA, it should allow the user to use these resources in any way it sees fit.

In fact, given nowadays technology, one could even implement a software equivalent of the FPGA that could, in a sense, offer limitless resources to its configuration administrator. The idea behind this statement is that, no matter what language the application is implemented in, one could declare dynamic classes by implementing an "entity manager" which could instantiate entities (to be read classes) with an unknown structure and eventually allowing the user, or better said a very restricted group of users, to establish these structures at runtime. This would in fact be a great example of the "two-level programs" described in (Kamin & Clausen, 2001), in which one level of the application (the entity manager part) generates the second level of the application, the actual part that the better part of the user will be interacting with.

Let's take an example to better illustrate this principle - the data management application we referred to in the

previous sections, when discussing the software product families. One could in theory declare, in code a class *CEntity* that employs an variable number of attributes implemented in a yet another variable number of classes derived from a base *CAttribute*. Thus, once the common functionalities of an entity have been implemented without caring what the attributes actually represent, a control panel could also be implemented to allow the system administrator to specify an unlimited number of entities that it would like to keep track of, each entity with its own particular structure.

Furthermore, assuming the previous principle was implemented in a well thought through manner, there is no reason why the system developers could not go a step further with the concept and allow the administrators to also define the way in which the entities interact among themselves, the way in which the data is presented, processed an so on, the whole process being based on the dynamicity brought by the Entity manager.

## 7. Conclusions

In conclusion, reconfigurable software architectures represent the path to be followed in order to produce high quality applications in the shortest amounts of time. While this kind of architecture does not match each and every application it can be incorporated in most data processing systems.

The idea of reconfigurable software is, however, still relatively young and, as such, challenges are met when trying to implement such an architectures due to the lack of formal methods to design and model the architectures. When trying to implement reconfiguration on large scale, on highly distributed systems, the complexity of the problem is highly increased due to accessibility, concurrency and consistency issues.

However, as more and more systems are progressing on the reconfigurable path, solutions emerge, such as those presented in the previous sections of this paper: a formal method to model the reconfigurable architecture based on bigraphs has been devised and has become quite popular, and the principle proposed by **ARMOR** could prevent most concurrency and accessibility issues when trying to apply a reconfiguration to a highly distributed system.

Last, but not least, in the last section of this paper, a design paradigm has been presented to show the true potential of the reconfigurable software architectures, which is, virtually, unlimited.

## References

Chang, Z., Mao, X., & Qi, Z. (2008). Towards a formal model for reconfigurable software architectures by bigraphs. In *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)* WICSA '08 (pp. 331–334). Washington, DC, USA: IEEE Computer Society.

Gomaa, H., & Hussein, M. (2004). Software reconfiguration patterns for dynamic evolution of software architectures. In *WICSA* (pp. 79–88).

Kamin, S., & Clausen, L. (2001). Dynamically reconfigurable software components, .

Mun, J., Ryu, K., & Jung, M. (2006). Self-reconfigurable software architecture: Design and implementation. *Comput. Ind. Eng.*, *51*, 163–173.

Scott, K., Kumar, N., Velusamy, S., Childers, B., Davidson, J. W., & Soffa, M. L. (2003). Retargetable and reconfigurable software dynamic translation. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization* CGO '03 (pp. 36–47). Washington, DC, USA: IEEE Computer Society.

Whisnant, K., Kalbarczyk, Z. T., & Iyer, R. K. (2003). A system model for dynamically reconfigurable software. *IBM Syst. J.*, *42*, 45–59.