Particle Swarm and Genetic Algorithms applied to the identification of Induction Machine Parameters

L. Baghli, A. Rezzoug.
Groupe de Recherche en Electrotechnique et Electronique de Nancy
GREEN-UHP CNRS UMR 7037
Université H. Poincaré, BP 239, 54506 Vandœuvre-lès-Nancy, France
Tel. +33 3 83 68 41 32 Fax. +33 3 83 68 41 33
lotfi.baghli@green.uhp-nancy.fr
http://www.green.u-nancy.fr

Keywords
Particle Swarm Optimisation, Genetic Algorithm, Identification, Induction Motor.

Abstract
This paper deals with the identification of induction machine parameters using Genetic Algorithms (GA) and Particle Swarm Optimisation (PSO) method. Thanks to these algorithms, optimisation is carried without the evaluation of the gradient. The optimisation is carried using current and speed experimental curves of a direct start-up. Convergence speed, algorithms tuning and local-minima problems are discussed. Results during the evolution process are presented as well as a comparison with the experimental results.

Introduction
Getting model parameters is often a hard problem when we have to simulate or to control a process. A direct identification is not always possible and hardly relays on the process considered. The models we use in vector control simulation or operation and also in transient studies of induction motor needs full parameters knowledge [1],[2].
It is impossible to have a vector of parameters that ideally "works" for all operating points. Hence, we have to choose a "mean" vector that matches well for nearly all operating modes (at nominal saturation and for multiple loads). Therefore, for transient motor operation, as in the case of electric drives, we need more information about these parameters. This information is self contained in the speed and stator current experimental curves of a direct start-up [3].

An optimisation process has to be started in order to get machine parameters. The comparison between experimental and simulation speed and current curves is considered. A quadratic error factor is built thanks to a "weight function" that takes more into account the transient and steady state operating range than the sub-transient one. If the parameters are correct then this error would be very small. It will never reach zero because the experimental machine can not be represented by a unique model with constant parameters.

The aim of this paper is to discuss two multi agent optimisation algorithms: Genetic Algorithm (GE) and Particle Swarm Optimisation (PSO). In this study, we will apply them to get induction machine parameters following the method described above.

Modelling
The induction motor is modellised thanks to a *dq* classical model and the whole system including the control and the optimisation algorithms are implemented on a software "MASVECT" developed in C++ at our laboratory [3].
We will focus on the genetic and the particle swarm optimisation algorithms.

Genetic algorithms (GA)

Genetic algorithms are nowadays well known in many optimisation processes and have shown their effectiveness even in electrical engineering and machine parameters identification [4],[5].

Genetic algorithms can be classified as non-linear adaptive networks [6]. These networks are composed of a large number of elementary units or agents which are in our case chromosomes (also called strings). These agents process the information in a distributed and parallel scheme. They interact between themselves non-linearly without a central supervisor. In an environment that is able to give them feedback, the agents and the way they interact are modified through "operators" in order to adapt the global system to the environment and to improve its response.

The analogy with biological processes gives a clearer idea on how a genetic algorithm operates. Genetic algorithms are baseModelling Multiple Saliencies in Rotor-Faulty Induction Machine :
Rotor Position Estimation
d on natural selection and evolution mechanisms for optimisation purposes. A genetic algorithm makes its population evolve using these mechanisms.

The optimisation process starts with a random population composed of a large amount of individuals (Figure 1).
Each individual represents a vector of parameters to be optimised. The vector chosen for our application is :
$[R_s, \tau_s, \tau_r, \sigma, J, a_1, a_2, a_3]$

R_s, τ_s, τ_r, σ are the electromagnetic parameters whereas J, a_1, a_2, a_3 are the mechanical ones that represent the inertia and the coefficients of the polynomial load torque.

We choose to compare the speed and the current of the induction motor on a direct start-up with the ones given thanks to a simulation model using the vector to be evaluated.

Each individual of the population is evaluated by performing the simulation of the start-up, using its parameters. Then, a "fitness" is attributed to the individual. It is the inverse of a cost function that is based on the quadratic error between the experimental and the simulation curves :

$$E_r = \int \left((\Omega sim - \Omega exp)^2 + \lambda (I_{as} sim - I_{as} exp)^2 \right) dt$$

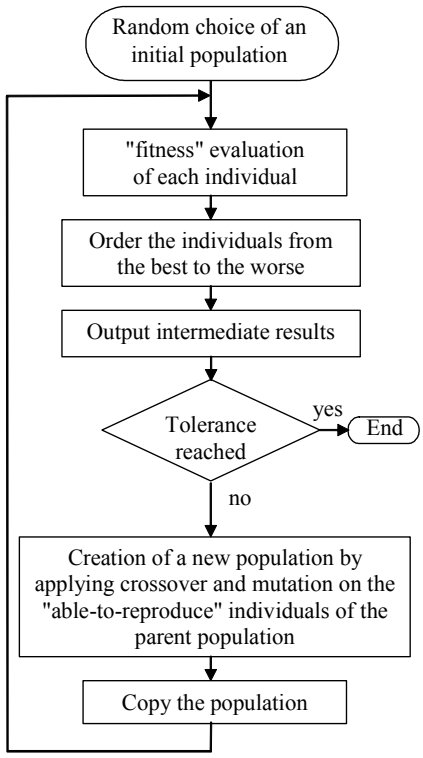


Figure 1 – Genetic algorithm flowchart

After that, the individuals are classified from the best to the worse. Being at the top increases its chance to be selected for the reproduction. This reproduction phase takes place in different steps of mutation and crossover. We then obtain the new population.
The mutation consists in modifying randomly one or more genes of a chromosome (individual characteristics), whereas the crossover occurs while inheriting genes from two parents at the same time.

Different ways to proceed exist. Each user conceives its own variant that he considers the best to fit his problem. We present, here after, the one we developed for our application.

The coding of individuals was originally done with strings of binary digits representing the genes put one aside the other [7]. Now real code algorithms are used and give more flexibility to manipulate and interact with the self-meaning vector of parameters without having to deal with a binary to decimal transformation.
The real code algorithms also offer the possibility of combination with classical optimisation methods and leads to hybrid methods [8].
After multiple tries, we retained a population of 55 individuals. This number is a compromise between, the optimisation-convergence speed, the ability to get out of local minima and the computation time. If the population is large, the convergence will result within few generations but the computing time of the optimisation process will be longer.

The crossover and mutation rates are not computed from a probability [9] that an individual has to mute or crossover but are fixed by partitioning the population. Different sub-populations are set with specific behaviour among the ordered individuals. Table 1 summarises the retained configuration.
The strongest individual, in term of fitness, is copied from generation to generation to avoid losing the best vector.
The sub-population from individual 1 to 9 (Table 1) corresponds to individuals generated from muting randomly selected individuals that are considered as able to reproduce (selected from the 10 best previous individuals). A maximum deformation factor for the mutation is fixed to 0.1 %.
Various factors are attributed to the sub-population, making it specialised in generating strong individuals at specified phase of the optimisation process.
The crossover implemented, generates 10 individuals for each generation. Each offspring is obtained thanks to the parameters (genes) of two parents chosen randomly among the 15 best individuals.

The genetic population is divided into classes like human beings or animals. Not everyone can copulate with everyone. It is a philosophical point of view that we won't discuss but, indeed, it exists and lot of species and societies use it and make it a rule. The strongest (in respect to an established rule) wins the best female.

To summarise the process, we can say that the strongest individuals have more probability to reproduce and will have more descendants than the others. Each chromosome is constituted of smaller elements called characteristics or genes. The goal is to find the optimal combination of these elements, which gives the highest fitness. At each iteration (population generation), a new population is created from the previous one. The process ends when the error falls under a fixed tolerance.

Step	**Number of individuals**	**Position**	**Composition of the sub-population**
Best	1	0 to 0	Copy : we keep the best individual
Mutation	9	1 to 9	Random mutation of individuals chosen within the 10 firsts with a maximum factor of 0.1 %
	5	10 to 14	Random mutation of individuals chosen within the 15 firsts with a maximum factor of 10 %
Crossover	10	15 to 24	Random crossover of two individuals chosen within the 15 firsts, inheritance is done equally
Strong mutation	10	25 to 34	Random mutation of individuals chosen within the 35 firsts with a maximum factor of 50 %
Soft mutation	20	35 to 54	Random mutation of one gene of the first individual with a maximum factor of 1 %

Table 1 Sub-populations details

Particle swarm optimisation algorithm (PSO)

Particle swarm optimisation has roots in bird flocking, fish schooling and swarming theory [10],[11]. Scientists tried to simulate animals' social behaviour and their movements. By observing bird flocking choreography, they tried to discover the rules that enabled large numbers of birds to flock synchronously, often changing direction suddenly, dispersing and regrouping… It appears that modelling the birds as individuals like cellular automates without taking into account group social behaviour gives inexact results.
Models have to take into account inter-individual distances for example to maintain synchronous flocking. So, it is a permanent search of optima that is performed by each individual and that reflects reality.

Presenting PSO is rather simple if we consider birds flocking while searching for their food. The birds will be the "agents" of our optimisation algorithm and they behave in the same manner [12].
The PSO can be used to optimise many processes by using a cost function and an adapted configuration. It is particularly used for neural networks learning processes [13]. We will adapt it to our case, but first, we have to explain the principle of PSO.

Let's take an example with a population composed of 5 birds called agents. These agents "move" on a x-y plane. They have to find a privileged position; place where there is food, let's say point (0,0). They can evaluate if they are near it or not and they can also communicate between each other.
For this simple example, the cost function is: $Eval_i = x_i^2 + y_i^2$
The agents keep information of their personal best position and its value ($pbest_i$), which corresponds to the lowest evaluation ($Eval_i$) in the searched places. They also have access to the general best position and its value (g*best*), that is the overall best place found by one member of the swarm. While exploring the search space, the agents adjust their speed, in direction and amount. At each step, called epoch, we evaluate the cost function and we calculate the new speed for each agent and the new position. The way that the agents change their speed gives different variants of the PSO algorithm. It has been shown that the speed must be updated according to the present position, the personal best and the general best positions [10]. We can also add knowledge of the current speed, so the new speed at each epoch for both directions *x* and *y* become:

$$\begin{cases} new\_v\_x_i = (rand + w) \cdot v\_x_i + rand \cdot pincr(pbest\_x_i - x_i) + rand \cdot gincr(gbest\_x - x_i) \\ new\_v\_y_i = (rand + w) \cdot v\_y_i + rand \cdot pincr(pbest\_y_i - y_i) + rand \cdot gincr(gbest\_y - y_i) \end{cases}$$

Where "*rand*" is a random number, "*w*" a weight coefficient introduced to give some inertia to the movement, so the "bird" can not change direction abruptly. "*pincr*" and "g*incr*" are increment amounts by which the speed can be changed according to the personal best position and the general best one found by the swarm.

We programmed the PSO algorithm according to the flowchart presented on Figure 2.

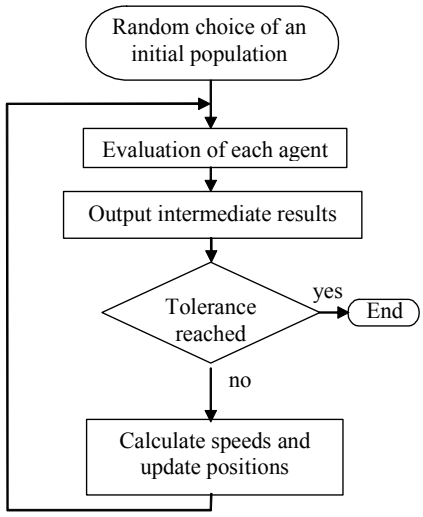
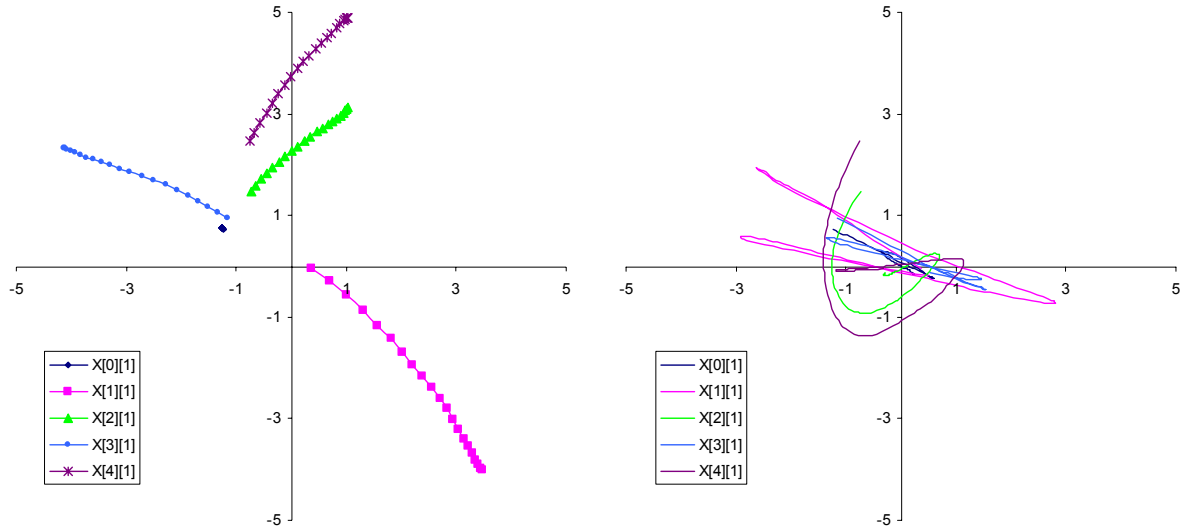


Figure 2 – PSO algorithm flowchart

On Figure 3, we present the successive positions of the 5 agents over the search plane. From epoch 0 to 20, we notice that the agent (curve X[0][1]) that was initially but randomly closer to the (0,0) optimum point does not move a lot because he has a low personnel best and that there is no better solution than his. At epoch 20, the agent of curve X[1][1] passes next to the optimum point but at a high speed, so he continues on his way.

The agent notices and informs all the others about his new personnel best, so the agents X[0][1] will head to the new global best and the other agents will change their directions and bend them (X[2][1] and X[4][1]). After many epochs, we notice that some agents are close to the optimum point while some of them continue over-flying the target and exploring the neighbours.



From epoch 0 to 20 From epoch 20 to 120

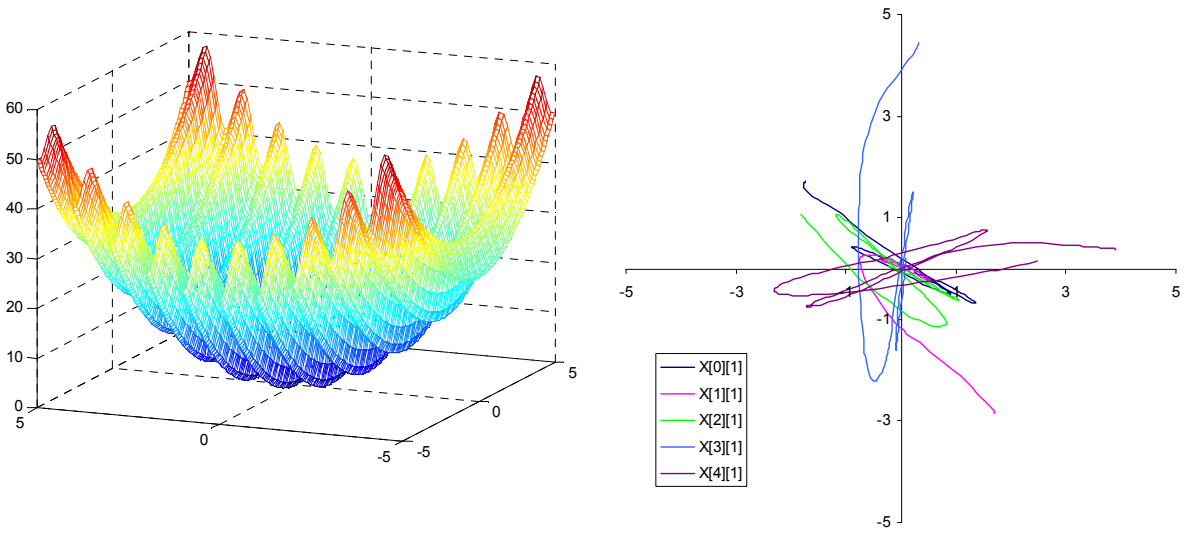
Figure 3 – PSO

If the cost function presents multiple local minima and an absolute one, like the *m*-dimensional Rastrigin function [13]: $Eval_i = 3m + \sum_{j=1}^{m}\left(x_{ij}^2 - 3\cos\left(2\pi x_{ij}\right)\right)$

then the optimisation process could be harder. We have to increase the weight factor in order to have more agents flying over their optimum point and continuing the exploration over a wider area (Figure 4).

We can also split the population and choose "explorers" that have this feature while other birds will continue refining the research around the optimum.

Actually, we use a more important population of agents (10 to 50). Of course, this increases the time for the evaluation of one population but it can accelerate the convergence in term of generations or epochs.



Two-dimension Rastrigin function From epoch 0 to 120

Figure 4 – PSO of a Rastrigin function

Experimental and simulation results

Using GA

We carried out many identifications. The progression of each one differs from the other because we start with a random population and also because the mutation and crossover are done randomly. Figure 5 represents the origin of the best individual along the generation evolution. The rank and sub-population from which this individual comes, at each generation, are represented on the y-axis. In Figure 6, a zoom on the generations between 400 and 420 is performed. Horizontal lines delimit the sub-populations.
In fact, each sub-population is more or less solicited in each optimisation process phase. In the beginning of the optimisation, it is from here and there, that the best individual comes. There is however a small predominance of the sub-population "soft mutation".

The sharpen is the selection, the more often will the individuals issued from the crossover sub-population found themselves at the top of the chart.
At the end, when the system has converged to the optimal solution, the individual stemmed from the unaltered population stay often as the best, in the sense of the selection criterion.

Figure 7 shows the fitness evolution of the best individual.
In the first 1000 generations, we often notice step changes in the fitness of the best individual as well as in its parameters. These results are due to the random creation and alteration feature of some individuals. This method enhances the convergence speed and allows to get out of local minima.

We also show the evolution of the inertia moment and the electromagnetic parameters of the model according to the optimisation process (Figure 8 to Figure 10). Friction factors were also optimised in the same manner and are not represented here.

After full optimisation (Figure 12), one can notice that the speed and current responses of the simulation match the experimental curves, in both transient and steady state operation modes.

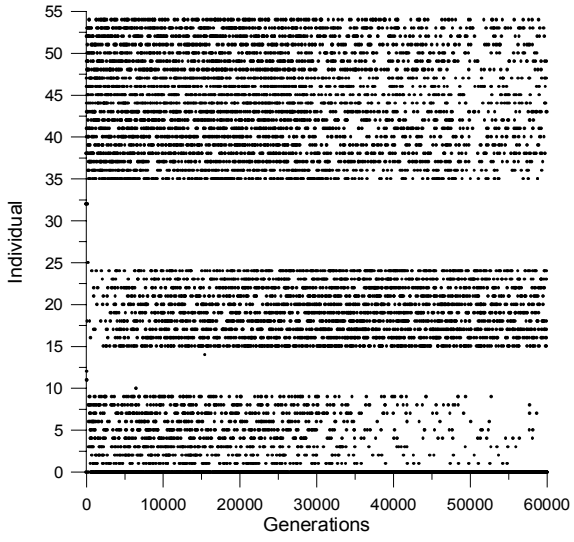


Figure 5 – Best individual position

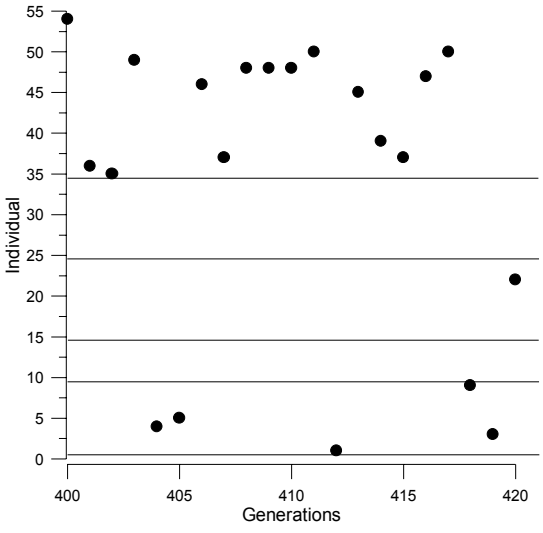


Figure 6 – Best individual position (zoom)

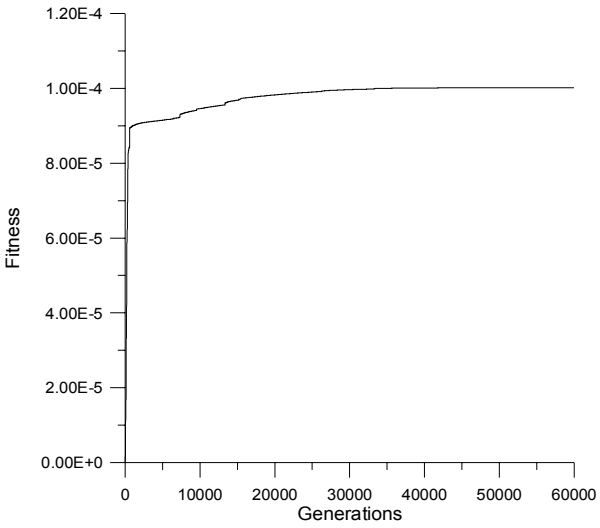


Figure 7 – Fitness evolution

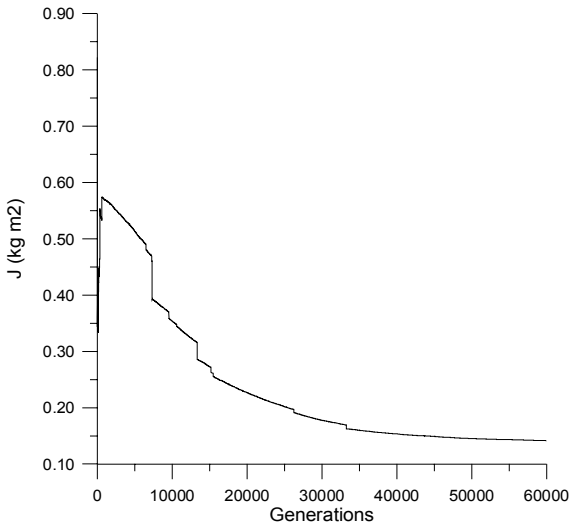


Figure 8 – Evolution of J

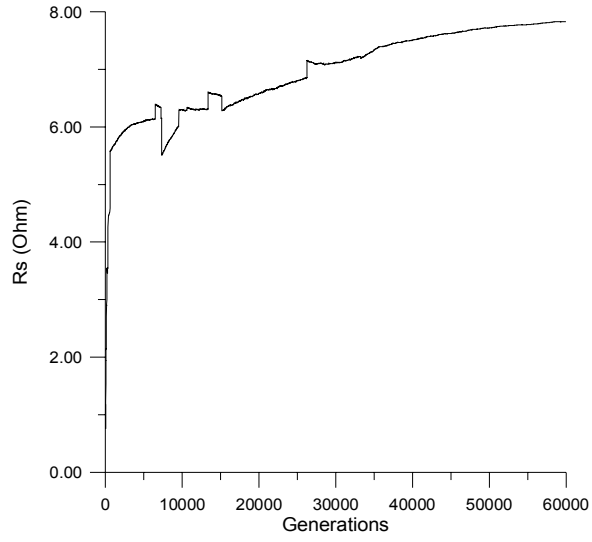


Figure 9 – Evolution of R_s

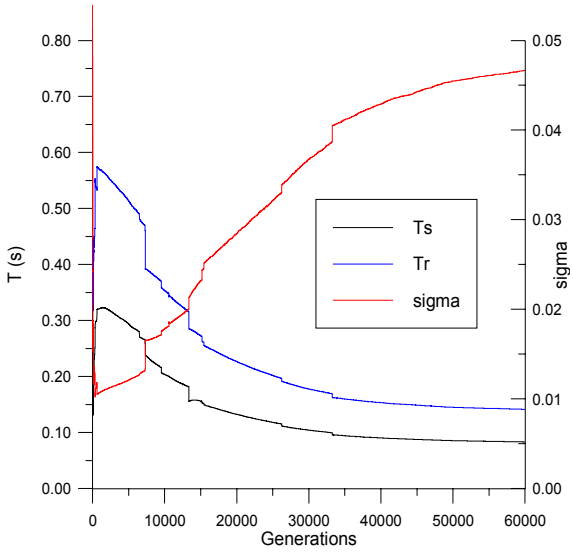


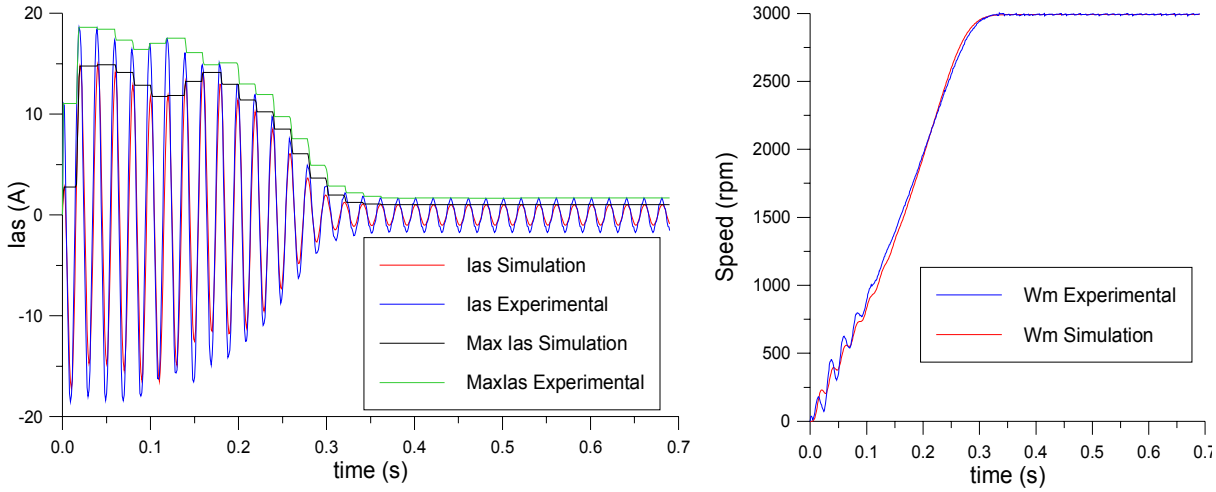
Figure 10 – Evolution of τ_s, τ_r et σ

Using PSO

For PSO, the evaluation process during optimisation is the same as the one of the GA. The difference resides in the way the agents find their way to the optimal vector.
It is harder to represent graphically the search space of the study since it is more than three dimension one. This also affects the robustness of the algorithm. The algorithm has more difficulties to get rid of local minima than it has on classical function optimisation.
We except finding better weight parameters and also correct swarm size to accelerate the convergence and prevent sticking in local minima. A way to consider is adaptive weight: if the swarm is too close to a possible solution, members of the swarm must go away to look for another "feeding point" and explore if there is another optimum elsewhere.

While in optimisation process, the software "MASVECT" can also output a comparison of the experimental curves in respect to the best optimum found at the moment.

We do evaluate multiple cost functions; both involve speed and current curves. We privilege transient and steady state operating points to find the "average" vector of parameters.
On Figure 11, we present an interesting result while optimising using the PSO algorithm and a "sliding max comparison" function. The cost function combines the speed and the envelope of the current curves.
As this envelope has to be computed while the simulation is in progress, we used a sliding window over one period to track the moving maximum. The function is less dependent on the phase shift due to an error of identifying the initial phase of the voltage prior to experimental signals acquisition.



Current curves Speed curves

Figure 11 – Intermediate results (epoch 80)

Comparison and prospects

It is very important to keep in mind that an optimal vector of parameters represents a machine, as well as it can, for a specified operating range. We can find as many vectors as we want, but if we want to obtain a set of parameters that gives satisfactory responses overall, we have to define a performance criterion over an operating area.
Therefore, on the start-up optimisation, we give more importance to the steady state and to the end of the transient start-up phase than we give to the beginning of the transient response. This is achieved by using a weight sigmoid function. We also favoured the match accuracy of the current curves in respect to the speed ones. The resulting cost function is given by:

$$E_r = \int \left(\frac{1}{1+e^{gA(gt0-t)}} \left[(\Omega sim - \Omega exp)^2 + \lambda (I_{as} sim - I_{as} exp)^2 \right] \right) dt$$

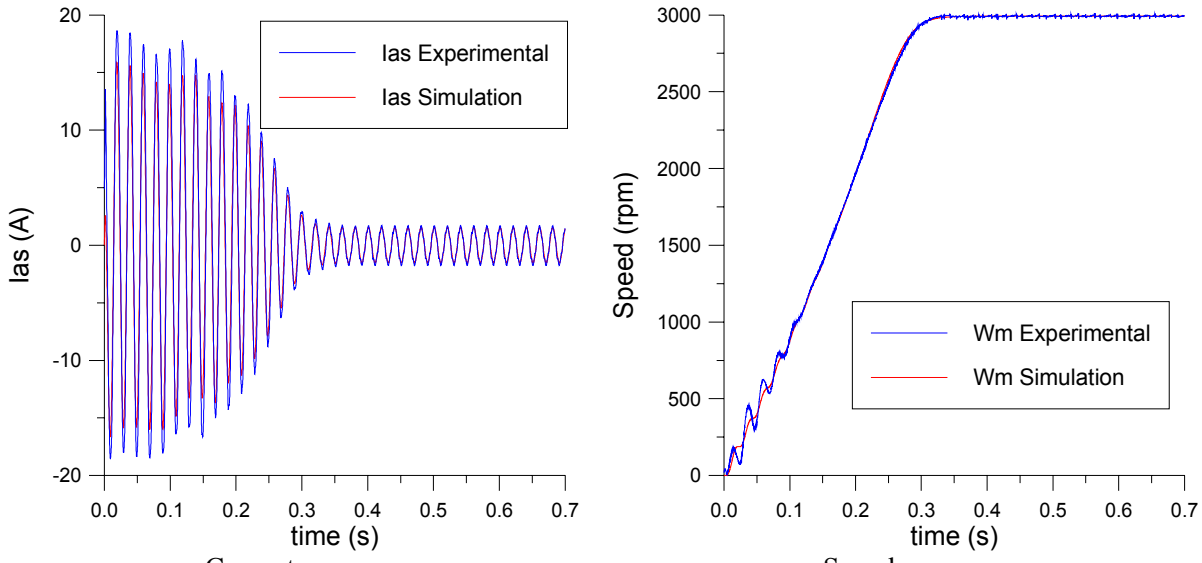
It is then logical to obtain an "optimal vector of parameters" which gives a less good response in the beginning of the start-up (Figure 12).

One can notice (Figure 5 to Figure 10), that the number of generations to achieve the convergence is high and that even when the fitness (Figure 7) has more or less reached its final value, the parameters continue to evolve particularly τ_s, τ_r and σ. However, if we study the evolution of the products $\sigma\tau_s$ and $\sigma\tau_r$, which are linked to the leakages, we notice, after 15000 generations, that there is quiet no evolution. This means, that the process is very sensitive to the leakage but not to the parameters taken separately. Consequently, we are able to write a simplified transient model of the machine exclusively with $\sigma\tau_s$ and $\sigma\tau_r$ instead of the parameters taken individually [3].
What characterises the motor is more its leakage, so it is important, in an optimising tool, that the three parameters τ_s, τ_r and σ, have to be optimised simultaneously.

From Figure 5 and Figure 6, we can notice that not all the sub-populations give individuals that become the best. It is a wrong idea to lower their importance. Indeed, one of the great advantages of genetic algorithms is the ability to go through local minima. Random mutation and a large number of individuals are necessary to confirm this quality.
Of course, the computation requirement is heavier and slows the optimisation process.
Another advantage of the genetic algorithm and particle swarm optimisation is that they do not require any function derivative.

It is interesting to investigate hybrid methods that allow an exploration of the search space prior to begin the fine optimisation. Simplex based algorithms seem adequate for such an application because they start with multiple vectors. Once we are close to the global minimum, Newton-Raphson like methods converge very fast to the solution. The problem is to obtain a derivative function.

In PSO, an agent always keeps its best position and information of the global best though it can fly away from them whereas in GA, the individuals hold the information because the information is self coded. Therefore, the PSO agents continue to explore the search space even if they have found a possible solution.
The PSO has shown a drawback in this study for Induction Machine parameter identification because the parameters must change at the same time to decrease or increase the leakage inductance for example. The probability that the three parameters τ_s, τ_r and σ, changes simultaneously in the best direction is low. So after a rapid rough convergence, the evolution is slower or seems stopped. GA overcomes this by having a sub population that is able to change one gene (parameter) within the best individual by a very small factor, thus, fine tuning the optimisation process.



Current curves Speed curves

Figure 12 – Identification on a direct start-up

Conclusion

Genetic algorithms and particle swarm optimisation method offer an interesting alternative to classical optimisation processes. Their great advantages are that they require no derivation and that they are less affected by local minima problems.
Although, when the cost function to evaluate is computing-time consuming, the genetic algorithm becomes very slow in respect to the PSO algorithm. Both algorithms suit well parallel computing since each evaluation of a vector is independent of another one.

References

[1]. Baghli, L.; Razik, H.; Rezzoug, A.; "Neuro-fuzzy controller in a field oriented control for induction motors", in *EPE Journal*, vol. 10, n° 2, pp. 21-26, August 2000.

[2]. Vas, P.; *Parameter estimation, condition monitoring and diagnosis of electrical machines*, Ed. Oxford University Press, 1993, 360p

[3]. Baghli, L.; *Contribution à la commande de la machine asynchrone, utilisation de la logique floue, des réseaux de neurones et des algorithmes génétiques*, Thèse de Doctorat de l'Université Henri Poincaré, January 1999.

[4]. Baghli, L.; Razik, H.; Rezzoug, A.; "Genetic algorithm for the identification of the parameters of an induction motor on start-up", *SEEPCI'99*, pp 132-138, 17-18 May 1999, Oujda, Maroc.

[5]. Accarnley. P.P.; Finch, J.W.; Da Silva, W.G.; "Tuning of a brushless DC drive speed controller with an on-line genetic algorithm", *EPE'99*,019.pdf, pp.1-9, 7-9 September 1999, Lausanne, Switzerland.

[6]. Renders, J. M. *Algorithmes génétiques et réseaux de neurones*, Ed. Hermès, 1995, 349p.

[7]. Pillay, P.; Nohan, R.; Haque, T.; "Application of genetic algorithms to motor parameter determination for transient torque calculations", in *IEEE Trans. Ind. Applicat.*, vol. 33, pp. 1273-1282, Sept./Oct. 1997.

[8]. Clerc, G.; Chouiter, D. R.; Beson, C.; Bellaaj-M'Rabet, N.; Rétif, J. M.; "Comparative study of identification methods for induction machines", *EPE'97*, pp 1.524-1.528, 8-10 Sept. 1997, Trondheim, Norway.

[9]. Bellaaj-M'Rabet, N.; Jelassi, K.; "Comparaison de méthodes d'identification des paramètres d'une machine asynchrone",in *Eur. Phys. Journal*, AP, vol. 3, pp. 71-80, 1998.

[10]. Kennedy, J.; Eberhart, R.; "Particle swarm optimization" in *1995 IEEE International Conference on Neural Networks Proceedings* (Cat. No.95CH35828), 1995, V4, pp 1942-1948.

[11]. Kennedy, J.; Spears, W. M.; "Matching algorithms to problems : an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator" in *Proceedings of the 1998 International Conference on Evolutionary Computation,* pp 78-83.

[12]. El-Sharkawi, M.A.; "Role of Computational Intelligence in Machine Diagnosis", *SDEMPED'01*, pp.11-18, 1-3 September 2001, Grado, Italy.

[13]. Van den Bergh, F.; Engelbrecht, A.P.; "Cooperative learning in neural networks using particle swarm optimizers", in *South African Computer Journal SACJ / SART*, n° 26, 2000, pp 84-90.